
**THE GEORGE
WASHINGTON
UNIVERSITY**

WASHINGTON, DC

Columbian College of Arts & Sciences

MS in Data Science

Sarcasm Detection Project

Shashank Shivakumar

G30631923

Instructor: Prof. Ning Rui

Date: 04th Dec 2024

Table of Content

1. Introduction	3
2. Dataset Description	4
3. Description of Individual Work	5
3.1 Proposed System	5
3.2 Proposed Algorithms	7
Rule-based Algorithms	7
3.2.1 Naive Bayes	7
3.2.2. Logistic Regression	8
Pretrained NLP (Transformers Based Networks)	9
3.2.3 BERT and RoBERTa	9
3.3 Interpretability and Explainability	9
3.3.1 LIME	9
3.4 Text Summarization	10
3.5 Evaluation metrics	10
3.6 SHAP	10
3.7 Streamlit	11
4. Results	12
4.1 Model Performance	12
4.2 Model Explainability	14
4.3 LIME Explanation Examples	16
4.4 Text Summarization Performance	20
4.5 Model Optimization: Hyperparameter Tuning and Overfitting Prevention Strategies	22
5. Summary	23
6. Conclusion	24
7. Future Scope	24
8. Percentage of code	24
9. References	24

1. Introduction

The challenge of detecting sarcasm in textual content is a fascinating and complex problem in the realm of Natural Language Processing (NLP).

- **Contextual Understanding and Misinterpretation:** Sarcasm poses a significant challenge in news headlines due to its nuanced and context-dependent nature. The subtle interplay of language in sarcastic headlines can lead to misinterpretations, potentially causing readers to misunderstand the intended message. Detecting sarcasm becomes crucial to ensure accurate comprehension and prevent the spread of misinformation or unintended messages in news reporting.
- **Maintaining Credibility and Trustworthiness:** Sarcasm, when undetected, can undermine the credibility and trustworthiness of news sources. Readers may perceive sarcastic statements as factual, leading to misinformation and potentially influencing public opinion. The need for sarcasm detection in news headlines stems from a broader concern for maintaining the integrity of journalistic content, ensuring that headlines accurately reflect the intended tone and meaning.
- **Mitigating Negative Social Impact:** Sarcasm in news headlines, if not identified, can contribute to the dissemination of biased or inflammatory information. Incorrectly interpreting sarcastic content may lead to unwarranted emotional reactions, social polarization, or even the propagation of false narratives. Detecting sarcasm in news headlines is, therefore, essential for mitigating negative social impact and fostering a more informed and discerning readership.

Our project aims to tackle this issue by focusing on sarcasm detection in news headlines. Sarcasm detection is crucial because it helps in understanding the underlying sentiment and tone in written language, which can significantly differ from the literal meaning of the words used. In addition to sarcasm detection, our project explores the application of text summarization techniques to distill key information from sarcastic news articles. This comprehensive analysis involves utilizing various NLP techniques and models to distinguish between sarcastic and non-sarcastic news headlines, providing a holistic approach to understanding and processing sarcastic textual content.

2. Dataset Description

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 link to the original news article.
- **Data Enrichment:** To add depth to our analysis, we have extended the dataset by scraping the main content of the articles using the BeautifulSoup Python package. This allows us to compare the headlines with their corresponding articles, providing a more nuanced understanding of sarcasm in news media. This enhanced dataset is hosted in the cloud, facilitating efficient access and analysis.
- **Web Scraping:** We also scraped the URLs available in our dataset where is_sarcastic has value 1. The actual news text was saved and used for Text Summarization.

