Analysis on New York Times Data (2020)

Course Code: CS 328

Submitted By:

23110021	Akash K P
23110189	Luv Agarwal
23110261	Rachit Mehta
23110269	Vivek Rajput

NYT Article and Comments Data Analysis

Abstract

This project explores user interaction and engagement on the New York Times online platform using article and comment data from the year 2020. We address multiple tasks, including predicting comment volume and engagement, identifying editorial selections, and uncovering thematic trends in news headlines. By applying machine learning and natural language processing techniques, the project provides actionable insights into reader behavior and editorial influence. The results can support content recommendation systems and editorial decision-making processes.

Introduction

In the digital age, user engagement has become a critical metric for online content platforms. The New York Times, one of the most prominent news outlets, offers readers the ability to leave comments and react to published content. This project uses publicly available data from 2020 to analyze and model this interaction. We investigate four primary tasks:

- Predicting the number of comments an article receives,
- Forecasting the number of recommendations a comment gets,
- Generating news headlines using a LSTM neural network,
- Classifying whether a comment is selected as a Times Pick,

These tasks provide a multifaceted view of content impact and reader preferences.

Preliminaries

The dataset consists of two files:

- Articles Dataset: Contains over 16,000 articles with metadata such as headline, section, and n_comments.
- Comments Dataset: Contains nearly 5 million comments, including text, timestamps, recommendations, and whether it was a Times Pick (editorsSelection).

Libraries used include:

- pandas, numpy for data processing
- sklearn for machine learning pipelines
- matplotlib, seaborn for visualization
- nltk, scikit-learn, and gensim for NLP tasks

Related Work

Previous studies, such as Aashita Kesarwani's 2018 NYT dataset analysis and the MITx "Analytics Edge" competition, have examined similar themes of reader engagement. Other academic research has focused on predicting virality in online content and topic modeling using LDA. These works have inspired our approach in classification, regression, and unsupervised clustering of news content.

Approach

Task 1: Predict Number of Comments on NY Times Articles

Introduction

Understanding reader engagement has become vital for content platforms like the New York Times. In this task, we aim to predict the number of comments an article will receive based on its metadata and textual features. The insights gained from such predictions can help optimize article strategies and user interactions.

Baseline Models Without Feature Engineering

Initially, basic models were trained using only raw features: "section" and "word_count", without any transformations or enhancements.

Model	R^2 Score
Linear Regression	0.33
Neural Network (MLPRegressor)	0.36

Without meaningful feature construction, the models performed poorly, as they could not capture complex patterns present in the data.

Feature Engineering

Feature engineering introduced richer relationships between variables. New features like the word counts of the headline and abstract, as well as an interaction term (headline word count $\times$ abstract word count), were created.

Model	R^2 Score
Linear Regression	0.3361
Neural Network	0.3794
Random Forest	0.2800
XGBoost	0.3710

The models showed improvement, indicating that engineered features captured non-linear aspects of article metadata.

Target Transformation (Logarithmic)

The distribution of the number of comments was highly skewed, dominated by articles with low engagement but a few with very high counts. A log transformation $y = \log(1 + \text{number of comments})$ was applied to stabilize variance and normalize the data.

Model	R^2 Score
Linear Regression	0.4183
Neural Network	0.4712
Random Forest	0.4277
XGBoost	0.4713

Post-transformation, all models exhibited significant performance gains, validating the necessity of this preprocessing step.

TF-IDF Vectorization

TF-IDF (Term Frequency-Inverse Document Frequency) vectorization was applied to headline and abstract texts. TF-IDF highlights words that are frequent in a document but rare across the corpus, providing better feature representation than simple word counts.

Model	R^2 Score
Linear Regression	0.4148
Neural Network	0.4722
Random Forest	0.4237
XGBoost	0.4658

The incorporation of TF-IDF features provided marginal improvements, particularly for neural networks that benefited from high-dimensional text features.

Model Architecture and Motivation

a) Why Build Fusion Architectures?

While BERT embeddings provided powerful text representations, combining them with structured metadata was crucial for improving performance. Fusion architectures were developed to jointly learn from both numerical and semantic inputs, addressing the shortcomings of using either type alone.

b) Fusion Network Designs

- **Fusion Layers + Dropout**: Dense neural layers combined BERT embeddings with numerical features, interspersed with Dropout layers to prevent overfitting. In this architecture, we utilized All Mini LM L6 v2, a more efficient variant of BERT, as the embedding layer. This model generates compact, high-quality text representations, making it well-suited for environments with computational constraints. The combination of these embeddings with numerical features allows the model to leverage both textual and structured data effectively.

