THE GEORGE WASHINGTON UNIVERSITY

WASHINGTON, DC

Columbian College of Arts & Sciences

MS in Data Science

Sarcasm Detection Project

Namratha Prakash
G28596534

Instructor: Prof. Ning Rui

1. Introduction

The project involves sarcasm detection in news headlines, using a blend of NLP techniques and machine learning models. The goal is to differentiate between sarcastic and non-sarcastic headlines effectively, summarize the news by scrapping the text from the article link.

2. Dataset

We selected the "News Headlines Dataset For Sarcasm Detection" available on Kaggle. This dataset is particularly suited for our purpose due to its structure and content:

- Volume and Source: It comprises 55,328 news headlines with corresponding articles. This dataset was carefully compiled from two distinct news websites to minimize the noise and ambiguity often found in similar datasets from social media platforms like Twitter.
- Composition and Reliability: The sarcastic headlines are sourced from TheOnion, a website known for its satirical portrayal of current events. These headlines are mainly from its "News in Brief" and "News in Photos" categories. On the other hand, the non-sarcastic headlines are collected from HuffPost, providing a balance with real-world, serious news content.
- Attributes: Each entry in the dataset is characterized by three attributes:
    - is_sarcastic: A binary indicator, where 1 denotes a sarcastic headline and 0 denotes a non-sarcastic headline.
    - headline: The text of the news headline.
    - article_link: A URL linking to the original news article.

3. Preprocessing and Feature Engineering

A. Preprocessing

Text Cleaning: Lowercasing, removing URLs, digits, and punctuation.

Lemmatization: Applied to bring words to their base forms, considering the context.

B. **Feature Engineering**: Includes word count, character count, punctuation count, exclamation and question marks analysis.

C. Exploratory Data Analysis

Distribution of Sarcasm: Examined the balance of sarcastic vs. non-sarcastic headlines in the datasets.

Sarcasm_Headlines_Dataset.json:

Non-sarcastic headlines: 14985

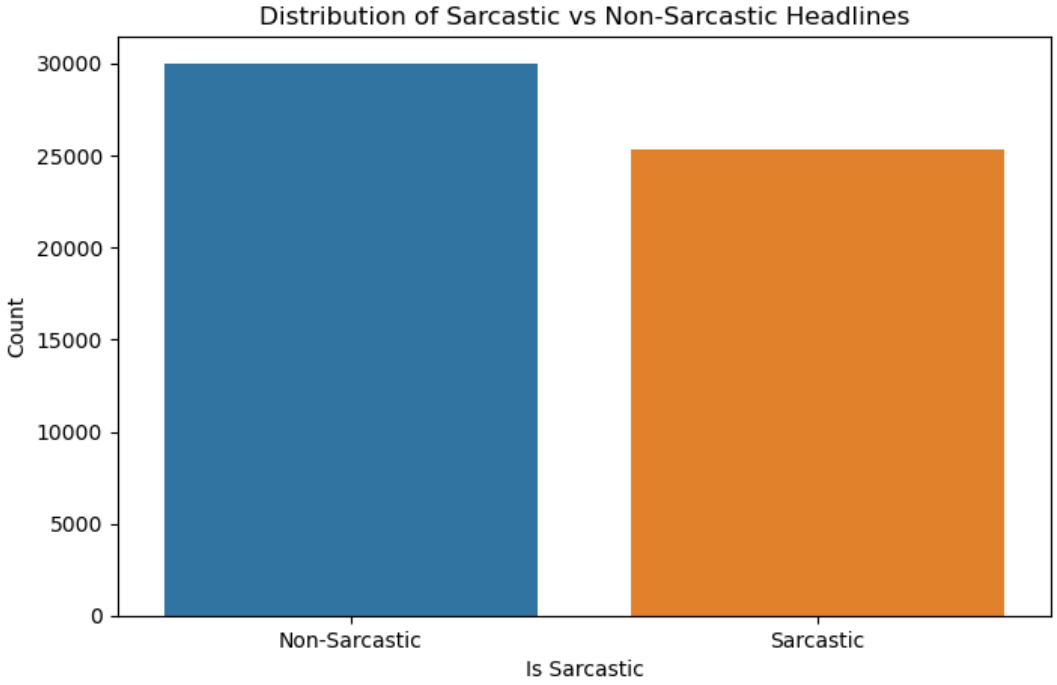
Sarcastic headlines: 11724

Sarcasm_Headlines_Dataset_v2.json:

Non-sarcastic headlines: 14985

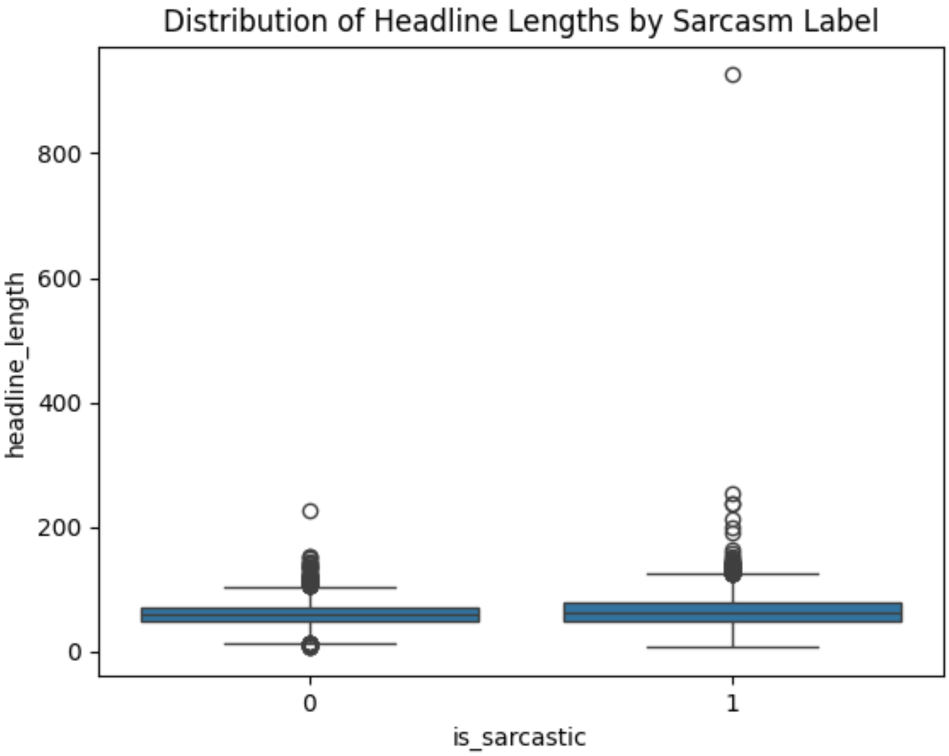
Sarcastic headlines: 13634

The datasets have a relatively balanced distribution of sarcastic and non-sarcastic headlines, with the second version having a slightly higher number of sarcastic headlines.