3. Description of Individual Work

3.1 Proposed System

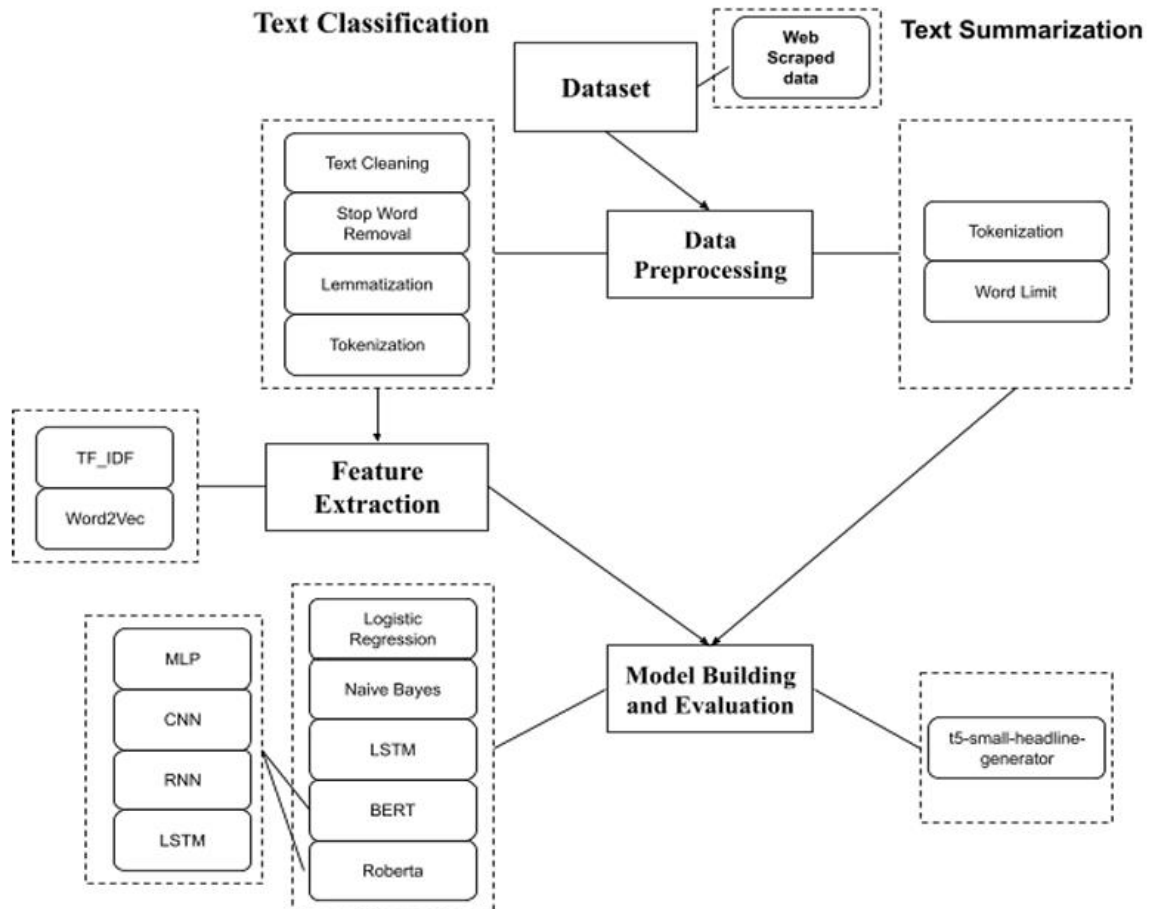


Figure 1: Proposed System

- **Data Preprocessing:** The initial phase involves cleansing the dataset by removing stop words, applying lemmatization to reduce words to their root form, and tokenizing the text, which breaks it down into individual units for analysis.
- **Feature Extraction:** We leverage both traditional and advanced methods to extract features from the text. Techniques such as TF-IDF (Term Frequency-Inverse Document Frequency) and Word2Vec are used to transform text into a format that can be easily interpreted by our models. These features serve as the input for the subsequent modeling phase.
- **Model Building and Evaluation:** Our approach explores a variety of models to determine the most effective for our objectives. For text classification, we experiment with models ranging

from Logistic Regression and Naive Bayes to more sophisticated neural network architectures like LSTM (Long Short-Term Memory) and transformer-based models such as BERT and Roberta.

Throughout the project, we continuously refine our models and evaluate their performance to ensure high accuracy and reliability in classifying and summarizing text, which could be pivotal for applications in content analysis, sentiment detection, and information retrieval.