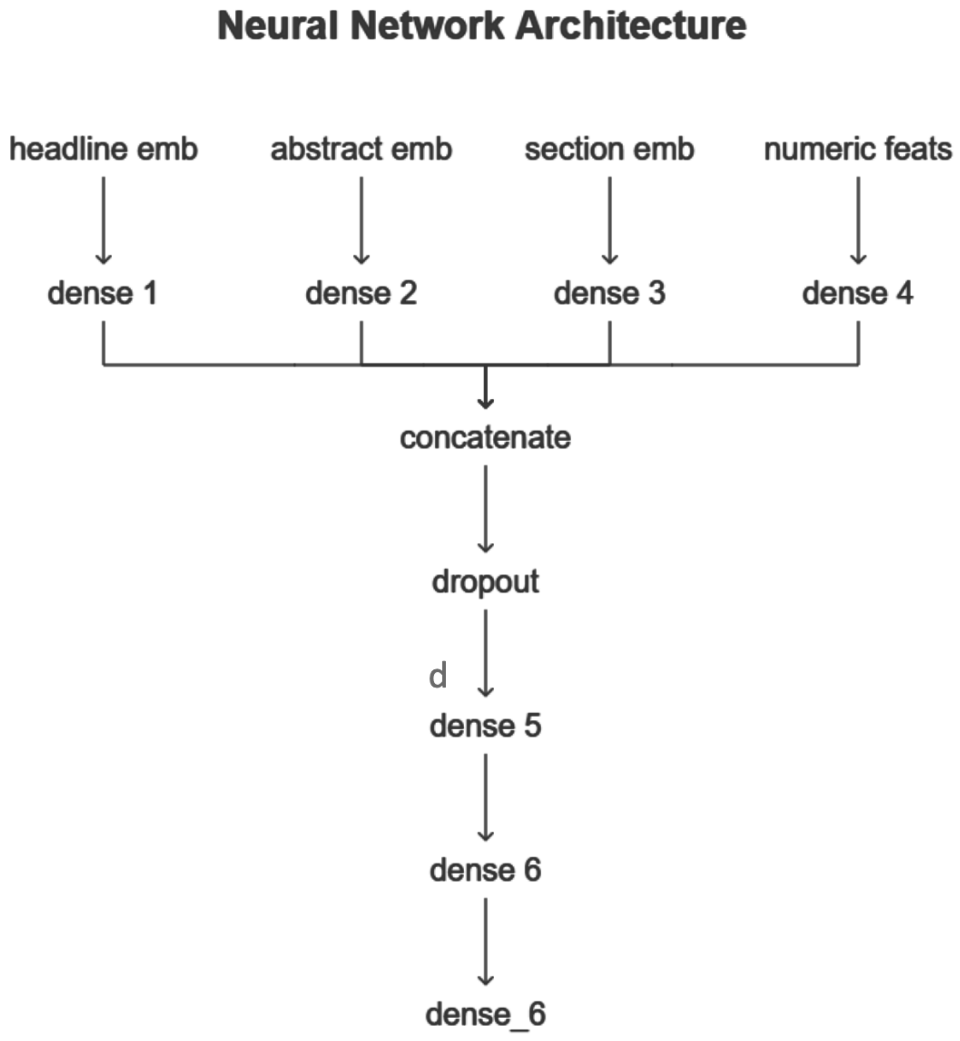


Figure 1: Architecture 1 using fusion layers for embedding with All Mini LM L6 v2

- **Fusion Layers + Batch Normalization**: Batch normalization was added after dense layers to stabilize and accelerate learning, leading to modest improvements.

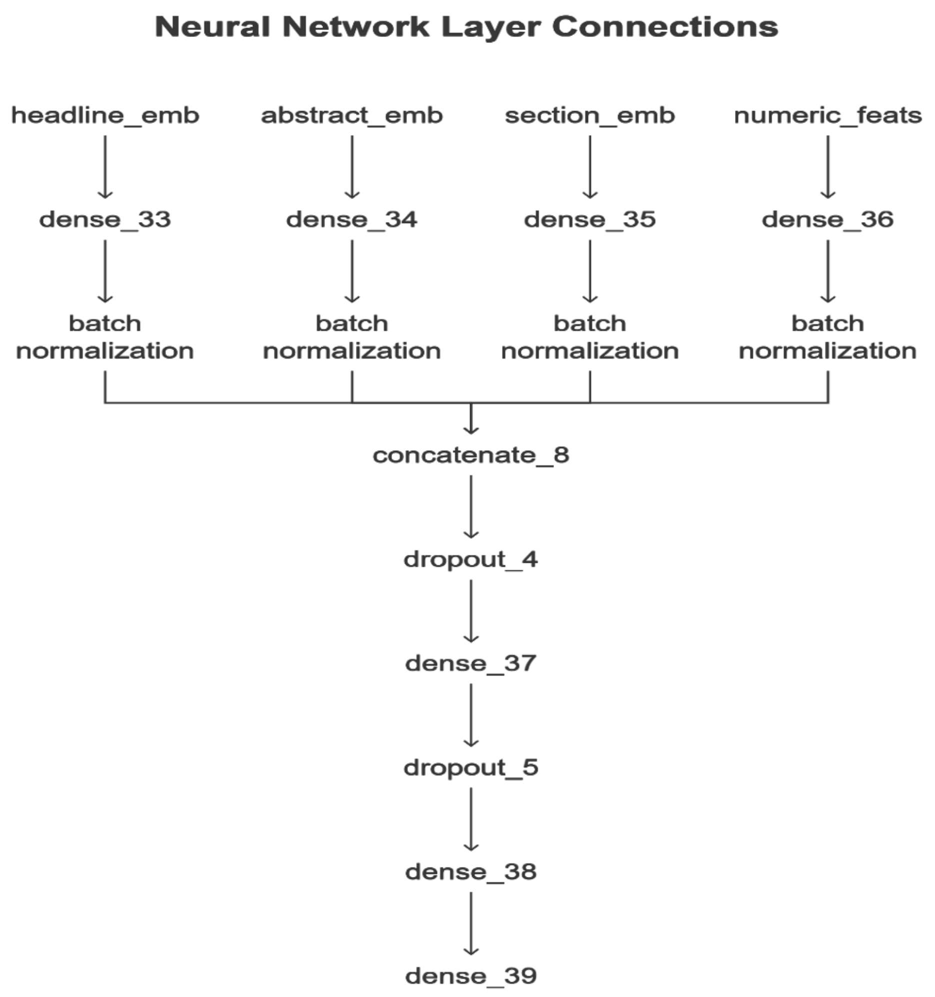


Figure 2: Architecture 2 using Batch Normalisation

- **Fusion Layers + Residual Learning (Skip Connections)**: Inspired by ResNet, skip connections allowed gradients and information to flow more easily through deeper layers, resulting in the best performance.