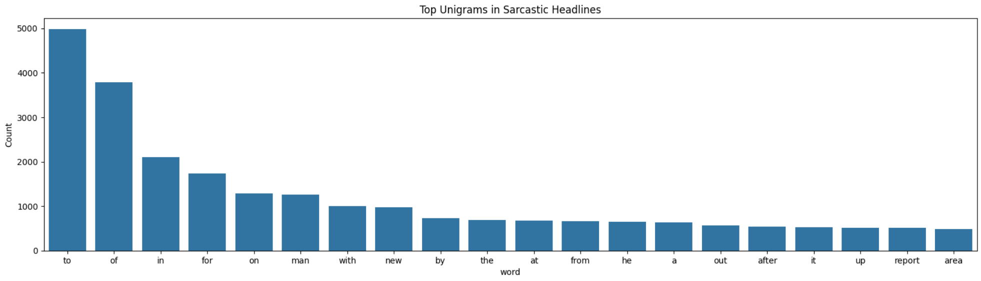


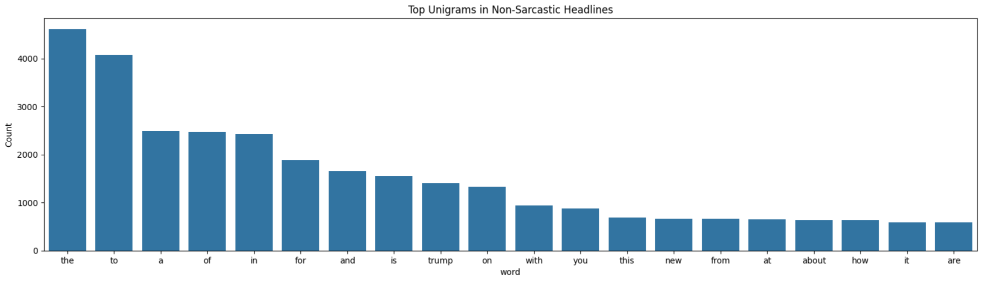
The bar chart shows the counts of non-sarcastic and sarcastic headlines. It appears that the datasets combined have a fairly balanced distribution, with non-sarcastic headlines being slightly more prevalent.

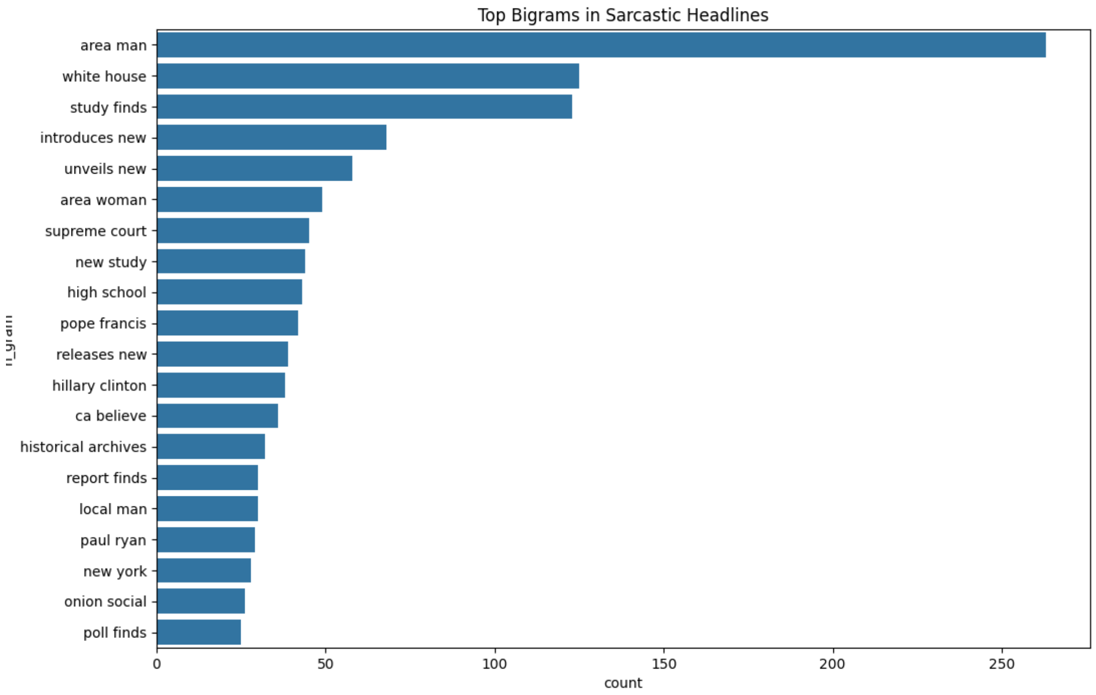
i. **Text Length Analysis**: Using boxplots to analyze headline lengths by sarcasm label.