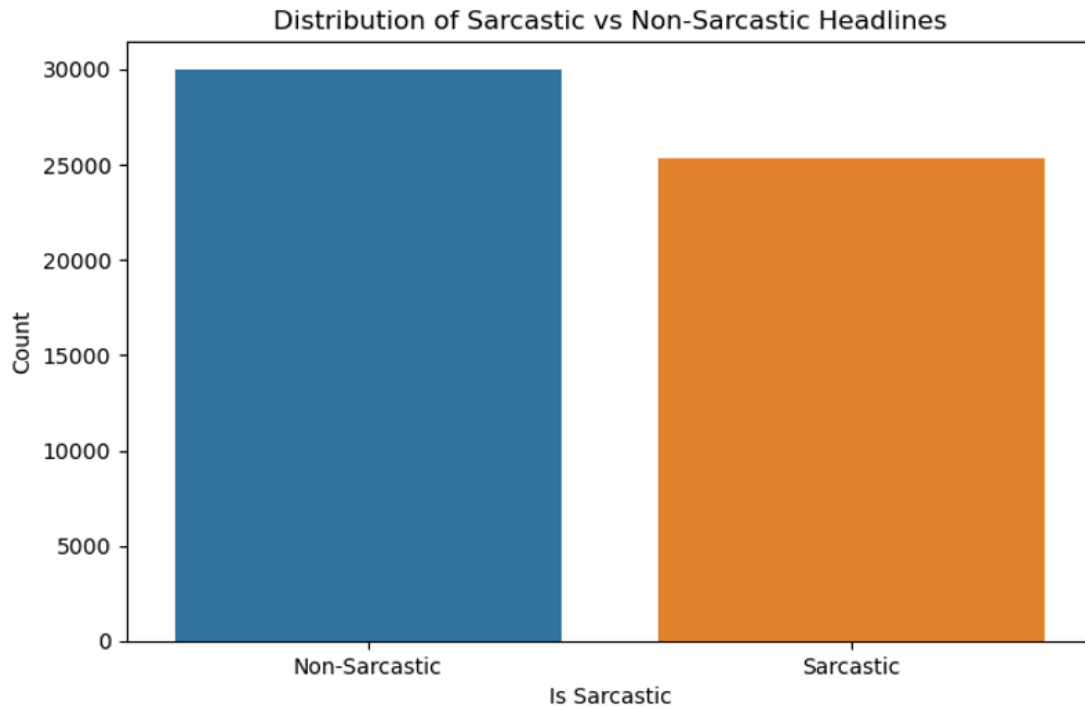


Figure 2: Histogram for Distribution of Sarcastic Vs Non-Sarcastic Headlines

In the implementation, a series of text-processing techniques were applied to enhance the textual data for downstream natural language processing tasks. The process began with tokenization to break down the text into individual units, followed by the removal of stop words, common words that add little semantic meaning, thereby reducing noise in the dataset. Subsequently, all text was converted to lowercase to ensure uniformity and simplify subsequent analyses. Lemmatization was then employed to normalize words, reducing them to their base forms, thereby aiding in capturing the root meaning of words. Following these preprocessing steps, TFIDF (Term Frequency-Inverse Document Frequency) calculations were performed, quantifying the importance of each word in a document relative to a larger corpus. Alternatively, the Word2Vec word embedding technique was implemented, generating continuous vector representations for words in a high-dimensional space, and capturing semantic relationships. This comprehensive text processing pipeline prepared the textual data for more effective utilization in natural language understanding and related applications.

3.2 Proposed Algorithms

1. Naive Bayes
2. Logistic Regression
3. BERT
4. RoBERTa

Rule-based Algorithms

3.2.1 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:

$$P(c|x) = \frac{P(x) \times P(c)}{P(x|c) \times P(c)}$$

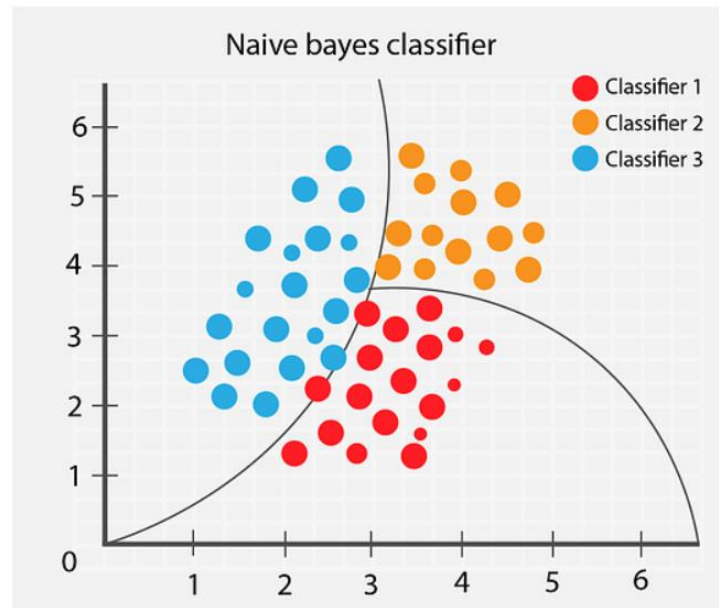


Figure 3: Naïve Bayes Classifier

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.