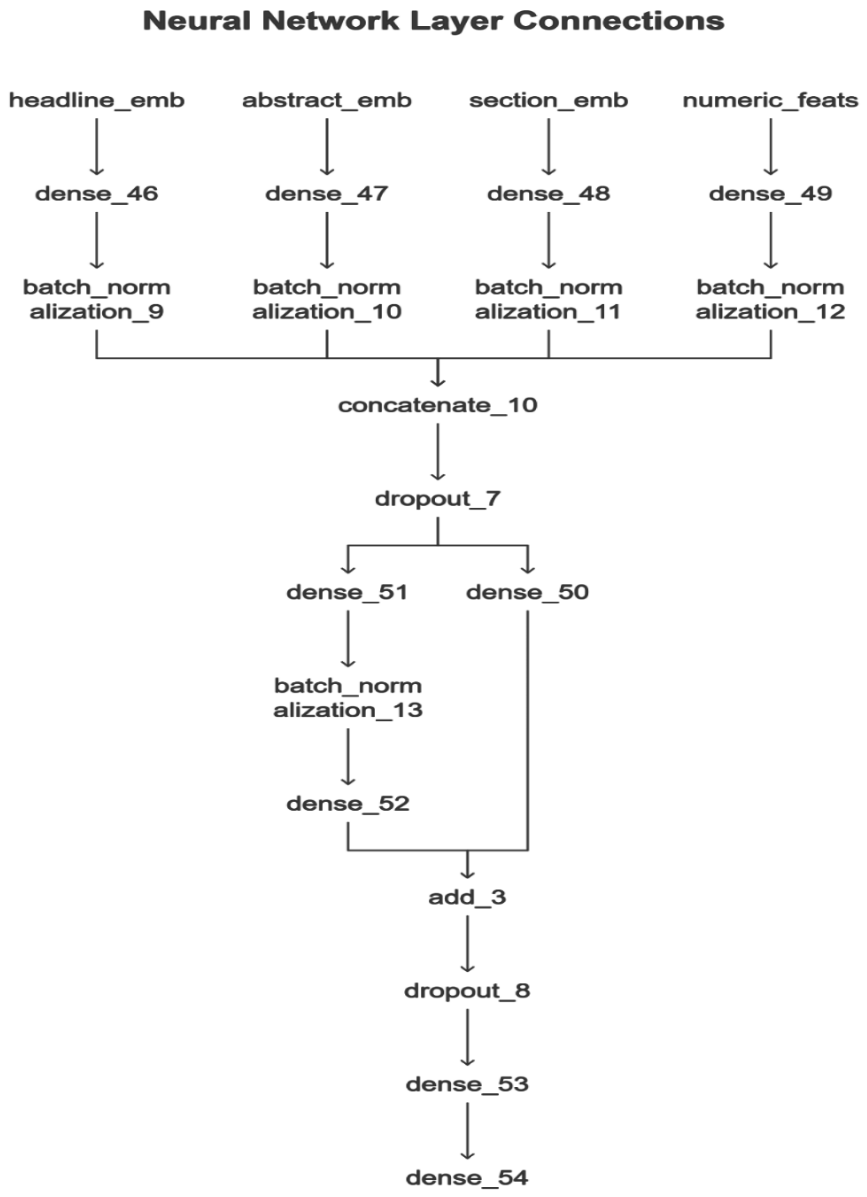


Figure 3: Architecture 3 using Residual Learning

c) Summary of Benefits

Fusion architectures effectively captured both high-level semantic information and low-level structured patterns, producing superior results.

Deep Learning Architectures: Fusion Models

Architecture	R^2 Score
Fusion Layers + Dropout	0.49
Fusion Layers + Batch Normalization	0.50
Fusion Layers + Residual Learning	0.5325

Residual learning clearly provided the highest accuracy, demonstrating the value of advanced deep learning techniques in multi-modal fusion problems.

BERT Embeddings (Without Feature Engineering)

- Used pre-trained BERT embeddings to convert text (e.g., headlines) into dense vectors.
- Captured contextual meaning of words based on their position and usage in a sentence.
- Provided richer, more accurate text representation compared to traditional methods.
- Used as input for deep learning models, boosting prediction accuracy.

Model	R^2 Score
Linear Regression	-3.01×10^{18}
Neural Network	-0.0637
Random Forest	0.5417
XGBoost	0.5474

BERT with Feature Engineering

- Extracted BERT embeddings from textual data like headlines and/or abstracts.
- Engineered a custom interaction term: word count $\times$ headline word count.
- Combined BERT vectors with structured features (e.g., interaction term, word counts) into a unified input.
- Enabled models to learn from both semantic text context and quantitative patterns.
- Resulted in improved model performance by leveraging both deep language understanding and structured signals.

Model	R^2 Score
Linear Regression	-1.04×10^{18}
Neural Network	-0.046
Random Forest	0.5419
XGBoost	0.5499

Tuned BERT Embeddings

- Used raw BERT / DistilBERT models (AutoModel from Hugging Face).
- Extracted outputs from the last hidden states (sequence of token embeddings).
- Optionally included the [CLS] token representation.
- Provided raw contextualized word representations (token-level embeddings for each input token).

Flexibility and Control:

- Fine-tuned the transformer model on the specific task.
- Explored multiple strategies to extract sentence-level representations:
  - **Mean Pooling**: Averaged across token embeddings.
  - **[CLS] Token**: Used the embedding of the [CLS] token.
  - **Attention Pooling**: Applied learnable attention weights to pool token representations.
- Allowed more control over how final features were formed.

Model Performance:

Method	R^2 Score
Tuned BERT Embeddings	0.5589

Neural Network Architecture:

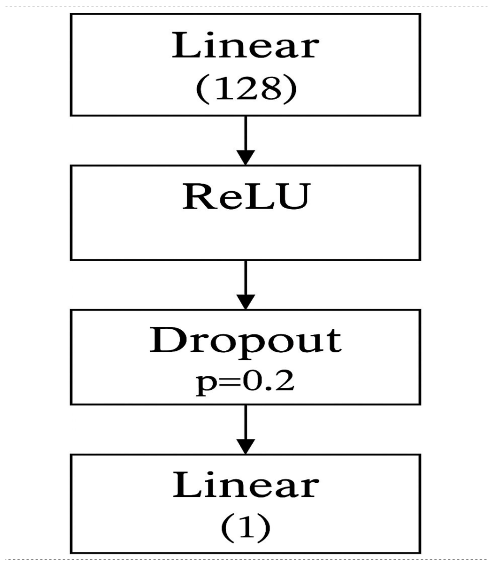


Figure 4: Architecture using Tuned BERT Embeddings

```
nn.Linear(self.bert.config.hidden_size, 128),
nn.ReLU(),
nn.Dropout(0.2),
nn.Linear(128, 1)
```

Task 2: Predict Recommendations on Comments

The main goal of this task was to identify hidden themes within NY Times headlines from 2020 by applying two well-known unsupervised machine learning techniques: KMeans Clustering and Latent Dirichlet Allocation (LDA). Both methods aim to group similar headlines together and extract the main themes from each group. To begin with, the headlines were cleaned by removing unnecessary words (such as stopwords) and symbols, and converting all text to lowercase to maintain uniformity.

K-Means Clustering:

The Elbow Method was applied to determine the optimal number of clusters. This involved plotting the inertia (the sum of squared distances between data points and their respective cluster centers) against varying values of K, the number of clusters. It was observed that inertia decreased as the number of clusters increased, but the rate of decrease slowed after a certain point. The elbow was identified at K = 5, indicating that five clusters provided an optimal balance between model complexity and clustering accuracy.