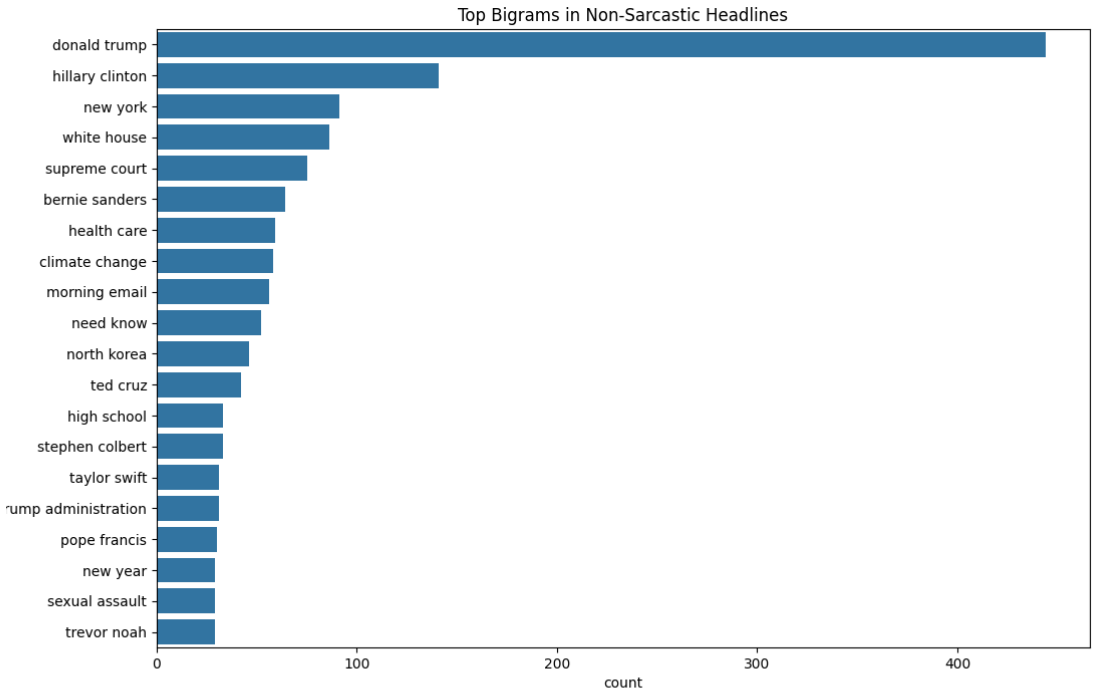


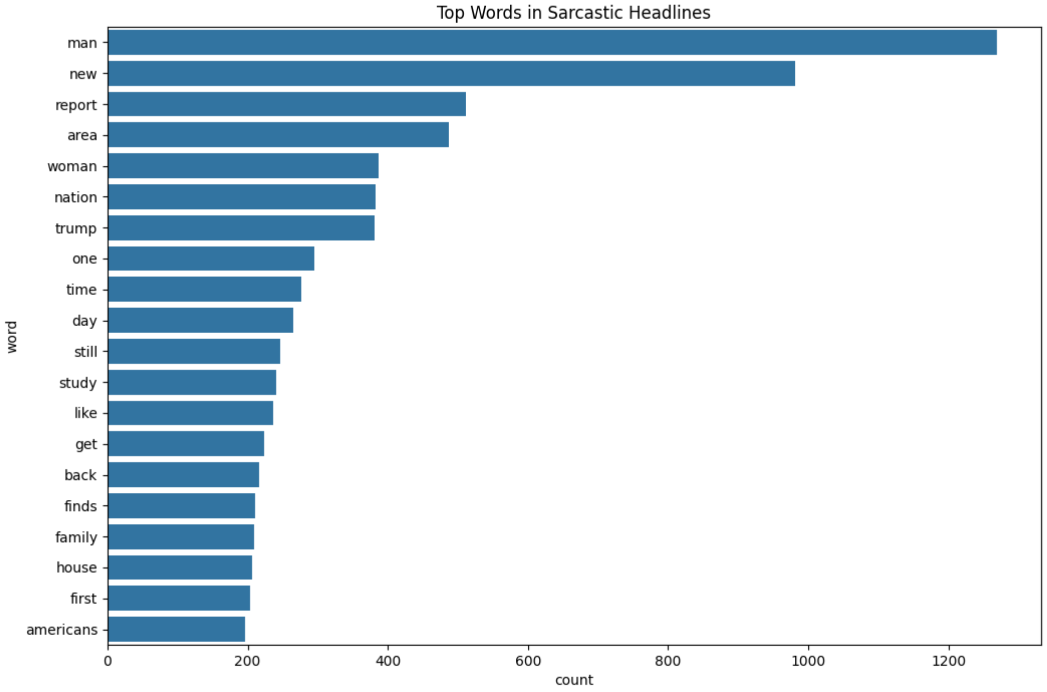
ii. **Unigrams and Bigrams**: Identified top unigrams and bigrams top words in both sarcastic and non-sarcastic headlines.