3.2.2. 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:

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

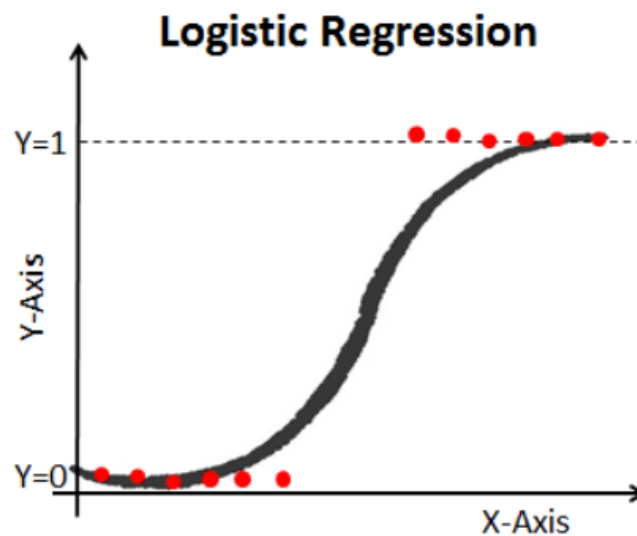


Figure 4: Logistic Regression

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.

Pretrained NLP (Transformers Based Networks)

3.2.3 BERT and RoBERTa

Background: BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (a robustly optimized BERT approach) are 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.

3.3 Interpretability and Explainability

3.3.1 LIME

Background: LIME (Local Interpretable Model-agnostic Explanations) is a technique for explaining the predictions of any machine learning classifier. It works by approximating the model locally using an interpretable model.

Role in Sarcasm Detection:

LIME enhances the understanding of sarcasm detection models by elucidating why certain predictions were made. It breaks down the model's decision-making process, highlighting the specific text features that most influenced the classification, thereby offering transparency in sarcasm detection.

3.4 Text Summarization

To ensure effective training, the dataset undergoes preprocessing first by using tokenization of the t5-small model. Imposing a word limit on input text for text summarization models is a pragmatic approach to address computational constraints and enhance the coherence of generated summaries. Thus, articles are filtered based on a word limit of 30 words to manage the complexity of the summarization task.

T5 is an encoder-decoder model pre-trained on a multi-task mixture of unsupervised and supervised tasks for which each task is converted into a text-to-text format. T5 works well on a variety of tasks out-of-the-box by prepending a different prefix to the input corresponding to each task. T5-small is a variant of the T5 model, specifically representing a smaller and computationally lighter version. Despite its reduced size, T5-small retains the core architecture and capabilities of T5, making it suitable for applications with limited computational resources like this.

3.5 Evaluation metrics

For model evaluation, various metrics were strategically employed to assess the performance and characteristics of the trained summarization model. Train accuracy and test accuracy provided insights into the model's ability to correctly predict headlines during both the training and evaluation phases, respectively. The number of epochs and batch size were crucial parameters monitored during training, offering a glimpse into the model's convergence and computational efficiency. Train loss, a measure of the model's error during training, served as an indicator of its learning progress over time. Learning rate, a key hyperparameter, was optimized to regulate the magnitude of updates during training. The choice of the optimizer, in this case, AdamW or Adam, influenced the optimization process. These metrics collectively formed a comprehensive evaluation framework, enabling a thorough understanding of the model's training dynamics, generalization performance, and parameter tuning effectiveness. Additionally, for text summarization, the ROUGE metric is employed to assess the effectiveness of the baseline on the validation dataset. ROUGE scores provide insights into the quality of the generated summaries concerning the ground truth. ROUGE-1 measures unigram overlap, ROUGE-2 evaluates bigram overlap, ROUGE-L assesses the longest common subsequence, and ROUGE-Lsum considers unigrams, bigrams, and longest common subsequence in automatic text summarization evaluation.