K-Means Clustering Results:

Cluster 0: Focused on Trump, Republicans, and elections.
`['trump', 'donald', 'say', 'republican', 'election', 'administration', 'virus', 'campaign', 'president', 'court']`

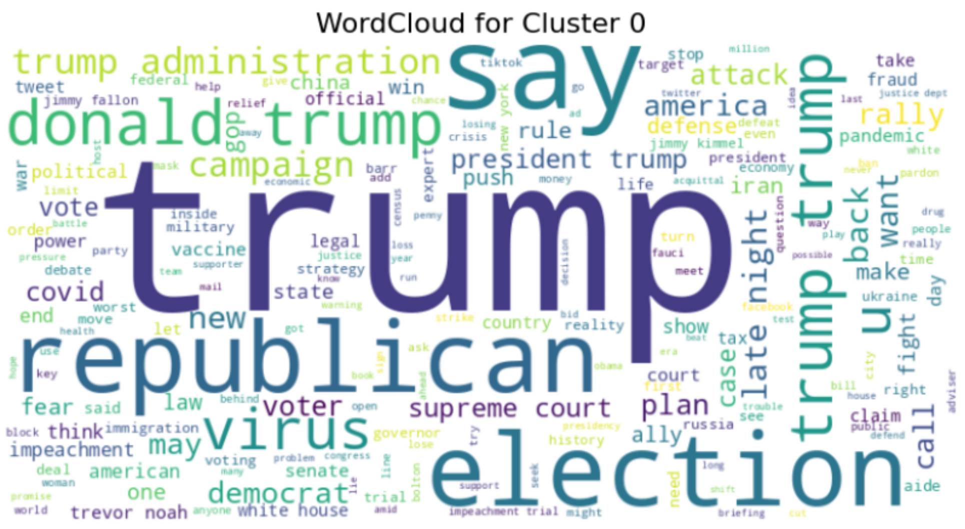


Figure 5: Cluster 0

Cluster 1: Related to home sales, housing, and real estate.
`['home', 'california', 'million', 'sale', 'new', 'york', 'manhattan', 'brooklyn', 'nursing', 'sold']`

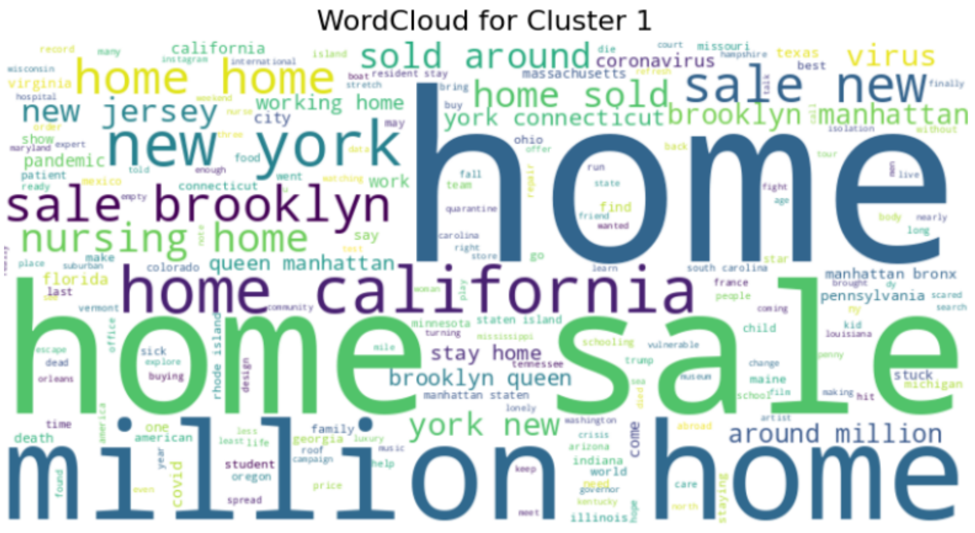


Figure 6: Cluster 1

Cluster 2: Dominated by coronavirus-related discussions.
['coronavirus', 'vaccine', 'test', 'case', 'latest', 'china', 'new', 'update', 'nyc', 'outbreak']

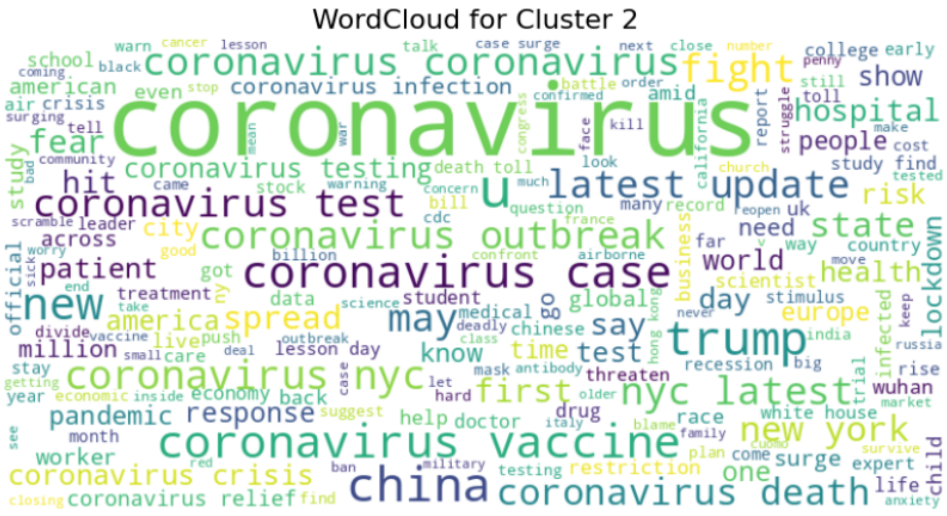


Figure 7: Cluster 2

Cluster 3: Centered around the pandemic and new virus.
`['new', 'pandemic', 'virus', 'time', 'day', 'life', 'covid', 'year', 'york', 'one']`

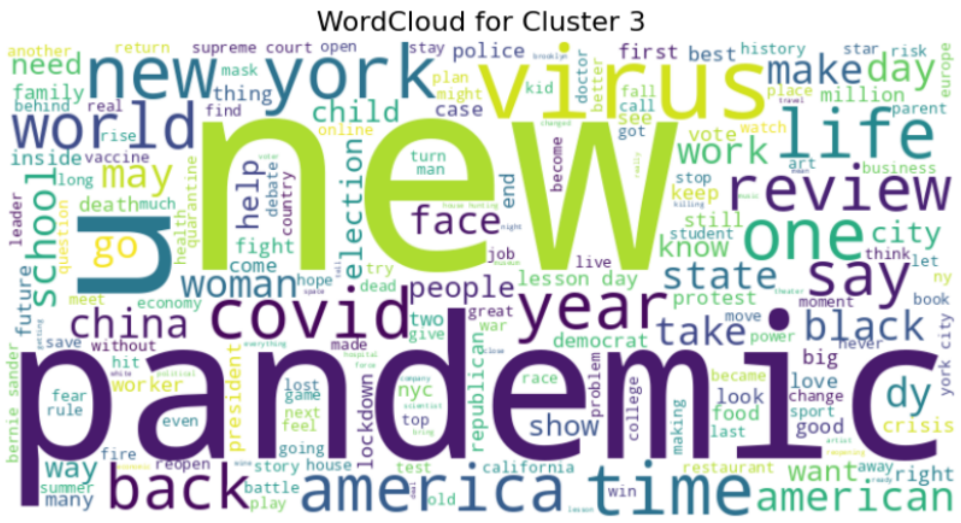


Figure 8: Cluster 3

Cluster 4: Focused on Biden, Trump, and the 2020 election.
`['biden', 'joe', 'trump', 'win', 'lead', 'voter', 'sander', 'pick', 'debate', 'could']`

