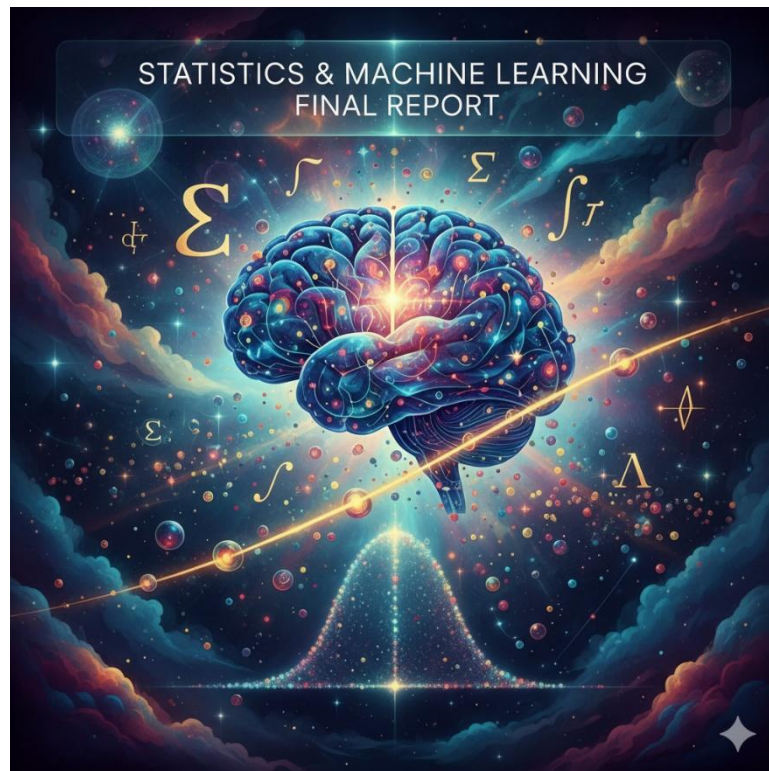


Data Analytics – 1

Statistics and Machine Learning



Online News Popularity Prediction Report

Team Members:

Muhammed Razi Mullappalli (100004340)

Megharaj Shiva Kumar (100004501)

Komal Gubbi Deepak (100004243)

Kavana Gopladevarahalli Papegowda (100004573)

Introduction

About The Dataset:

Source: The dataset was collected from the UCI Machine Learning Repository and contains articles published by Mashable.

DOI (Digital Object Identifier): 10.24432/C5NS3V

Size: The dataset consists of 39,644 samples (rows), each representing a unique news article. There are 61 attributes (columns) for each article.

Feature Categories:

The dataset includes 58 predictive attributes which can be grouped as:

1. Basic Content Features

- `n_tokens_title`: Number of words in the title
- `n_tokens_content`: Number of words in the content
- `num_imgs`: Number of images in the article
- `num_videos`: Number of videos in the article

2. Metadata Features

- Day of the week indicators (`weekday_is_monday`, ... `weekday_is_sunday`)
- Data channel indicators (e.g., `data_channel_is_lifestyle`, `data_channel_is_entertainment`)

3. Natural Language / Keyword Features

- `kw_min_min`, `kw_max_max`, `kw_avg_avg`: statistics about keywords used in the article
- `self_reference_min_shares`, `self_reference_avg_shares`: stats about shares of related articles

4. Time Features

- `timedelta`: Days between the article's publication and dataset collection

5. Target Variable:

- `shares` → the number of times an article was shared on social networks
- Often transformed into a binary class (popular vs not popular, for classification).

Problem Statement

News editors at publishers like Mashable must efficiently allocate promotional resources across hundreds of daily articles to drive traffic and revenue. Identifying potentially viral content is crucial, but it also provides a valuable case study; understanding what makes an article popular helps editors refine future content strategy. This project builds a machine learning system to predict an article's popularity based on its features. The system's final goal is to automatically rank and identify the top articles most likely to succeed, providing editors with both immediate promotional targets and long-term strategic insights.