3.6 SHAP

SHAP (SHapley Additive exPlanations) is a model-agnostic framework for interpreting the output of machine learning models. It assigns fair contributions to individual input features, allowing for a nuanced understanding of a model's decision process. By providing feature-level insights, SHAP values help quantify the impact of each feature on model predictions, facilitating enhanced interpretability. This method is applicable across various model architectures, making it versatile

for both simple and complex models. Importantly, SHAP values aid in addressing the "black-box" nature of certain models, offering transparency and explanations for specific predictions. The framework is widely used to gain insights into feature importance and enhance trust in machine learning models.

3.7 Streamlit

In our project, we developed a Streamlit application focused on detecting sarcasm in news headlines, leveraging machine learning and natural language processing. The application encompasses several integral features: This introductory section outlines the application's objective to analyze and identify sarcasm in news headlines. Users have the flexibility to upload datasets in JSON format. The application includes real-time text summarization and sarcasm prediction, allowing users to input text for immediate analysis and results, showcasing the application's practical application in processing and understanding natural language data.

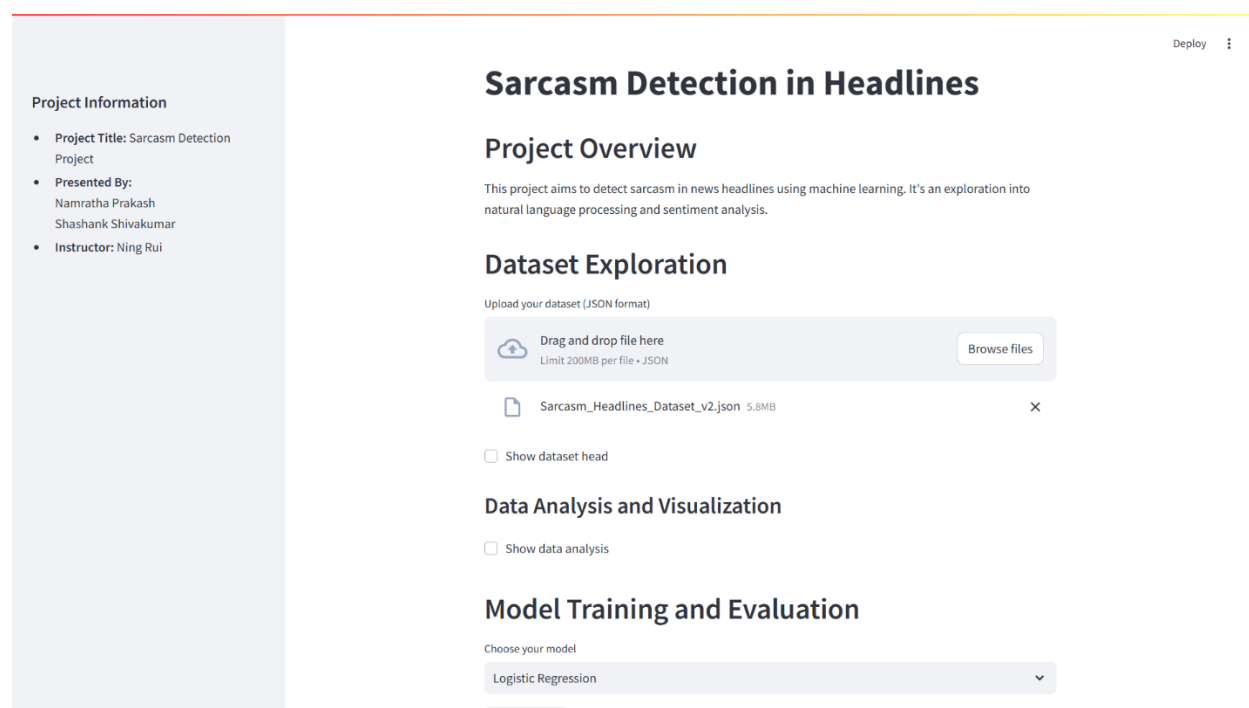


Figure 5: Streamlit App

4. Results

The dataset is split into training, validation, and test sets in 80%, 10%, and 10% splits respectively. Subsequently, the models were trained on the training data and evaluated on validation data.