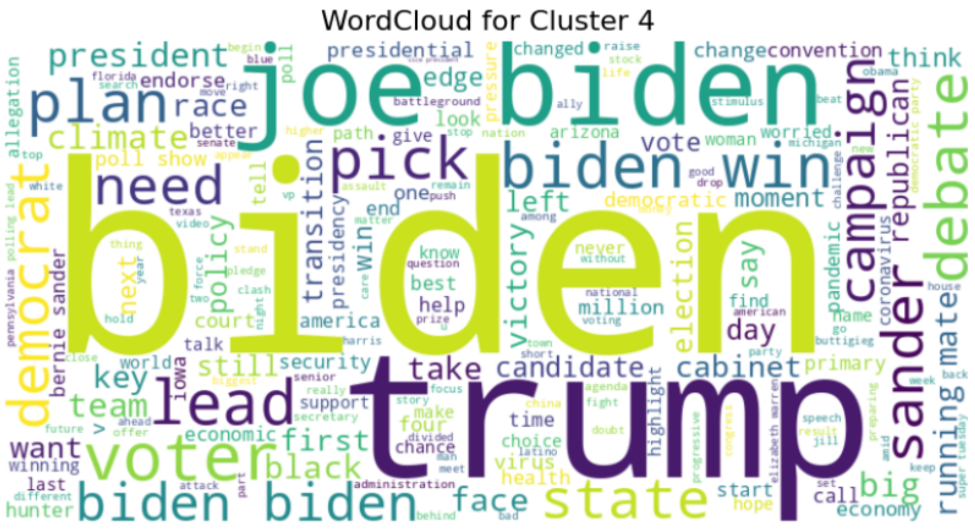


Figure 9: Cluster 4

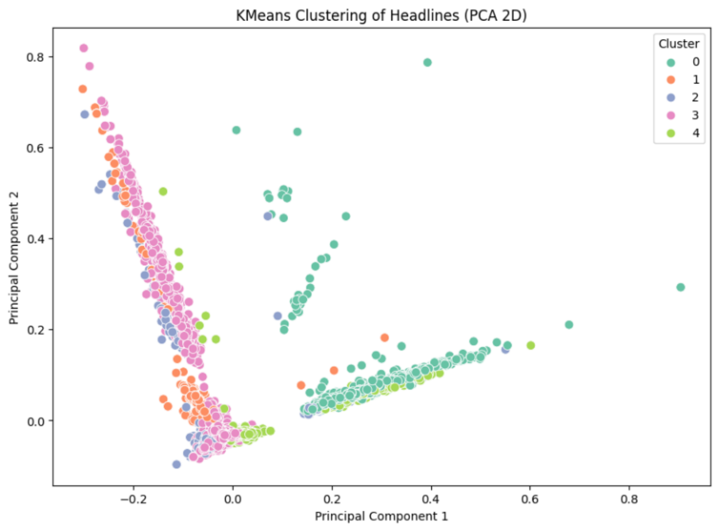


Figure 10: PCA 2D Clusters of Headlines

The PCA (Principal Component Analysis) 2D plot visualizes the distribution of these clusters, revealing that although there is some overlap, the clusters generally form distinct groups, indicating effective separation of topics.

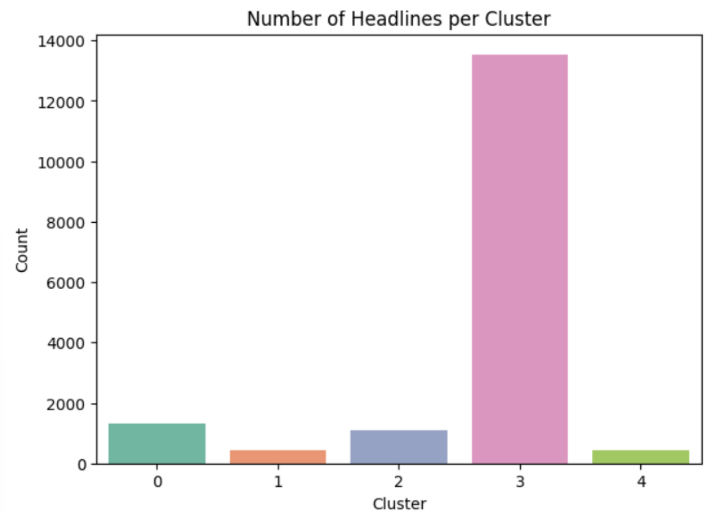


Figure 11: Cluster Distribution

The cluster distribution plot shows that Cluster 3 dominates, indicating that a single major theme captured most headlines, while Clusters 0, 1, 2, and 4 represent less prominent topics in 2020.

Latent Dirichlet Allocation (LDA):

Unlike K-Means, where each document is assigned to exactly one cluster, LDA allows a document to belong to multiple topics with different probabilities.

LDA Clustering Results:
Topic 0: Related to New York, year, and city.
`['new', 'york', 'year', 'time', 'one', 'city', 'way', 'could', 'next', 'thing']`

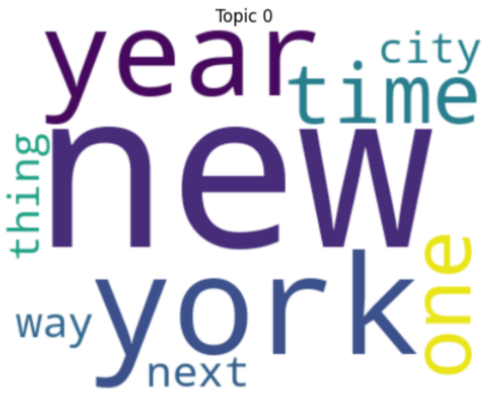


Figure 12: Topic 0

Topic 1: Political topics involving Biden, Trump, and election.
`['trump', 'biden', 'election', 'day', 'dy', 'court', 'republican', 'president', 'say', 'week']`