Analysis of Missing Data Distribution and Imputation Strategy

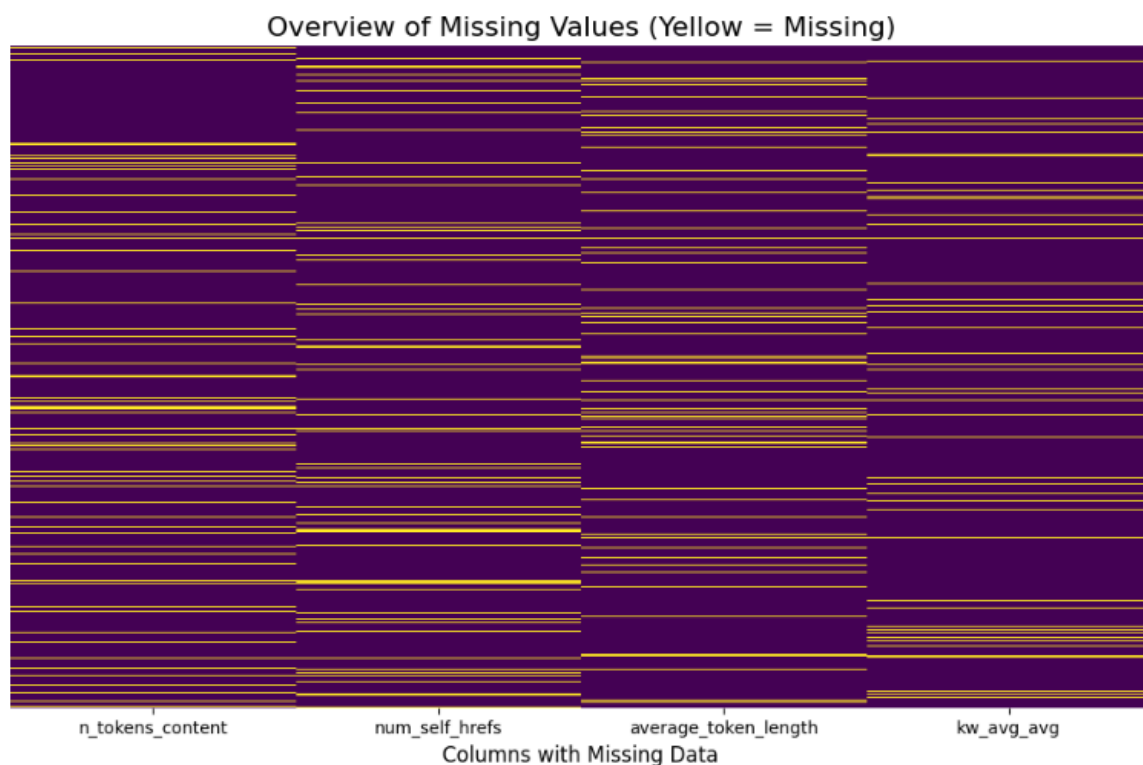
Data Inspection Methodology

This analysis was conducted to inspect all columns with missing values and determine the best statistical method for filling those gaps.

Analytical Methodology:

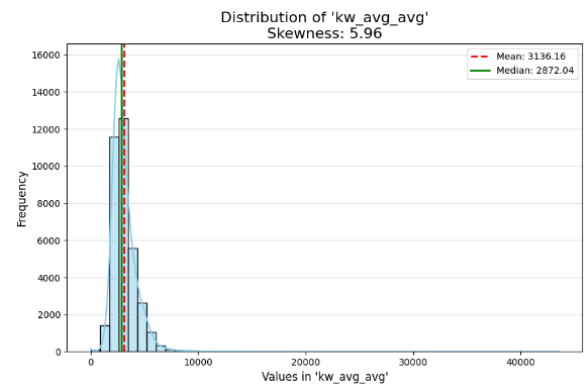
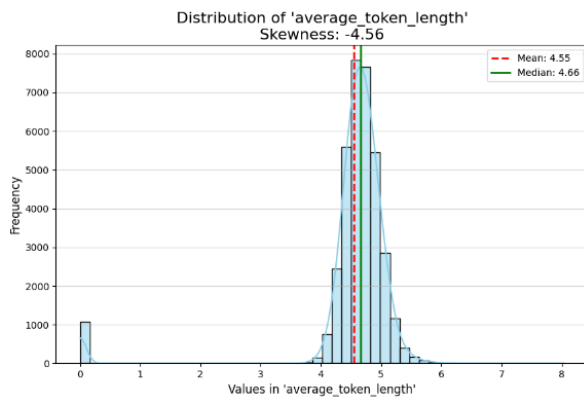
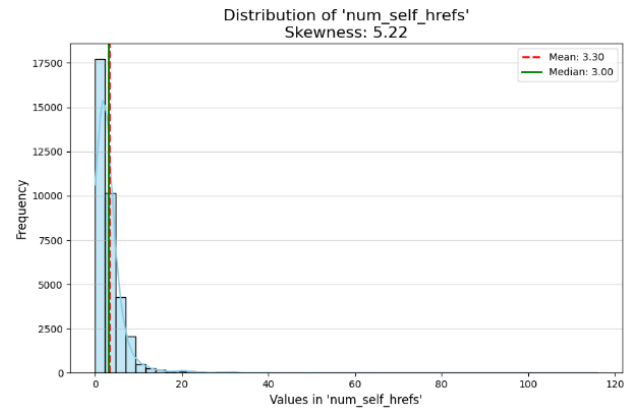
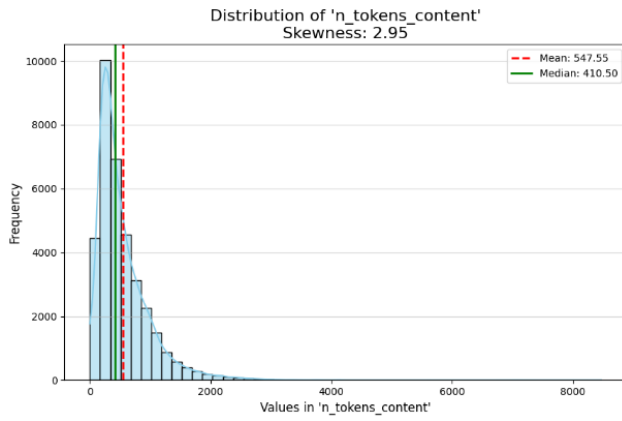
1. Identification and Visualization: The process began by identifying four specific columns with missing data. A heatmap was generated, visually confirming the location of these gaps across the dataset.