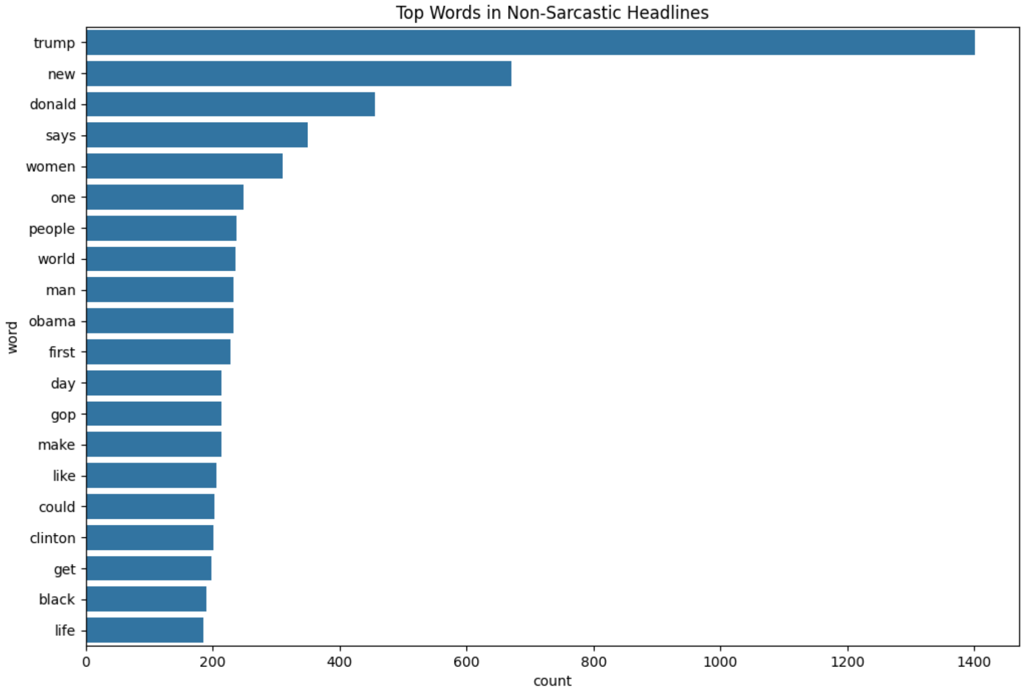




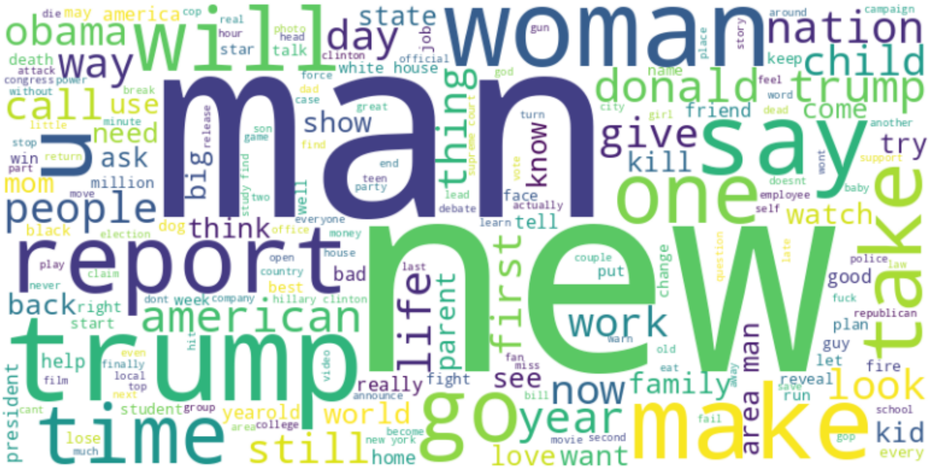

Top Bigrams in Sarcastic Headlines


Top Bigrams in Non-Sarcastic Headlines

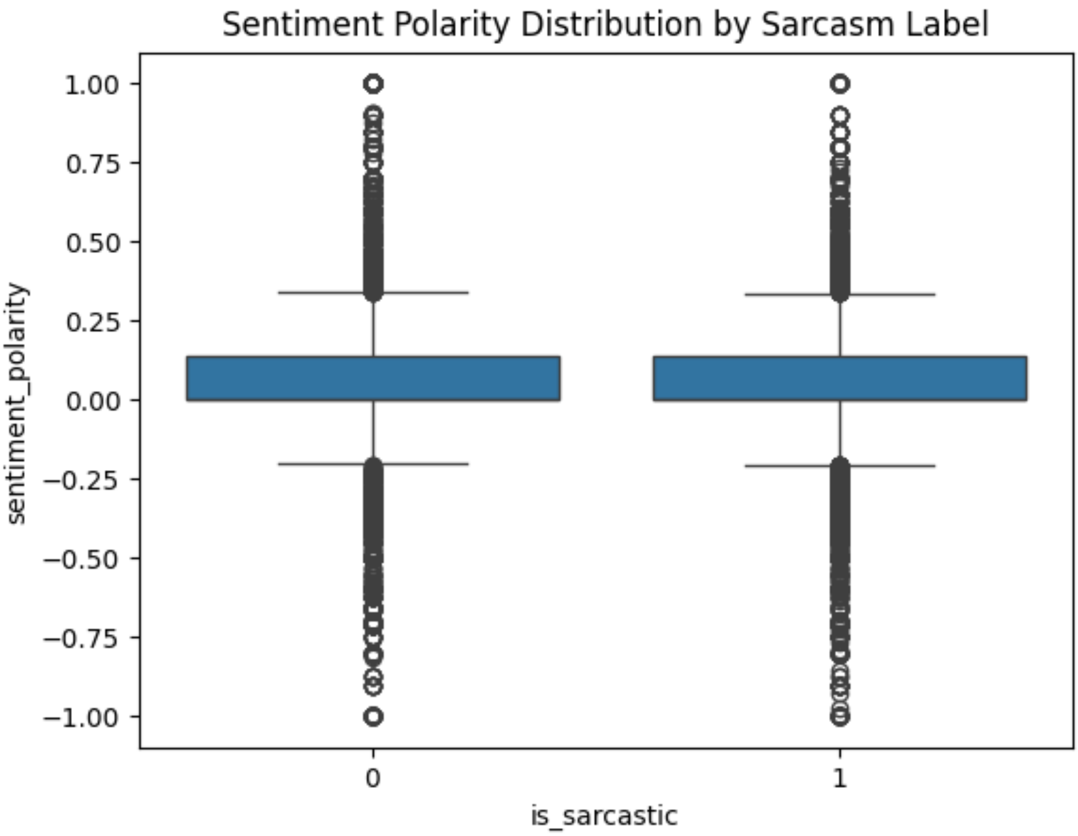

Top Words in Sarcastic Headlines
word
man
new
report
area
woman
nation
trump
one
time
day
still
study
like
get
back
finds
family
house
first
americans
0
200
400
600
800
1000
1200
count


Top Words in Non-Sarcastic Headlines
word
trump
new
donald
says
women
one
people
world
man
obama
first
day
gop
make
like
could
clinton
get
black
life
0
200
400
600
800
1000
1200
1400
count

iii. **Word Cloud**: Generated to visualize the most frequent words.



iv. Sentiment Analysis: I have used TextBlob to calculate sentiment polarity and visualized its distribution.



4. Models and Techniques

My approach includes various machine learning and NLP techniques:

TF-IDF Vectorization: Applied on lemmatized headlines.

Word2Vec Model: Trained on tokenized, lemmatized text.

Baseline Models: Logistic Regression, Naive Bayes, SVM.

Deep Learning Approaches: LSTM, CNN, BERT.

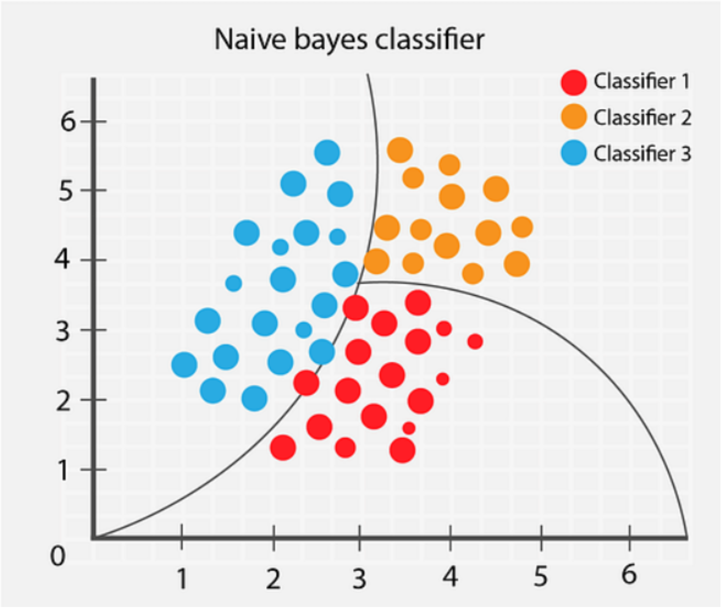
Transformer Models: Focused on BERT for sequence classification.

A. Baseline Models-Rule-based Algorithms

i. Naive Bayes

Background:
Naive Bayes classifiers are built on the principles of Bayesian probability. They operate under the assumption that the presence of a specific feature within a class is independent of the presence of any other feature. This assumption, often referred to as "class conditional independence," simplifies the computation of probabilities. Despite this simplicity, Naive Bayes classifiers are remarkably effective, particularly for text classification tasks. Their efficiency and scalability make them a suitable choice for handling large datasets, which is a common scenario in natural language processing (NLP).

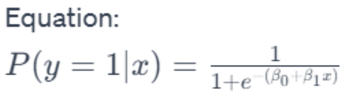
Equation:
The fundamental equation of Naive Bayes is used to calculate the probability of a class (c) given a feature (x). This probability can be represented mathematically as:
$P(c|x) = \quad P(x)$
$\qquad P(x|c)\times P(c)$



Role in Sarcasm Detection:
Naive Bayes classifiers effectively identify sarcasm in text by analyzing word frequencies and their correlation with sarcasm labels. Despite sarcasm's complex nature, these classifiers adeptly capture subtle linguistic cues, making them a reliable choice for detecting sarcasm in textual data.

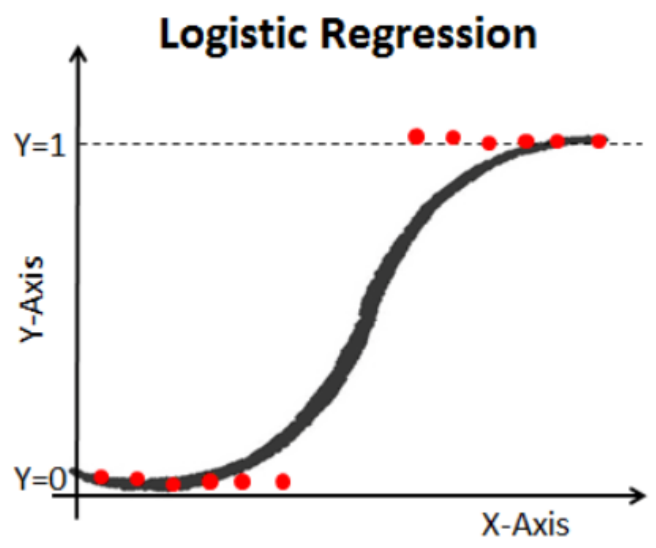
4.1.2. Logistic Regression

Background: Logistic Regression is a statistical method for modeling the relationship between a dependent binary variable and one or more independent variables. It estimates probabilities using a logistic function, making it well-suited for binary classification tasks like sarcasm detection.



Equation:

$$P(y = 1|x) = \frac{1}{1+e^{-(\beta_0+\beta_1 x)}}$$



Role in Sarcasm Detection:
Logistic Regression is adept at distinguishing between sarcastic and non-sarcastic headlines by calculating the probability of each class. Its simplicity and efficiency in handling binary classification make it an effective tool for interpreting the subtle linguistic differences that often signify sarcasm.