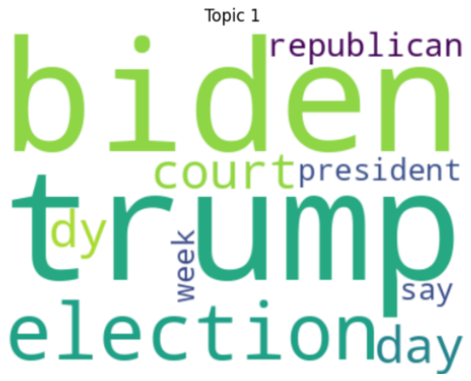


Figure 13: Topic 1

Topic 2: Pandemic-focused words like coronavirus, COVID, death, and risk.
`['coronavirus', 'america', 'covid', 'world', 'death', 'trump', 'may', 'pandemic', 'risk', 'virus']`

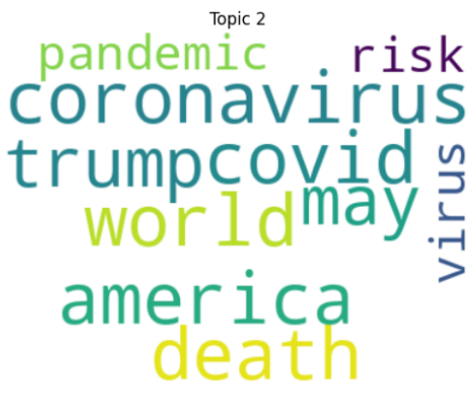


Figure 14: Topic 2

Topic 3: Home and life-related themes like home, life, and million.
['home', 'life', 'million', 'review', 'face', 'need', 'american', 'virus', 'california', 'may']

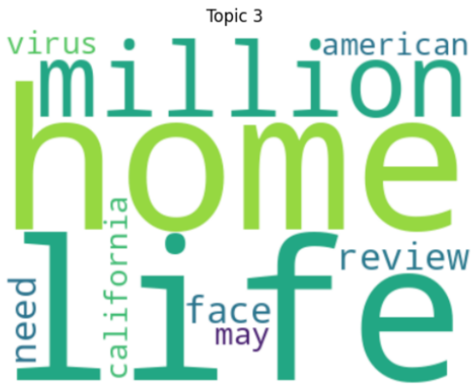


Figure 15: Topic 3

Topic 4: Topics covering pandemic, China, work, and virus.
['coronavirus', 'pandemic', 'get', 'like', 'day', 'work', 'black', 'want', 'china', 'back']

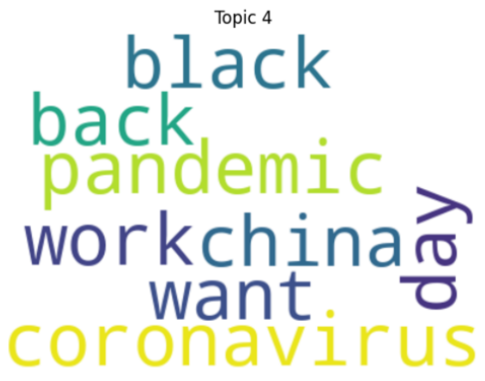


Figure 16: Topic 4

Explanation for "Trump" Appearing Twice:

The word *"Trump"* appears twice across different clusters and topics due to the contextual significance of the term in different discussions.

1. **Cluster 0** and **Topic 1**: In these clusters, "Trump" is associated with political discussions, particularly regarding elections, leadership, and administration. The mention of "Trump" here reflects his role as a key political figure, particularly in the context of his involvement in the 2020 election cycle and his political influence.

2. **Cluster 4** and **Topic 2**: In contrast, the word "Trump" in this cluster is more linked to the broader context of the pandemic, his political stance, and his government's response to the crisis. The word is frequently associated with discussions involving the pandemic, policy, and the political climate during 2020.

This repetition highlights the multi-dimensional role that "Trump" plays in both political discourse (related to the election and governance) and in discussions of public health and policy during the COVID-19 pandemic. Thus, while the word "Trump" appears twice, it represents different thematic contexts in the clusters and topics.

Both **KMeans** and **LDA** successfully identified meaningful and relevant topics from the NY Times headlines of 2020.

- **KMeans** provided **clear, distinct groups** of headlines.
- **LDA** offered a more **flexible** view, showing how headlines might relate to **multiple topics** at once.

This task highlights the strength of **unsupervised learning** techniques in discovering hidden patterns in text data without the need for labeled datasets.

Task 3: Headlines Generation

Website

To generate realistic and safe news headlines similar to The New York Times using a custom-trained LSTM model.

Data Preparation & Embedding

- Collected NYT headlines and cleaned the text (lowercasing, removing punctuation, etc.).
- Used Keras Tokenizer to convert text into integer sequences (word index).
- Created n-gram input sequences to train the model to predict the next word.
- Applied padding to ensure uniform input length.
- Passed tokenized input through an Embedding layer which maps each word index to a dense vector, capturing semantic relationships between words.

Model Architecture & Training

Input $\rightarrow$ Embedding Layer $\rightarrow$ LSTM Layer $\rightarrow$ Dense Layer (ReLU) $\rightarrow$ Dense Layer (Softmax)

- Input Layer: The model takes as input a sequence of word indices, where each index corresponds to a word in a headline. These sequences are created by progressively building n-grams; for example, the headline "the economy is improving" becomes [1, 5, 8, 22], forming pairs like [1, 5] $\rightarrow$ 8 and [1, 5, 8] $\rightarrow$ 22 to predict the next word.
- Embedding Layer: This first layer maps each word ID to a dense vector in a lower-dimensional space, capturing semantic relationships between words. An embedding matrix of size (Vocabulary $\times$ Embedding dimension), such as 5000 $\times$ 100, is learned during training, enabling similar words to have similar vector representations.
- LSTM Layer: It processes the sequence of embeddings, capturing long-term dependencies and patterns across words. It outputs a hidden state summarizing the sequence context, helping the model predict the most likely next word based on previous inputs.
- ReLU Activation: It addes non-linearity and enabling the learning of richer, complex features in the model.
- Softmax Acitvation: It generates a probability distribution over the vocabulary, predicticting the next word by either choosing the one with the highest probability or sampling based on a controlled randomness using temperature.
- Semantic Safety Filtering It uses a zero-shot classification pipeline from HuggingFace Transformeres. Each generated headline was checked against a list of sensitive topics: violence, tragedy, politics, hate speech. Headlines with high semantic similarity to these topics were filtered out, ensuring outputs are safe and neutral.

Text Generation
To generate text, a seed phrase such as "tech stocks" is provided as input. The model then predicts one word at a time, using the combination of the initial seed and each previously generated word to guide the next prediction. To control the creativity of the output, temperature sampling is applied: a low temperature (e.g., 0.2) makes the model's predictions more focused, repetitive, and conservative, while a high temperature (e.g., 1.0) encourages more diverse and creative outputs. The generation process continues until either a predefined maximum length is reached or an end token is predicted.