2. Distribution and Skewness Check: For each column, a distribution plot was created to check its shape. The skewness value was calculated, and the Mean and Median were plotted on the graph to assess how outliers were affecting the central tendency.
3. Imputation Rule: The strategy used was to recommend the Median for highly skewed data (absolute skewness ≥ 1) and the Mean for symmetrical data.



Distribution Shape and Imputation Choice

The analysis confirmed that all four columns containing missing values (`n_tokens_content`, `num_self_hrefs`, `average_token_length`, and `kw_avg_avg`) exhibited high degrees of skewness (absolute skewness ranging from $|2.95|$ to $|5.96|$).



Imputation Method

We used an imputation strategy that calculated skewness for all four columns. The analysis confirmed that all four columns were highly skewed, dictating the use of the Median for a robust imputation. The execution successfully cleaned the data as verified by the final visualization.

Execution Details:

- `n_tokens_content`: Skewness 2.95→ Imputed with Median (410.50).
- `num_self_hrefs`: Skewness 5.22→ Imputed with Median (3.00).
- `average_token_length`: Skewness -4.56→ Imputed with Median (4.66).
- `kw_avg_avg`: Skewness 5.96→ Imputed with Median (2872.04).



Encoding Method

This process streamlines the dataset by consolidating thirteen one-hot encoded (binary) columns into two singles, numerically labeled features to reduce dimensionality and improve model efficiency.

The two main transformations are:

1. **Weekday Consolidation:** Seven weekday columns (weekday_is_monday to weekday_is_sunday) are consolidated into a single day_of_week integer column (0 to 6). Labels 0 (Monday) through 6 (Sunday) were assigned, and the original seven binary columns were dropped.
2. **Channel Consolidation:** Six data channel columns are consolidated into a single data_channel integer column (0 to 5). Labels 0 ('lifestyle') through 5 ('world') were assigned based on the active channel, and the original six binary columns were dropped.

We used this transformation strategy to convert the one-hot encoded features to label-encoded features. This preserves all categorical information while replacing 13 binary columns with 2 compact integer columns.

Key Findings:

- **Missing Data Cleaned:** All four highly skewed columns containing missing values were successfully filled using the Median imputation strategy, which is robust against the influence of outliers.
- **Dimensionality Reduced:** The feature count of the dataset was reduced by eleven by consolidating thirteen one-hot encoded columns (weekdays and channels) into two compact, label-encoded features.
- **Data Preparation Complete:** The dataset is now fully cleaned of missing values and streamlined for efficient model training, having preserved all critical categorical and numerical data points.

Feature Engineering

Feature Engineering is the process of creating new variables or transforming existing ones to improve the performance of machine learning models. It helps the model capture hidden patterns in the data more effectively.

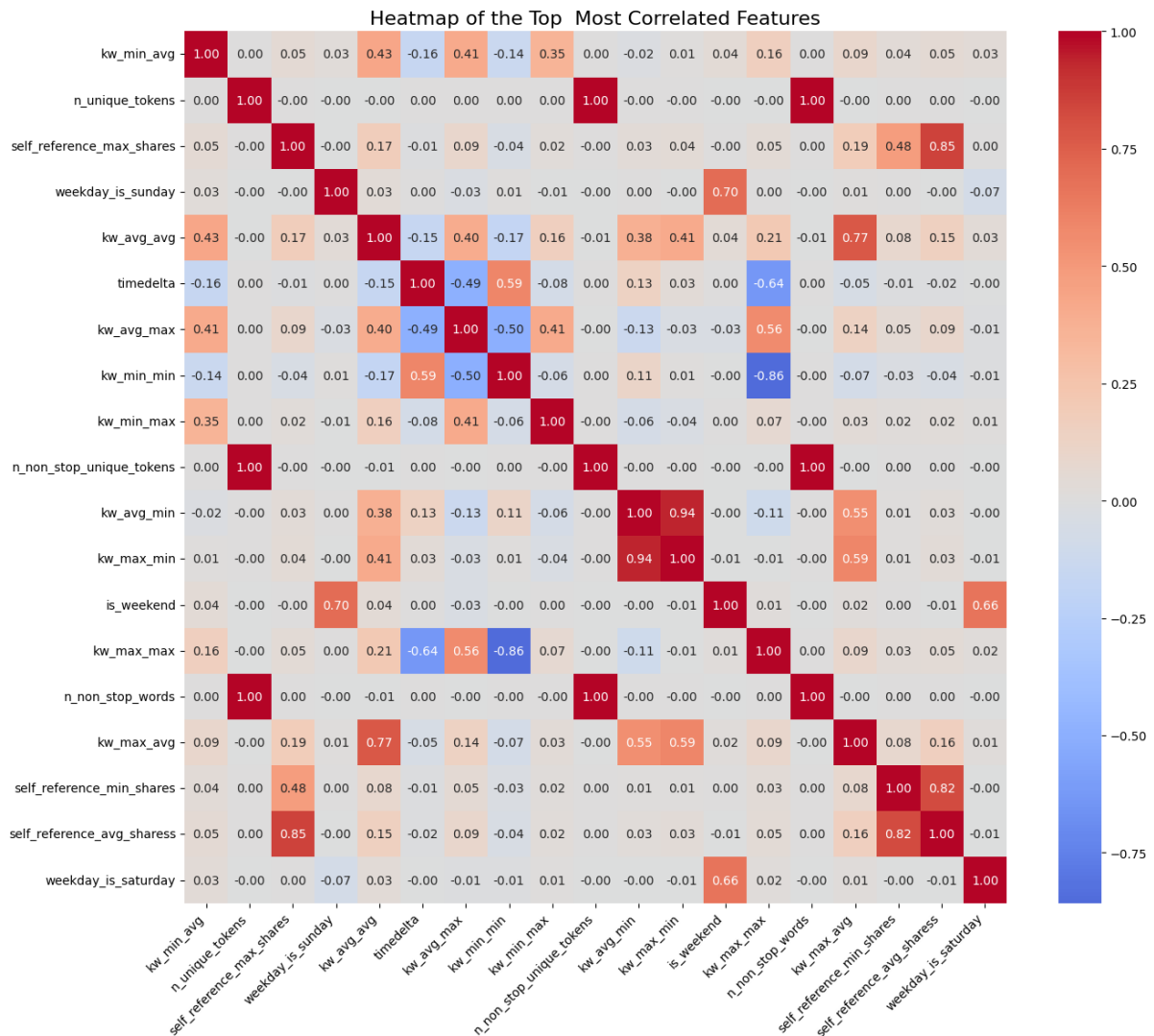
In the Online News Popularity project, feature engineering was applied in several ways:

Merging Features:

- Combined 7 weekday columns into a single `day_of_week` feature.
- Combined 6 data channel columns into one `data_channel` feature.
- This reduced redundancy, lowered dimensionality, and simplified analysis.

Creating New Features:

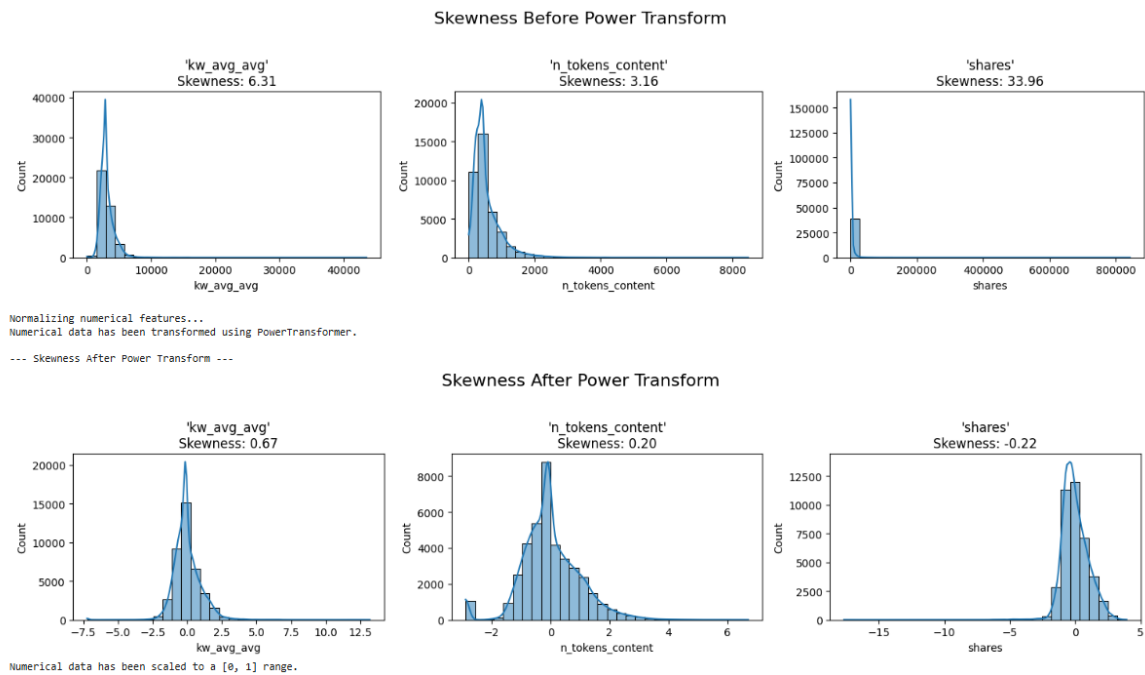
- Introduced `keyword_power_score = kw_avg_avg * kw_max_avg`, combining average keyword popularity with peak keyword popularity to better capture the influence of keywords.



Power Transformation and Normalization

Our data was highly skewed to reduce skewness, applied power transformations (Yeo-Johnson) to make skewed distributions more symmetric and stable for modelling.