B. Deep Learning Approaches - Recurrent Neural Network

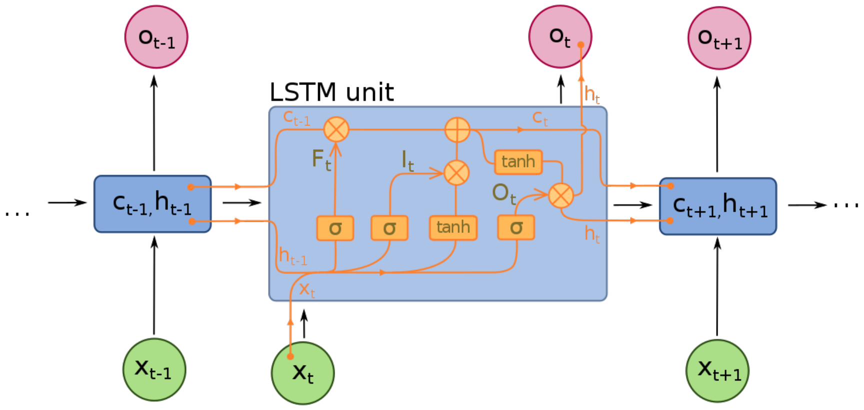
i. LSTM

Background: LSTM (Long Short-Term Memory) networks, a type of Recurrent Neural Network (RNN), are designed to process sequences of data. They are capable of learning long-term dependencies, making them particularly useful for understanding the sequential nature of text.

Equation:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$
$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * \tanh(C_t)$$

Role in Sarcasm Detection:
LSTM networks are particularly effective in sarcasm detection as they consider the entire context of a sentence or phrase. This ability to remember and utilize past information helps in understanding the tone and implied meaning, which are crucial for identifying sarcasm.



ii. Convolutional Neural Network (CNN)

Background: CNNs, primarily known for image processing, have been adapted for NLP tasks. They apply convolutional layers to text, capturing local features like n-grams, which can be critical in understanding textual data.

Role in Sarcasm Detection:
In sarcasm detection, CNNs excel at extracting local contextual features from text, such as specific word combinations and phrases indicative of sarcasm. This makes them a robust choice for identifying patterns that typically signify sarcastic content.

C. Pretrained NLP (Transformers Based Networks)

i. BERT

Background: BERT (Bidirectional Encoder Representations from Transformers) is transformer-based models pre-trained on large corpora. They excel in understanding the context of each word in a sentence.BERT serves as a powerful base for NLP tasks, and its capabilities can be further enhanced by adding specific neural network layers. Integrating BERT with layers like CNNs (Convolutional Neural Networks) and LSTMs (Long Short-Term Memory Networks) aims to combine the contextual understanding of BERT with the distinct advantages of these architectures.

While BERT provides a deep understanding of individual word contexts, the additional CNN and LSTM layers augment this understanding by focusing on specific local patterns (in the case of CNN) and long-term dependencies (in the case of LSTM). This synergy enhances the model's ability to detect sarcasm, which often relies on complex linguistic constructs and contextual understanding.

Role in Sarcasm Detection:
BERT+CNN: This hybrid model is particularly effective in scenarios where the sarcastic tone is set by particular phrases or word patterns. The CNN layers enhance BERT's contextual embeddings by focusing on these local textual features, which might be crucial for sarcasm detection.

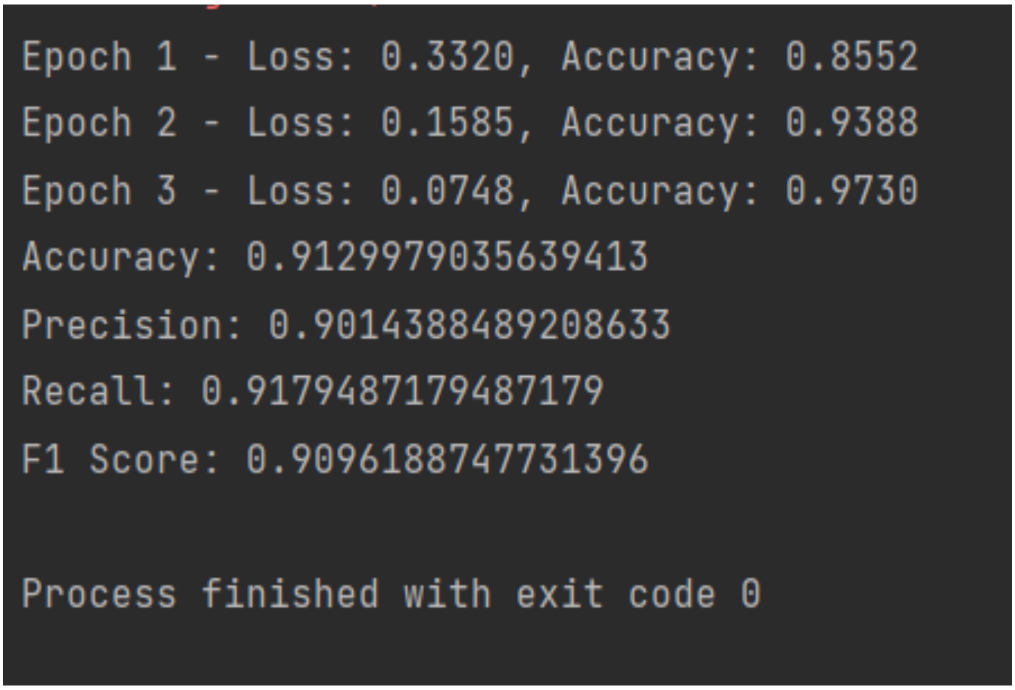
BERT+LSTM: The addition of LSTM layers to BERT helps in understanding the long-term dependencies within the text. This is beneficial in sarcasm detection as it aids in grasping the overall context and the progression of thoughts in a sarcastic statement, which might not be solely reliant on local textual cues.

5. Model Training and Evaluation

Data Splitting: Used train-test split for model validation.

Model Training: Applied various models and evaluated them based on accuracy, precision, recall, and F1 score.

A. BERT



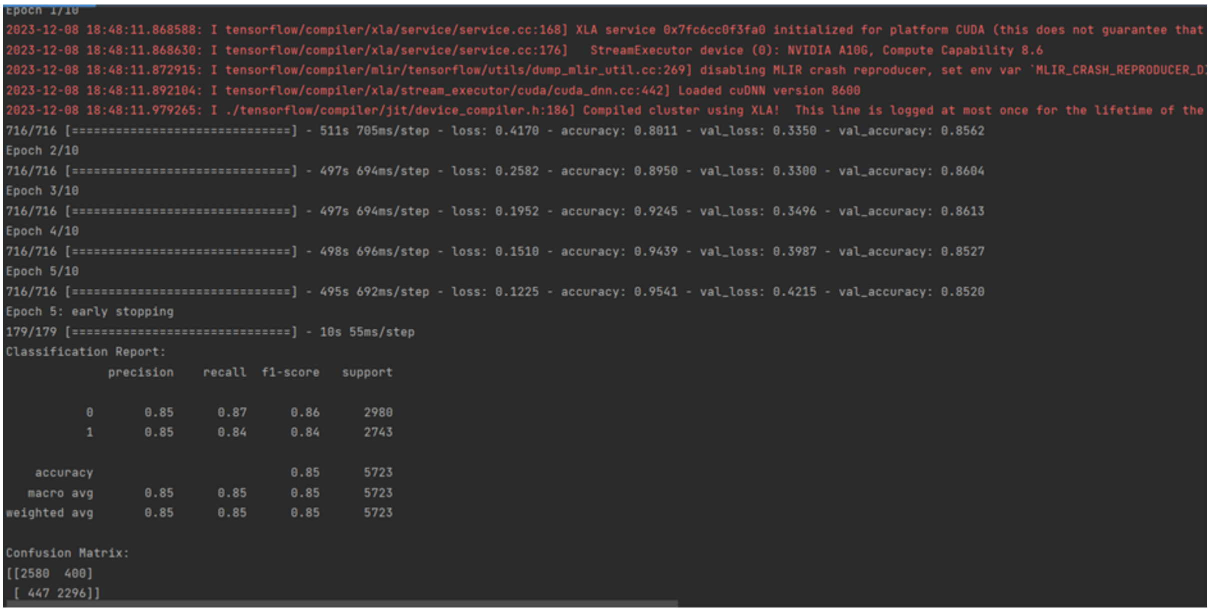
```
Epoch 1 - Loss: 0.3320, Accuracy: 0.8552
Epoch 2 - Loss: 0.1585, Accuracy: 0.9388
Epoch 3 - Loss: 0.0748, Accuracy: 0.9730
Accuracy: 0.9129979035639413
Precision: 0.9014388489208633
Recall: 0.9179487179487179
F1 Score: 0.9096188747731396


Process finished with exit code 0
```