Text Genertaion using Transformers
To improve headline generation quality, we utilized a fine-tuned Transformer model, specifically FLAN-T5-small, from Google. FLAN-T5 is an encoder-decoder style Transformer architecture designed for text-to-text tasks, where both the input and output are text strings. The model was accessed through the Hugging Face transformers pipeline and generates headlines by first encoding the input prompt (seed text) into a hidden representation using the encoder, and then decoding it into a natural language headline using the decoder. Key features of the model include multi-head attention, layer normalization, and positional encodings that help manage sequential information without recurrence. The model was prompted with carefully crafted instructions such as "Generate a short, meaningful news headline based on:" followed by the seed text. Sampling parameters like max length were controlled to ensure concise and relevant outputs. Sample seed inputs like "Breaking news about technology" and "Wall Street earnings" successfully produced realistic and meaningful headlines, demonstrating the capability of FLAN-T5-small in handling real-world headline generation tasks effectively.

Here are the sentences used in the table:

Sentence 1: Tech stocks re-entered the market, with a strong performance.

Sentence 2: White House officials announced a new bill to allow the state to make a tax cut.

Sentence 3: A new smartphone is coming out, and the company is releasing a new version.

Sentence 4: Tech stocks showed remarkable recovery, with analysts optimistic.

Sentence 5: The government unveiled a plan to support innovation in the technology sector.

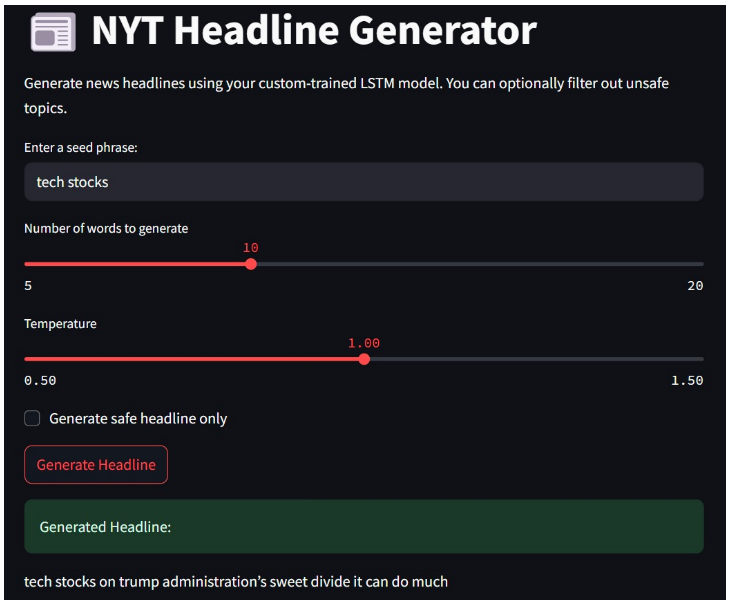
Sentence 6: The latest model of the smartphone features groundbreaking technology.

Generated by Transformer	Generated by App	Grammar Score (Transformer)	Grammar Score (App)
Sentence 1.	Sentence 4.	9/10	7/10
Sentence 2.	Sentence 5.	8/10	6/10
Sentence 3.	Sentence 6.	9/10	5/10

Table 1: Comparison of Generated Text by Transformer and Our App, and Their Grammar Scores

Instance of generated headlines with safe mode off and on with seed input ”tech stocks”:

- Safe Mode OFF (No filtering): ”tech stocks on trump administration’s sweet divide it can do much ”



NYT Headline Generator

Generate news headlines using your custom-trained LSTM model. You can optionally filter out unsafe topics.

Enter a seed phrase: tech stocks

Number of words to generate: 10 (5–20)

Temperature: 1.00 (0.50–1.50)

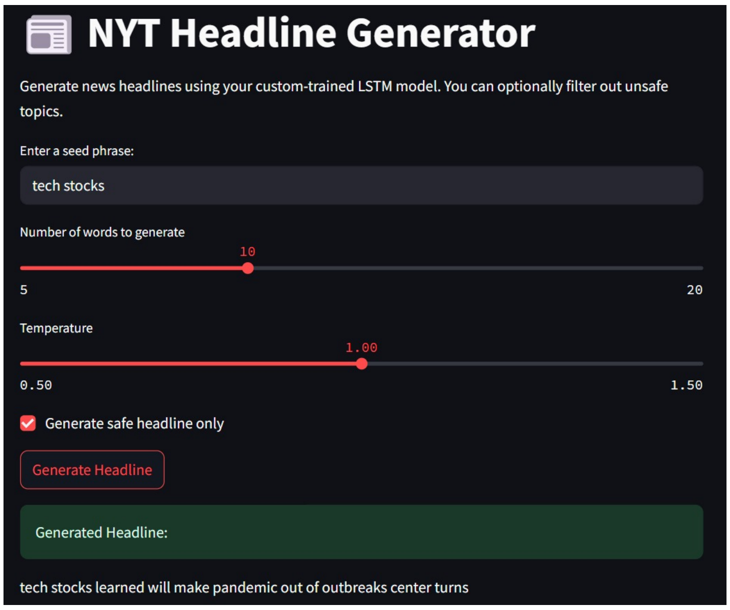
☐ Generate safe headline only

Generated Headline:

tech stocks on trump administration’s sweet divide it can do much

Figure 17: Safe Mode OFF

- Safe Mode ON (filtered): ”tech stocks learned will make pandemic out of outbreaks center turns”



NYT Headline Generator

Generate news headlines using your custom-trained LSTM model. You can optionally filter out unsafe topics.

Enter a seed phrase: tech stocks

Number of words to generate: 10 (5–20)

Temperature: 1.00 (0.50–1.50)

☑ Generate safe headline only

Generated Headline:

tech stocks learned will make pandemic out of outbreaks center turns

Figure 18: Safe Mode ON

Task 4: Classify Times Pick Comments

Initial Experiments on 50,000 Points dataset with XGBoost:

We use XGBoost classifier model because it is known for high performance on structured data, handles imbalances reasonably with weighting, and is very fast for prototyping. The model performed well overall, achieving 92% accuracy. Class 0 (the majority class) showed excellent precision, recall, and F1-score around 95%, while Class 1 (the minority class) had decent but noticeably lower scores, with an F1 of around 84%.