The Yeo-Johnson transformation is a statistical technique used to make skewed data more normally distributed. Many machine learning models work best when features follow a distribution closer to normal (bell-shaped), so this transformation helps improve model stability and performance.



Normalization:

- Scaling numerical features to a consistent range (usually [0, 1]).
- Ensures all features are weighted equally in the model.
- Prevents features with large values from dominating.

Min-Max Scaling:

- We applied Min-Max Scaling to scale all numerical features between 0 and 1.
- This guarantees all our features are weighted equally by the model.

Noise Injection & Cleaning

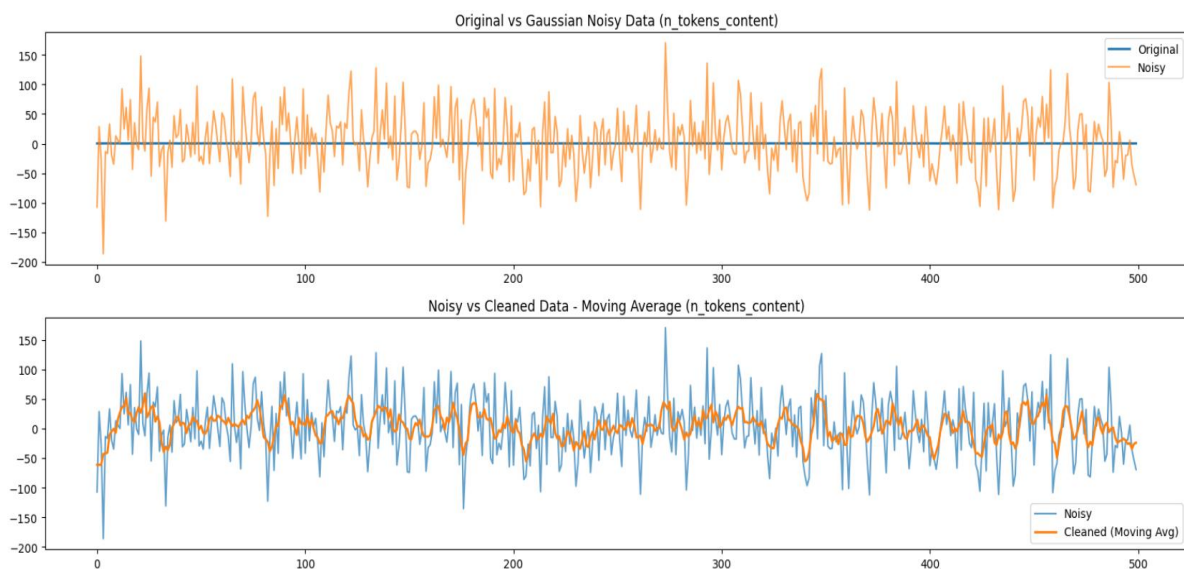
- Gaussian noise was added to simulate real-world imperfections in the “n_tokens-content” feature.
- The noisy data showed increased fluctuations and sharper peaks, deviating from the original signal.
- Applied smoothing techniques to reduce noise and restore signal clarity.

Visual comparison:

- Original vs Noisy: Clear distortion introduced by Gaussian noise.
- Noisy vs Cleaned: Smoothed data closely follows the original pattern.

Mean Squared Error (MSE) analysis:

- MSE between original and noisy: 2492.86
- MSE between original and cleaned: 504.33



After normalizing the feature, we injected Gaussian noise and applied Moving Average smoothing technique. The Mean Squared Error dropped from 2492.86 (noisy) to 504.33 for the cleaned versions—demonstrating the effectiveness of our noise reduction pipeline.

Unsupervised Learning (K-Means Clustering, k=2)

Clustering Process and Data Preparation

The K-Means clustering analysis aimed to segment news articles based purely on their intrinsic feature patterns. The methodology involved several critical pre-processing steps. First, non-numerical and target-related features (like a 'share' count) were excluded to maintain an unsupervised focus. The remaining quantitative data was then standardized to ensure all features had equal influence in defining the clusters. Finally, the K-Means algorithm was executed to assign each article to one of the two designated groups (k=2). For visualization, Principal Component Analysis (PCA) was applied to reduce the data dimensionality to two core components (PC1 and PC2).

Cluster Metrics and Distribution

This section examines the performance of the k=2 model using the Silhouette Score and analyzes the resulting data point distribution.

Silhouette Score Analysis

```
The Silhouette Score for k=2 is: 0.5054

Cluster counts:
cluster
0      38463
1       1181
Name: count, dtype: int64
```

Silhouette Score = 0.5054

Interpretation: This score suggests a reasonably good separation between the two clusters, as the value is significantly above 0 and slightly over 0.5. This indicates that articles are generally well-assigned, being closer to their own cluster center than to the other cluster's center, though some overlap may exist near the boundaries.