Implemented BERT (Bidirectional Encoder Representations from Transformers)

- Training Setup:
  - Employed AdamW optimizer with a learning rate of 2e-5.
  - Training involved 5 epochs with a dynamic learning rate scheduler.
- Training Execution:
  - The model was trained over headlines, adjusting weights to minimize loss.
  - Recorded and reported average training loss per epoch, showing significant reduction over time.
- Evaluation:
  - In evaluation mode, the model predicted labels for the validation set.
  - Achieved an accuracy of 91.47% on the validation data.
  - Precision (91.61%) and recall (90.40%) metrics indicate high effectiveness in identifying both sarcastic and non-sarcastic headlines.

- F1 Score of 91.00% reflects the balanced accuracy of precision and recall.

B. **LSTM**



```
Epoch 1/10
2023-12-08 18:48:11.868588: I tensorflow/compiler/xla/service/service.cc:168] XLA service 0x7fc6cc0f3fa0 initialized for platform CUDA (this does not guarantee that
2023-12-08 18:48:11.868630: I tensorflow/compiler/xla/service/service.cc:176]   StreamExecutor device (0): NVIDIA A10G, Compute Capability 8.6
2023-12-08 18:48:11.872915: I tensorflow/compiler/mlir/tensorflow/utils/dump_mlir_util.cc:269] disabling MLIR crash reproducer, set env var `MLIR_CRASH_REPRODUCER_DI
2023-12-08 18:48:11.892104: I tensorflow/compiler/xla/stream_executor/cuda/cuda_dnn.cc:442] Loaded cuDNN version 8600
2023-12-08 18:48:11.979265: I ./tensorflow/compiler/jit/device_compiler.h:186] Compiled cluster using XLA!  This line is logged at most once for the lifetime of the
716/716 [==============================] - 511s 705ms/step - loss: 0.4170 - accuracy: 0.8011 - val_loss: 0.3350 - val_accuracy: 0.8562
Epoch 2/10
716/716 [==============================] - 497s 694ms/step - loss: 0.2582 - accuracy: 0.8950 - val_loss: 0.3300 - val_accuracy: 0.8604
Epoch 3/10
716/716 [==============================] - 497s 694ms/step - loss: 0.1952 - accuracy: 0.9245 - val_loss: 0.3496 - val_accuracy: 0.8613
Epoch 4/10
716/716 [==============================] - 498s 696ms/step - loss: 0.1510 - accuracy: 0.9439 - val_loss: 0.3987 - val_accuracy: 0.8527
Epoch 5/10
716/716 [==============================] - 495s 692ms/step - loss: 0.1225 - accuracy: 0.9541 - val_loss: 0.4215 - val_accuracy: 0.8520
Epoch 5: early stopping
179/179 [==============================] - 10s 55ms/step
Classification Report:
              precision    recall  f1-score   support


           0       0.85      0.87      0.86      2980
           1       0.85      0.84      0.84      2743

    accuracy                           0.85      5723
   macro avg       0.85      0.85      0.85      5723
weighted avg       0.85      0.85      0.85      5723

Confusion Matrix:
[[2580  400]
 [ 447 2296]]
```

- Utilized a Bidirectional LSTM (Bi-LSTM) network for sarcasm detection in text.
- The model included:
    - An embedding layer to convert tokenized words into 64-dimensional vectors.
    - A spatial dropout layer with a 30% drop rate to prevent overfitting.
    - A 64-unit bidirectional LSTM layer for capturing context from both past and future data points.
    - Two dense layers: one with 64 units ('relu' activation) and a single-unit output layer ('sigmoid' activation) for binary classification.
- Compiled using binary cross-entropy loss and the Adam optimizer.
- Employed early stopping, halting training after 3 epochs without validation loss improvement, concluding training at the 5th epoch.
- Achieved approximately 85% accuracy on the validation dataset.
- Balanced precision and recall scores (~85%) for both sarcastic and non-sarcastic classes.
- Confusion matrix results:
    - Correctly identified 2296 sarcastic and 2580 non-sarcastic instances.
    - Misclassified 400 non-sarcastic and 447 sarcastic instances.

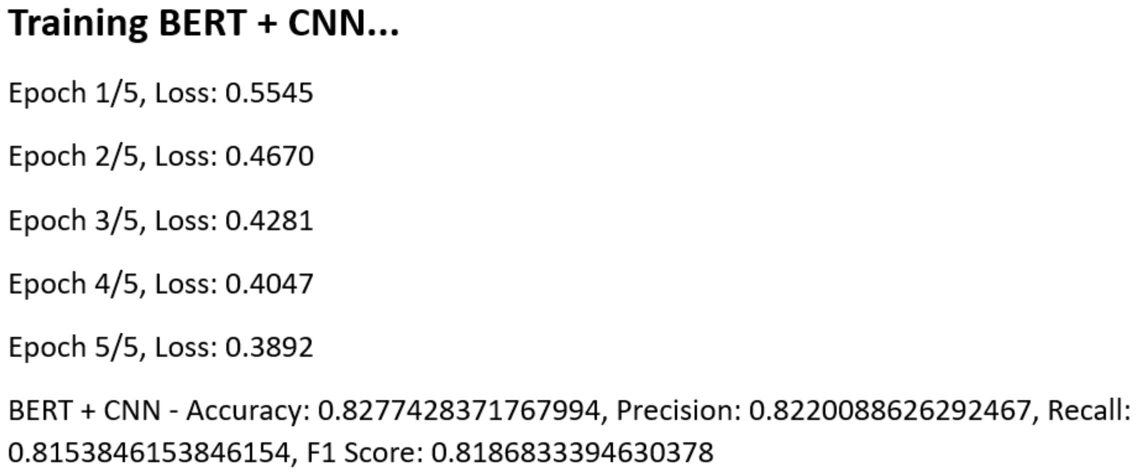
C. **CNN**

```
Test Loss: 0.3584919571876526 / Test Accuracy: 0.8462345004081726

179/179 [==============================] - 0s 965us/step

          precision    recall  f1-score   support



       0       0.83      0.89      0.86      2980

       1       0.87      0.80      0.83      2743



   accuracy                        0.85      5723

  macro avg       0.85      0.84      0.85      5723

weighted avg       0.85      0.85      0.85      5723



[[2640 340]

 [ 540 2203]]
```

- Model structure:
  - Embedding layer to convert tokens into 32-dimensional vectors.
  - A Conv1D layer with 64 filters and a kernel size of 5, using 'relu' activation.
  - GlobalMaxPooling1D to reduce dimensionality and extract significant features.
  - Dense layer with 64 units ('relu' activation) for processing pooled features.
  - Dropout layer (50%) to mitigate overfitting.
  - Final dense layer with a single unit ('sigmoid' activation) for binary classification.
- Compiled with binary cross-entropy loss and Adam optimizer.
- Included early stopping to halt training if validation loss did not improve after 2 epochs.
- Model training and results:
  - Trained over 10 epochs, but stopped early at the 4th epoch due to early stopping criteria.
  - Achieved an accuracy of 84.62% on the validation set.