4.1 Model Performance

BERT+RNN

```
/usr/bin/python3 /home/ubuntu/NLP/mywork/Project/Bertclassifictn.py
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 1/3
Training: 100%|██████████| 835/835 [02:32<00:00, 5.47it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.40it/s]
Train Loss: 0.263, Train Acc: 0.892
Val Loss: 0.131, Val Acc: 0.956
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 2/3
Training: 100%|██████████| 835/835 [02:33<00:00, 5.46it/s]
Evaluating: 100%|██████████| 835/835 [00:51<00:00, 16.37it/s]
Train Loss: 0.115, Train Acc: 0.961
Val Loss: 0.058, Val Acc: 0.983
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 3/3
Training: 100%|██████████| 835/835 [03:39<00:00, 3.80it/s]
Evaluating: 100%|██████████| 835/835 [01:52<00:00, 7.45it/s]
Train Loss: 0.063, Train Acc: 0.980
Val Loss: 0.027, Val Acc: 0.993
Training complete!
```

- The training output indicates that the BERT+RNN model improved consistently across all 3 epochs.
- Training and validation accuracy increased from 89.2% to 98.0% and 95.6% to 99.3%, respectively, while both training and validation losses decreased significantly, reflecting a successful learning process without apparent overfitting.
- The training duration and speed variations suggest changes in computational load, which is normal during such processes.

BERT+MLP

```
Epoch 1/3
Training: 100%|██████████| 835/835 [02:30<00:00, 5.56it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.52it/s]
Train Loss: 0.276, Train Acc: 0.890
Val Loss: 0.098, Val Acc: 0.970
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 2/3
Training: 100%|██████████| 835/835 [02:30<00:00, 5.56it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.42it/s]
Train Loss: 0.112, Train Acc: 0.962
Val Loss: 0.036, Val Acc: 0.994
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 3/3
Training: 100%|██████████| 835/835 [02:30<00:00, 5.56it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.42it/s]
Train Loss: 0.048, Train Acc: 0.986
Val Loss: 0.019, Val Acc: 0.996
Training complete!
```

- The BERT+MLP model displayed remarkable learning efficiency over three epochs, with both training and validation accuracies improving to 98.6% and 99.6% respectively, accompanied by substantial reductions in training and validation losses.
- These outcomes suggest the model's high proficiency in the task with consistent performance gains, indicating a well-tuned model with a low likelihood of overfitting.

BERT+LSTM

```
/usr/bin/python3 /home/ubuntu/NLP/mywork/Project/Bertclassifictn.py
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 1/3
Training: 100%|██████████| 835/835 [02:32<00:00, 5.46it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.41it/s]
Train Loss: 0.266, Train Acc: 0.893
Val Loss: 0.101, Val Acc: 0.969
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 2/3
Training: 100%|██████████| 835/835 [02:33<00:00, 5.45it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.39it/s]
Train Loss: 0.116, Train Acc: 0.961
Val Loss: 0.055, Val Acc: 0.987
Training: 0%|          | 0/835 [00:00<?, ?it/s]Epoch 3/3
Training: 100%|██████████| 835/835 [02:33<00:00, 5.45it/s]
Evaluating: 100%|██████████| 835/835 [00:50<00:00, 16.39it/s]
Train Loss: 0.062, Train Acc: 0.981
Val Loss: 0.028, Val Acc: 0.992
Training complete!
```

- The BERT+LSTM model training output indicates progressive learning across three epochs.
- The training accuracy started at 89.3% and improved to 98.1%, while the validation accuracy increased from 96.9% to 99.2%.
- There was a significant decrease in both training and validation losses, indicating effective learning and model generalization without overfitting, as evidenced by the high validation accuracy.