Class	Precision	Recall	F1-score
0	0.94	0.95	0.95
1	0.85	0.82	0.84
Accuracy		**0.92**	

This indicated that while XGBoost could manage moderate imbalance, it struggled to give equal attention to the rarer class.

Full Dataset with XGBoost

When scaling up to the full Times Pick dataset, the imbalance became more severe. Though the overall accuracy improved to 96.74%, this was misleading: Class 0's performance remained near perfect, but Class 1 detection collapsed entirely, with extremely low precision, recall, and F1-score (only about 3% F1).

Class	Precision	Recall	F1-score
0	0.9889	0.9780	0.98
1	0.0257	0.0502	0.0340
Accuracy		**0.9674**	

It became clear that simple boosting models would not be enough, and that a model with deeper understanding especially of the text content was necessary.

BERT-Based Modelling

BERT was chosen for its ability to capture rich contextual information from the text, allowing the model to recognize subtle cues that differentiated an editor's pick from a regular article. BERT's outputs were further combined with metadata features to ground the model in structured information.

This new model significantly improved Class 1 recall to around 86%, although precision was still low at 16%. The F1-score for the minority class rose to about 27%, marking a substantial gain.

Class	Precision	Recall	F1-score
0	0.9983	0.9488	0.97729
1	0.1627	0.8591	0.2736
Accuracy		**0.9478**	

However, since BERT models can be very sensitive and produce many false positives, further adjustment was necessary. To refine the model, precision-recall versus threshold analysis was performed.

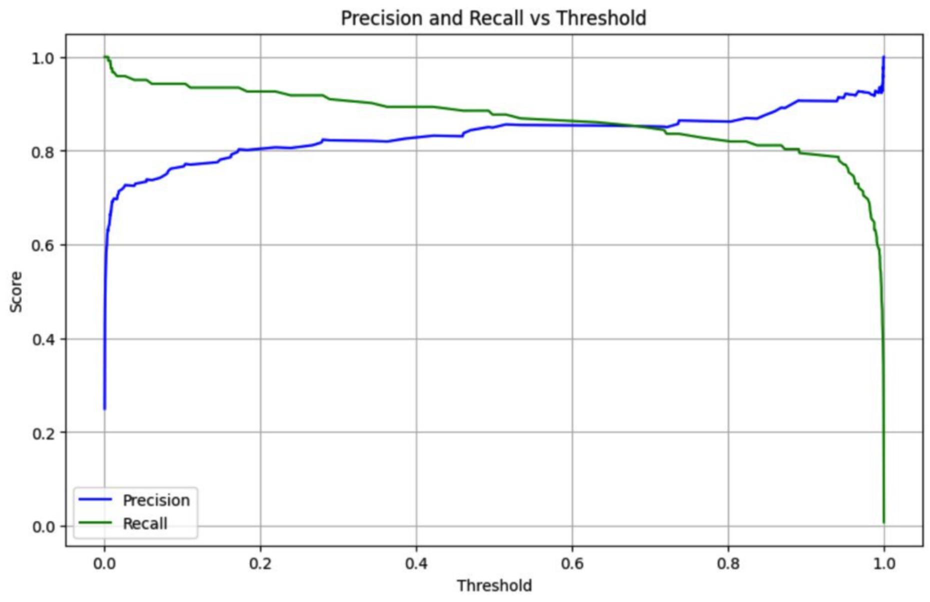


Figure 19: Precison & Recall Vs Threshold

It was found that the default classification threshold of 0.5 was too low, resulting in too many false positives. By increasing the threshold to 0.7, a better trade-off between precision and recall was achieved. After retraining with this optimized threshold, Class 1 precision rose to about 26%, recall stayed strong at 75%, and the F1-score improved to 38.5% – the highest so far. Importantly, Class 0 maintained its excellent performance, and overall model accuracy climbed to 97.25%, with strong macro and weighted F1-scores.

Class	Precision	Recall	F1-score
0	0.9971	0.9751	0.9860
1	0.2590	0.7523	0.3853
Accuracy		**0.9725**	

BERT-Based Modelling With SMOTE

Class	Precision	Recall	F1-score
0	0.9035	0.9489	0.9256
1	0.9462	0.8956	0.9218
Accuracy		**0.9237**	

After applying SMOTE, the model's accuracy dropped slightly to 92.37%, but the class-wise performance became much more balanced. Both Class 0 and Class 1 achieved high and comparable precision (0.90+) and F1-scores (0.92), highlighting a significant improvement in the model's ability to handle the minority class. Thus, while there is a trade-off in overall accuracy, the model becomes far more reliable across both classes.

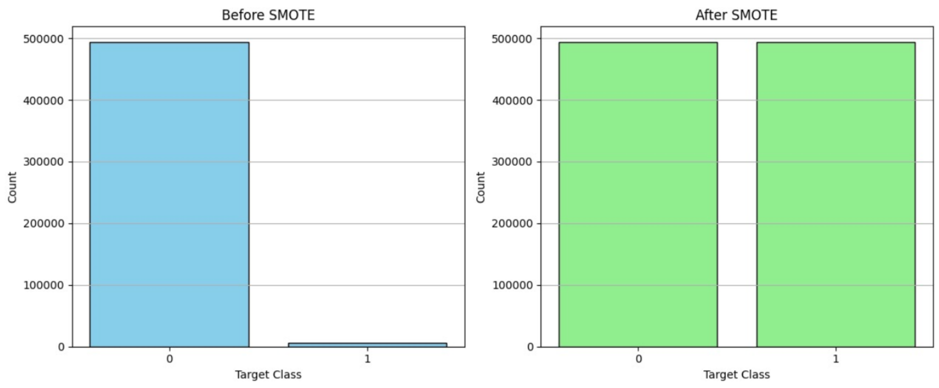


Figure 20: Distribution of Class 0 & 1 with/without SMOTE

The distribution plots further support this. Before SMOTE, Class 0 heavily dominated the dataset, while Class 1 had very few samples. After applying SMOTE, the two classes were balanced, allowing the model to learn from a more representative dataset and generalize better across classes.

Conclusion

This project demonstrated the use of machine learning for analyzing user engagement and editorial preferences in digital journalism. We successfully built predictive models for comment behavior and explored latent themes in article content. Future work could involve deeper sentiment analysis, transformer-based models for better text representation, and time-series forecasting to track engagement trends over time.

Reference

[1] B. Awad, "New York Times Articles & Comments (2020)," Kaggle, 2020. [Online]. Available: Website [Accessed: 27-Apr-2025].

[2] "Code Repository for New York Times Articles & Comments (2020) Analysis," GitHub, 2025. [Online]. Available: GitHub [Accessed: 27-Apr-2025].