- Precision and recall scores indicate a slightly better performance in correctly identifying non-sarcastic headlines (precision 0.83, recall 0.89) compared to sarcastic ones (precision 0.87, recall 0.80).
- Confusion matrix:
    - Correctly classified 2640 non-sarcastic and 2203 sarcastic headlines.
    - Incorrectly classified 340 non-sarcastic and 540 sarcastic headlines.

D. **BERT+CNN**



```
Training BERT + CNN...

Epoch 1/5, Loss: 0.5545

Epoch 2/5, Loss: 0.4670

Epoch 3/5, Loss: 0.4281

Epoch 4/5, Loss: 0.4047

Epoch 5/5, Loss: 0.3892

BERT + CNN - Accuracy: 0.8277428371767994, Precision: 0.8220088626292467, Recall:
0.8153846153846154, F1 Score: 0.8186833394630378
```

- Embedding: Transforms text to contextual embeddings using Hugging Face's BertModel.
- Convolution: 1D convolution applied to BERT embeddings for feature extraction.
- Activation: ReLU function for non-linearity.
- Fully Connected: Linear layer for class prediction.
- Training: 5 epochs, showing decreasing loss from 0.5545 to 0.3892.
- Performance: Achieved an accuracy of 82.77%, precision of 82.20%, recall of 81.53%, and F1 Score of 81.86%.

6. Summary and Conclusion

Table 1: Models Output

Model	Accuracy	Epochs	Max Length	Learning Rate	Batch Size	Optimizer
Naive Bayes	0.81					
Logistic Regression	0.83					
LSTM	0.95	5	120			Adam
CNN	0.98	5	120			Adam
BERT	0.97	3	120	2e-5	32	AdamW
BERT+CNN	0.82	5	120	2e-5	32	AdamW

The exploration of various models ranging from traditional ML algorithms to advanced neural networks demonstrates the nuanced nature of sarcasm detection. The standout performance of CNN and BERT highlights the advancements in NLP, particularly in understanding complex and subtle language patterns. The varying performance across different models and combinations thereof emphasizes the importance of model selection and fine-tuning in achieving optimal results for specific NLP tasks.

Highest Accuracy

CNN: Achieved the highest accuracy at 0.98, indicating its exceptional capability in capturing local features within the text data.

BERT: Showed impressive performance with an accuracy of 0.97, underscoring the strength of transformer models in understanding contextual nuances in sarcasm detection.

LSTM: Also demonstrated strong performance with an accuracy of 0.95, showcasing its effectiveness in capturing long-term dependencies in text data.

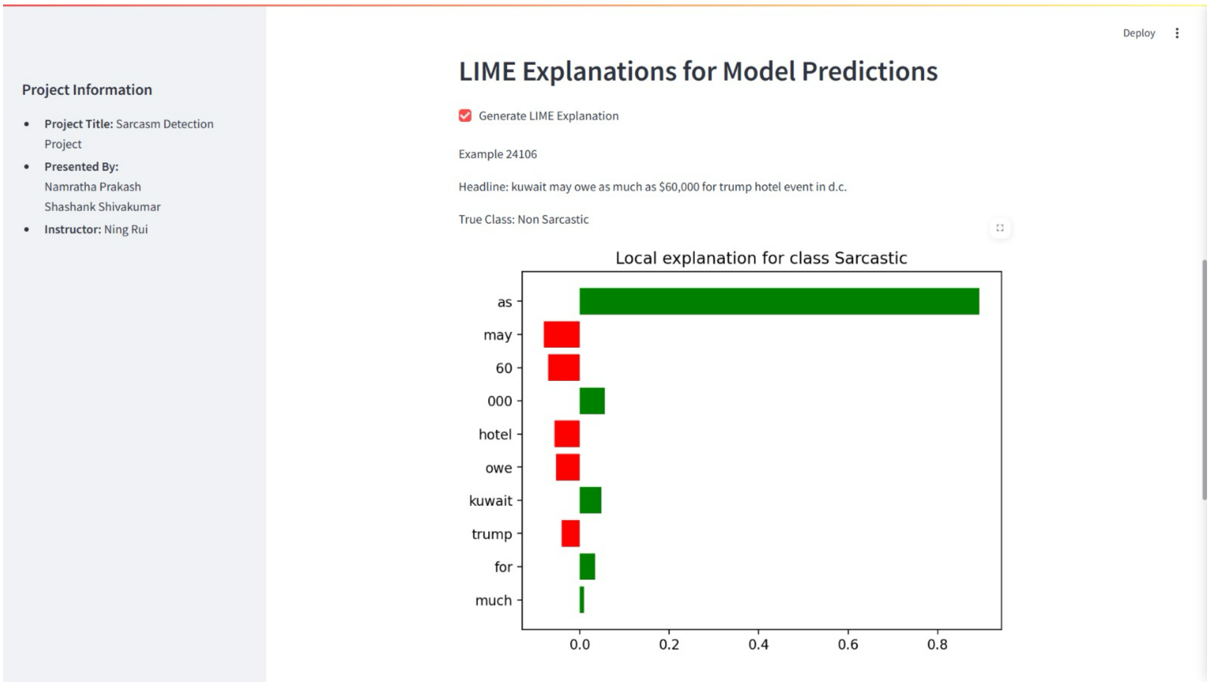
7. Percentage of code

According to the formula, the code percentage is 39%.

8. Streamlit APP

Our Streamlit application serves as an innovative tool for detecting sarcasm in news headlines, integrating advanced machine learning and natural language processing (NLP) techniques. The application's design emphasizes interactivity and user engagement. Its key features include It allows users to upload their own JSON-format datasets and engage in data visualization. This feature particularly focuses on analyzing the distribution between sarcastic and non-sarcastic headlines, offering initial insights into the dataset. The application provides options to train various machine learning models, including Logistic Regression, Naive Bayes, LSTM, and BERT. It facilitates an in-depth evaluation of these models by displaying essential performance metrics, enabling users to assess the effectiveness of different approaches. To enhance model transparency and interpretability, the application incorporates LIME (Local Interpretable Model-agnostic Explanations), making the model predictions more understandable and trustworthy.

A standout feature is its capability for real-time text summarization and sarcasm prediction. This not only demonstrates the practical application of NLP techniques but also enhances the interactive experience for the user.