RoBERTa

```
Training Epoch 1: 100%|██████████| 835/835 [09:13<00:00, 1.51it/s]
Evaluating: 100%|██████████| 895/895 [04:30<00:00, 3.30it/s]
Epoch 1: Train Loss: 0.580, Val Loss: 0.512, Val Accuracy: 0.749
Training Epoch 2: 100%|██████████| 835/835 [04:39<00:00, 2.98it/s]
Evaluating: 100%|██████████| 895/895 [04:15<00:00, 3.50it/s]
Epoch 2: Train Loss: 0.504, Val Loss: 0.502, Val Accuracy: 0.757
Training Epoch 3: 100%|██████████| 835/835 [04:39<00:00, 2.98it/s]
Evaluating: 100%|██████████| 895/895 [04:15<00:00, 3.50it/s]
Epoch 3: Train Loss: 0.490, Val Loss: 0.499, Val Accuracy: 0.759
Training complete!
```

- The training log for the RoBERTa model indicates modest improvement over three epochs. Starting with a validation accuracy of 74.9%, it improved slightly to 75.9% by the third epoch.
- The training and validation losses saw marginal decreases, suggesting a slow but steady learning process.
- The relatively small increase in validation accuracy and high validation loss may suggest the need for further optimization or a more complex model to capture the nuances of the underlying task.

4.2 Model Explainability

Logistic Regression Model:

LR Train Accuracy: 0.9987799918666125

LR Train F1-score: 0.9986691640378548

LR Test Accuracy: 0.9412615217784204

LR Test F1-score: 0.935515873015873

Classification Report:

	precision	recall	f1-score	support
0	0.94	0.95	0.95	5994
1	0.94	0.93	0.94	5072
accuracy			0.94	11066
macro avg	0.94	0.94	0.94	11066
weighted avg	0.94	0.94	0.94	11066

Logistic Regression - Example 9732

Headline: sir mix-a-lot wasn't trying to speak for women with 'baby got back'

Probability (Non sarcastic) = 9.142584158135015e-08

Probability (sarcastic) = 0.9999999085741584

True Class: Non Sarcastic

Naive Bayes - Example 9732

Headline: sir mix-a-lot wasn't trying to speak for women with 'baby got back'

Probability (Non sarcastic) = 0.29106486024040906

Probability (sarcastic) = 0.7089351397595908

True Class: Non Sarcastic

Logistic Regression - Example 1805

Headline: 13 year old boy diagnosed with incurable puberty

Probability (Non sarcastic) = 0.999999991470019

Probability (sarcastic) = 8.529981165850131e-10

True Class: Sarcastic

Naive Bayes - Example 1805

Headline: 13 year old boy diagnosed with incurable puberty

Probability (Non sarcastic) = 0.48315871654862524

Probability (sarcastic) = 0.5168412834513747

True Class: Sarcastic

Logistic Regression - Example 7594

Headline: palmolive attacks dawn for coddling grease

Probability (Non sarcastic) = 0.8724596531193708

Probability (sarcastic) = 0.1275403468806292

True Class: Sarcastic

Naive Bayes

NB Train Accuracy: 0.9101486602503276

NB Train F1-score: 0.8985174411186813

NB Test Accuracy: 0.863545996746792

NB Test F1-score: 0.8449055053410025

Classification Report:

	precision	recall	f1-score	support
0	0.85	0.91	0.88	5994
1	0.88	0.81	0.84	5072
accuracy			0.86	11066
macro avg	0.87	0.86	0.86	11066
weighted avg	0.86	0.86	0.86	11066