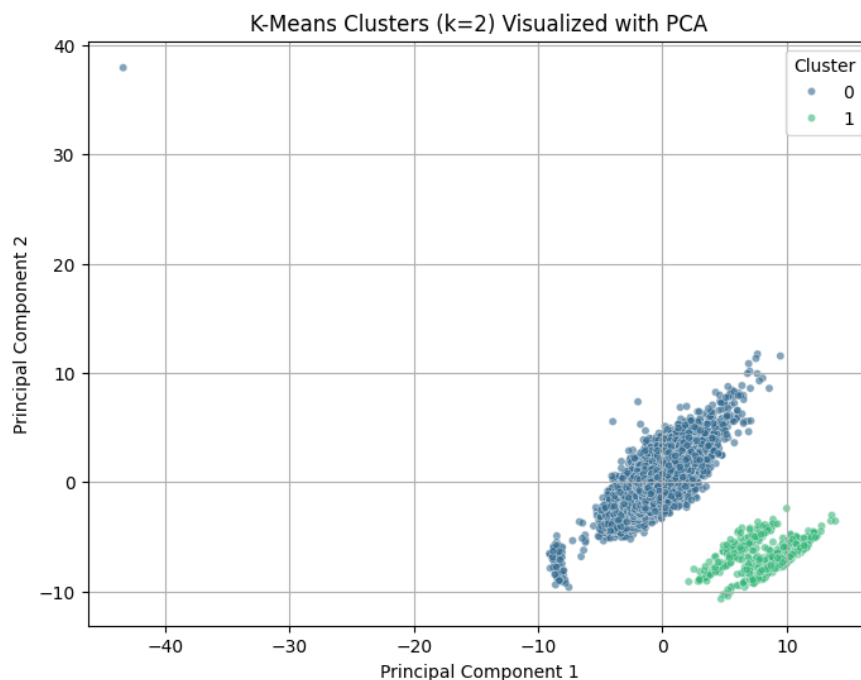
Cluster Distribution

The clustering resulted in an extremely imbalanced distribution:

Observation: This distribution shows that Cluster 1 isolates a very small, distinct minority group from the overwhelming majority in Cluster 0. Cluster 1 likely represents an anomalous or highly specialized segment of the data.

PCA Scattering Plot Interpretation

The K-Means clusters, visualized using the two principal components (PC1 and PC2) to maximize data variance, clearly show the geometric separation achieved by the model.



Visual Interpretation: Cluster 0 (Blue/Teal) forms a dense mass near the origin, representing the primary data bulk and confirming the uniformity of the majority segment. Cluster 1 (Green) is distinctly separated, located mainly in the positive PC1 and negative PC2 space, visually confirming that this minority group possesses fundamentally different feature characteristics from the majority. This separation aligns with the calculated 0.5054 Silhouette Score. Note that a few

distant, isolated blue points within the plot may represent potential outliers that were nevertheless absorbed into the larger Cluster 0 group.

Data Preparation and Imbalance Identification

This analysis focused on preparing the dataset for a classification task by addressing severe class imbalance.

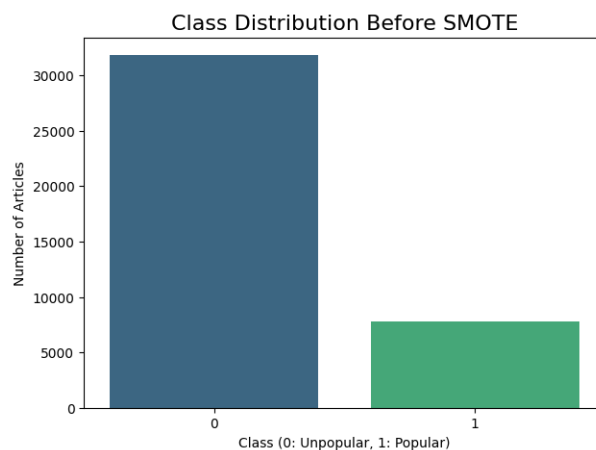
Analytical Methodology:

1. **Classification Target Definition:** The initial step involved converting the original 'shares' metric into a binary classification problem. A threshold was established to define two classes: "Unpopular" and "Popular".
2. **Imbalance Visualization:** The feature columns were separated from the new classification target. A visual check of the class distribution clearly demonstrated a significant imbalance, with the Unpopular class heavily outweighing the Popular class.
3. **Application of SMOTE:** To equalize the representation of both classes, the Synthetic Minority Over-sampling Technique was applied. This method generates synthetic data points for the minority class along the lines connecting existing minority instances, effectively increasing its count without simply duplicating existing entries.
4. **Final Output:** The resulting balanced dataset, containing the original majority instances and the newly synthesized minority instances, was saved for subsequent model training.

Evaluation of Balancing Technique, Distribution, and Graph Analysis

This section combines the evaluation of the SMOTE technique's effectiveness with an analysis of the visual confirmation provided by the bar graphs.

Class Distribution Before SMOTE (Imbalanced): The initial distribution was highly skewed. The graph visually confirms this necessity, showing a tall bar for Class 0 and a significantly shorter bar for Class 1, highlighting that Class 0 had approximately 4 times the number of articles as Class 1. This severe imbalance makes it difficult for traditional models to learn the characteristics of the minority class accurately.