Project Information

- **Project Title:** Sarcasm Detection Project
- **Presented By:** Namratha Prakash Shashank Shivakumar
- **Instructor:** Ning Rui

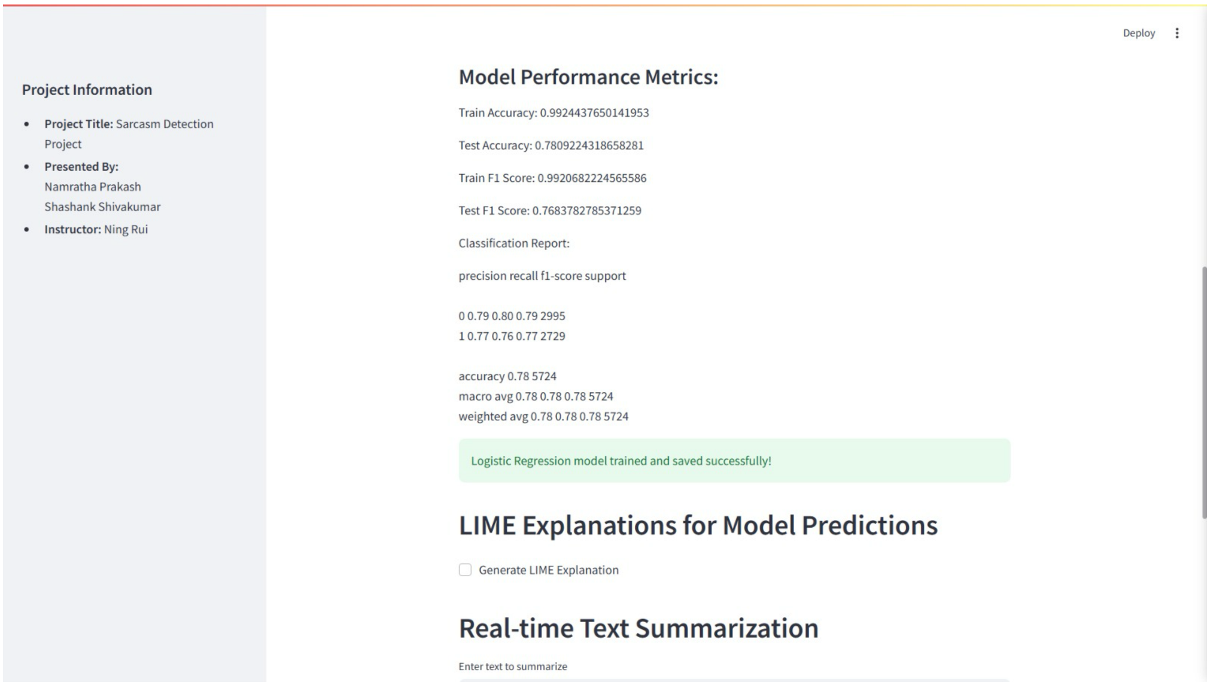
LIME Explanations for Model Predictions

☑ Generate LIME Explanation

Example 24106

Headline: kuwait may owe as much as $60,000 for trump hotel event in d.c.

True Class: Non Sarcastic



Project Information

- **Project Title:** Sarcasm Detection Project
- **Presented By:**
  Namratha Prakash
  Shashank Shivakumar
- **Instructor:** Ning Rui

Model Performance Metrics:

Train Accuracy: 0.9924437650141953

Test Accuracy: 0.7809224318658281

Train F1 Score: 0.9920682224565586

Test F1 Score: 0.7683782785371259

Classification Report:

```
precision recall f1-score support

0 0.79 0.80 0.79 2995
1 0.77 0.76 0.77 2729

accuracy 0.78 5724
macro avg 0.78 0.78 0.78 5724
weighted avg 0.78 0.78 0.78 5724
```

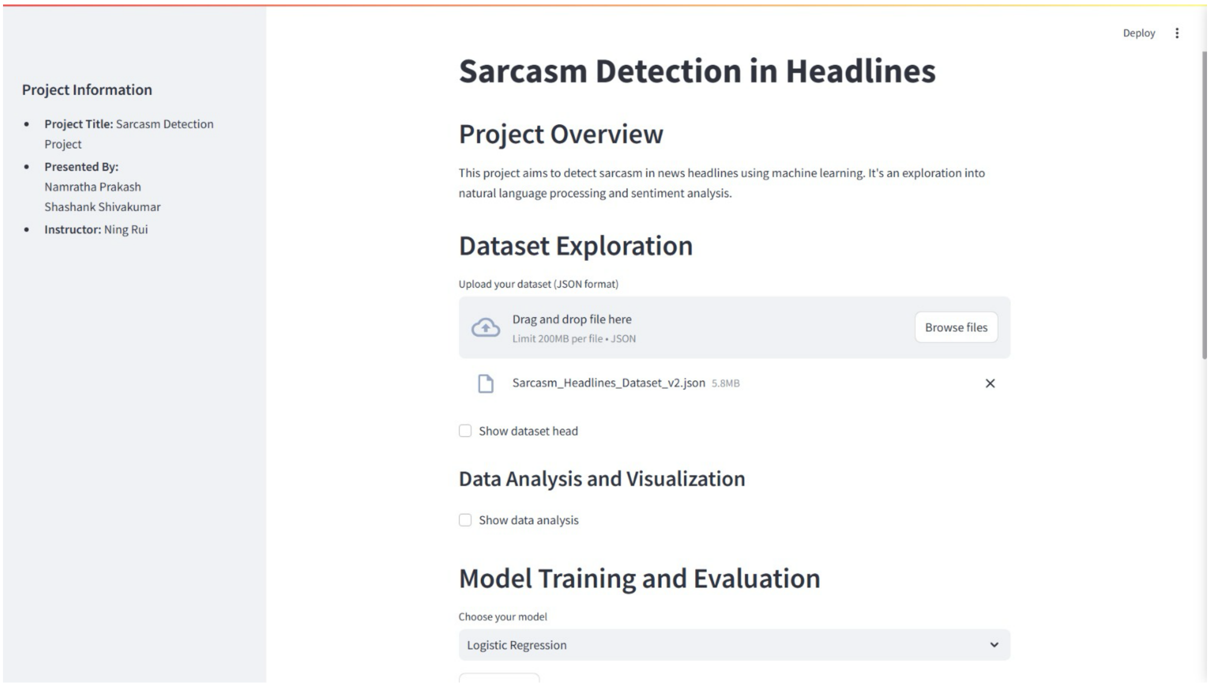
Logistic Regression model trained and saved successfully!

LIME Explanations for Model Predictions

□ Generate LIME Explanation

Real-time Text Summarization

Enter text to summarize



Project Information

- **Project Title:** Sarcasm Detection Project
- **Presented By:**
  Namratha Prakash
  Shashank Shivakumar
- **Instructor:** Ning Rui

Sarcasm Detection in Headlines

Project Overview

This project aims to detect sarcasm in news headlines using machine learning. It's an exploration into natural language processing and sentiment analysis.

Dataset Exploration

Upload your dataset (JSON format)

Drag and drop file here
Limit 200MB per file • JSON

Sarcasm_Headlines_Dataset_v2.json 5.8MB

□ Show dataset head

Data Analysis and Visualization

□ Show data analysis

Model Training and Evaluation

Choose your model

Logistic Regression

9. Reference

1. https://github.com/ning-rui/FA24_DATS6312_11
2. https://www.kaggle.com/code/quadeer15sh/transformers-for-text-classification
3. https://neptune.ai/blog/how-to-code-bert-using-pytorch-tutorial
4. https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-onhyperparameter-tuning-and-its-techniques/

5. https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime5adc035109b5

6. https://streamlit.io/gallery

7. https://realpython.com/python-web-scraping-practicalintroduction/#:~:text=One%20useful%20package%20for%20web,a%20URL%20within %20a%20program