Class Distribution After SMOTE (Balanced): After applying the over-sampling technique, the class counts were successfully equalized, and the objective was visually met. The second graph displays two bars of nearly identical height. Both Class 0 and Class 1 now contain approximately 31,000 data points each, achieving a nearly 50/50 split. This balanced dataset ensures that the model does not simply bias its predictions towards the more frequently observed "Unpopular" class, and the equalization removes the structural bias inherent in the original data distribution.



Supervised Learning

The Random Forest Classifier is an ensemble learning algorithm used for classification tasks. It works by combining the predictions of multiple decision trees to make a final decision.

How it works:

- Each decision tree is trained on a random sample of the data (bootstrapping).
- At each split in the tree, a random subset of features is considered.
- Each tree gives a “vote” for the class (e.g., popular or not popular).
- The class with the majority of votes becomes the model’s final prediction.

Why it is effective:

- Reduces overfitting compared to a single decision tree.
- Handles high-dimensional data and complex feature interactions.
- Robust to noise and works well even with missing or unscaled data.
- Provides feature importance scores, showing which variables are most useful for prediction.

In this project:

- Random Forest was chosen because the dataset is large, has many numerical features, and was initially imbalanced.
- After balancing with SMOTE, the Random Forest Classifier achieved 86% accuracy, precision, recall, and F1-score, showing consistent and reliable performance.

--- Model Evaluation ---

Accuracy: 0.8600

Precision: 0.8613

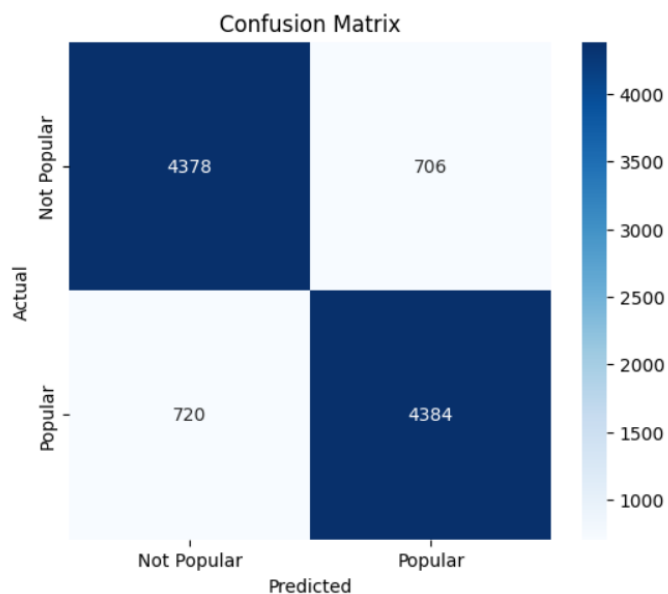
Recall: 0.8589

F1 Score: 0.8601

Classification Report:

	precision	recall	f1-score	support
0	0.86	0.86	0.86	5084
1	0.86	0.86	0.86	5104
accuracy			0.86	10188
macro avg	0.86	0.86	0.86	10188
weighted avg	0.86	0.86	0.86	10188

- Accuracy - Out of every 100 articles, the model correctly predicts about 86 of them.
- Precision - When the model says an article will be popular, it's right about 86% of the time.
- Recall - Out of all the articles that were actually popular, the model correctly found 86% of them.
- F1 Score - A balance between Precision and Recall, basically confirms the model is consistent and not favouring one side.

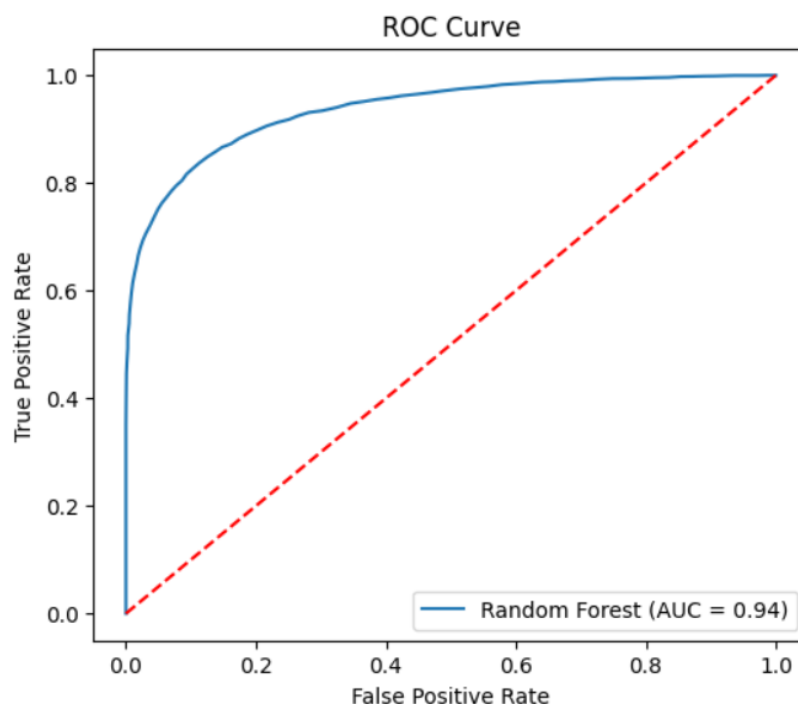


Confusion Matrix

A confusion matrix is a table that shows how well a classification model performs by comparing actual values with predicted values.

In the matrix:

- Top-left (4378) → True Negatives (TN) - Articles that were Not Popular and the model correctly predicted them as Not Popular.
- Top-right (706) → False Positives (FP) - Articles that were actually Not Popular, but the model incorrectly predicted them as Popular.
- Bottom-left (720) → False Negatives (FN) - Articles that were actually Popular, but the model incorrectly predicted them as Not Popular.
- Bottom-right (4384) → True Positives (TP) - Articles that were Popular and the model correctly predicted them as Popular.



What the Plot Shows:

- Blue curve: Performance of your Random Forest model.
- Red dashed line: A random guess baseline (AUC = 0.5).

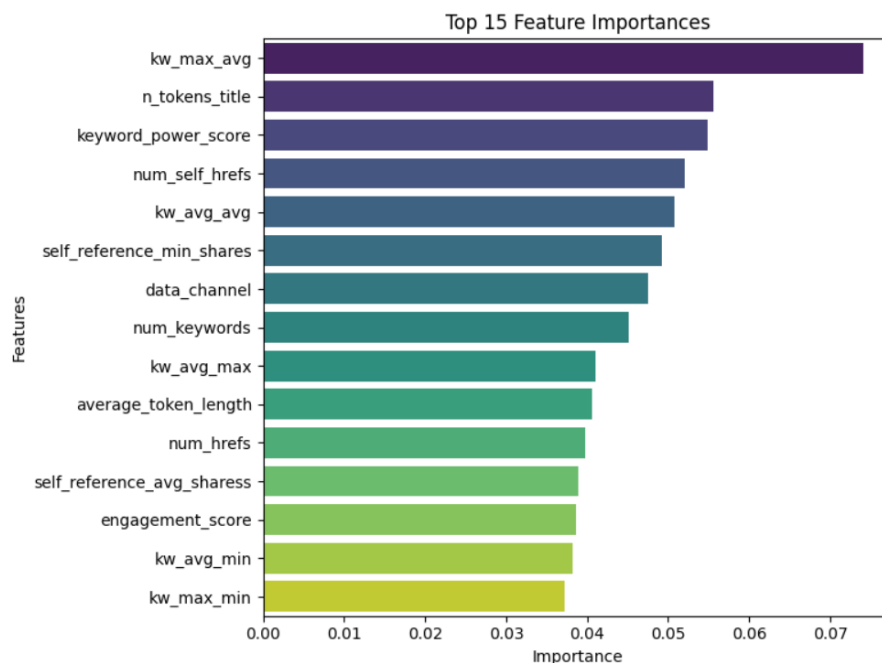
- $AUC = 0.94$: The area under the blue curve.

How to Interpret:

- The curve hugs the top-left corner, meaning the model achieves high True Positive Rate (Recall) while keeping the False Positive Rate low.
- $AUC = 0.94$ is excellent \rightarrow the model is very good at distinguishing Popular vs. Not Popular articles.
- Compared to random guessing (red line), the Random Forest performs much better.

Summary:

This ROC curve shows that your Random Forest classifier is highly effective ($AUC = 0.94$). It can correctly classify whether an article is popular or not with very high confidence, and it's much better than chance.



What the Plot Shows:

- Each bar represents a feature (a column from your dataset).

- The length of the bar shows how much that feature contributes to predicting whether an article is Popular or Not Popular.
- The top 15 most important features are displayed here.

Key Insights

- `kw_max_avg` (maximum average keyword weight) is the most important feature. Articles with strong, high-weighted keywords are much more likely to be popular.
- `n_tokens_title` (number of words in the title) also matters a lot. Titles with the right length/word count strongly influence popularity.
- `keyword_power_score`, `num_self_hrefs`, `kw_avg_avg` These relate to how keywords and links are used in the article. Good keyword usage and internal linking help increase article visibility → more popularity.
- `data_channel` (topic category, like lifestyle, tech, entertainment) Certain categories naturally attract more clicks and shares.
- Other features like `average_token_length`, `engagement_score`, and `num_keywords` These give additional signals about how the article is structured and how engaging it might be.

Summary:

This plot tells us what drives article popularity the most.

- Keywords and title length are the strongest predictors.
- Engagement signals (shares, links, self-references) also matter.
- The Random Forest model automatically ranks these features based on how useful they were in splitting the data into Popular vs Not Popular.

Conclusion

- The Random Forest model achieved strong, balanced performance (86% across metrics).
- Preprocessing (feature engineering, power transformation, noise cleaning) was critical.
- Unsupervised learning provided complementary insights into article clusters.
- Practical value: Editors can use the model to identify which articles are likely to go viral, saving promotional effort and increasing traffic.

References

- Fernandes, K., Vinagre, P., Cortez, P., & Sernadela, P. (2015). Online News Popularity. UCI Machine Learning Repository. DOI: 10.24432/C5NS3V.