Naive Bayes - Example 7594

Headline: palmolive attacks dawn for coddling grease

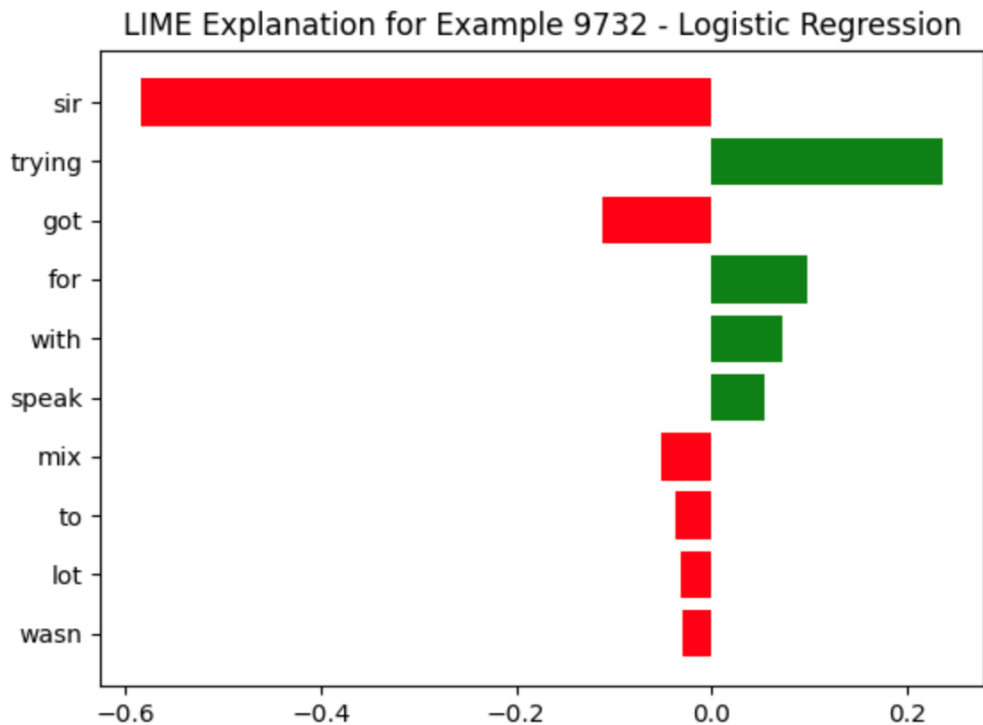
Probability (Non sarcastic) = 0.35799268871363604

Probability (sarcastic) = 0.6420073112863656

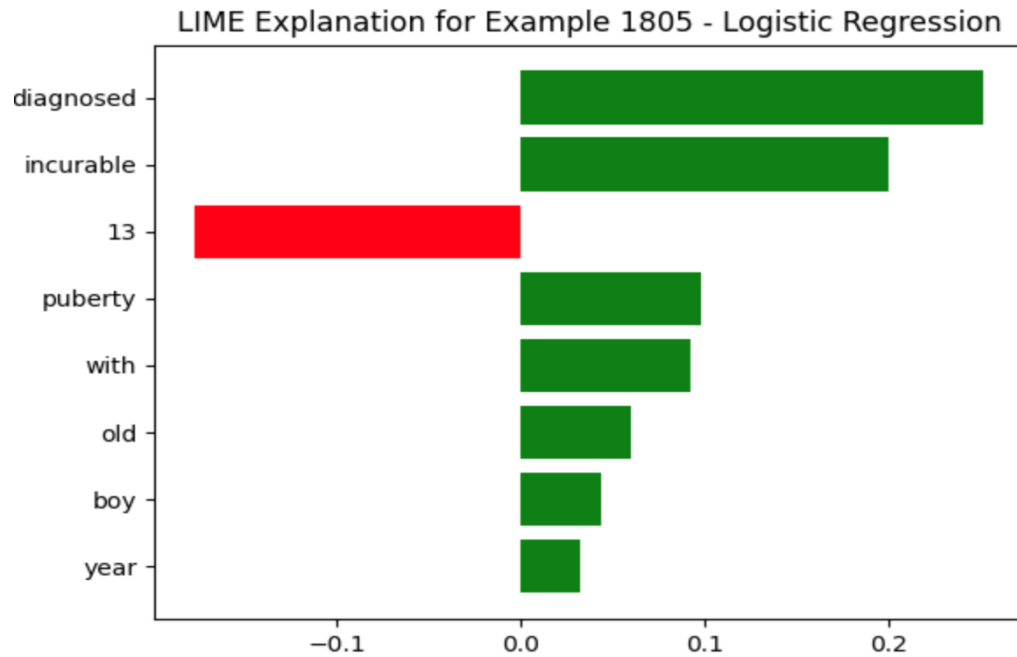
True Class: Sarcastic

- Logistic Regression shows excellent training performance with perfect accuracy and F1-score, which drops slightly on the test set but remains high with an accuracy of 95.05% and an F1-score of 94.54%. The classification report reflects a balanced precision-recall trade-off for both classes.
- Naive Bayes, while decent in training, shows a notable decrease in test performance, with an accuracy of 85.89% and an F1-score of 83.99%. The classification report indicates a higher precision for class 1, but lower recall compared to class 0, suggesting a bias towards false negatives for the sarcastic class.
- Specific examples reveal that Logistic Regression tends to have higher confidence in its predictions, closely aligning with the true class, while Naive Bayes exhibits less certainty, as evidenced by the closer probability scores between classes.

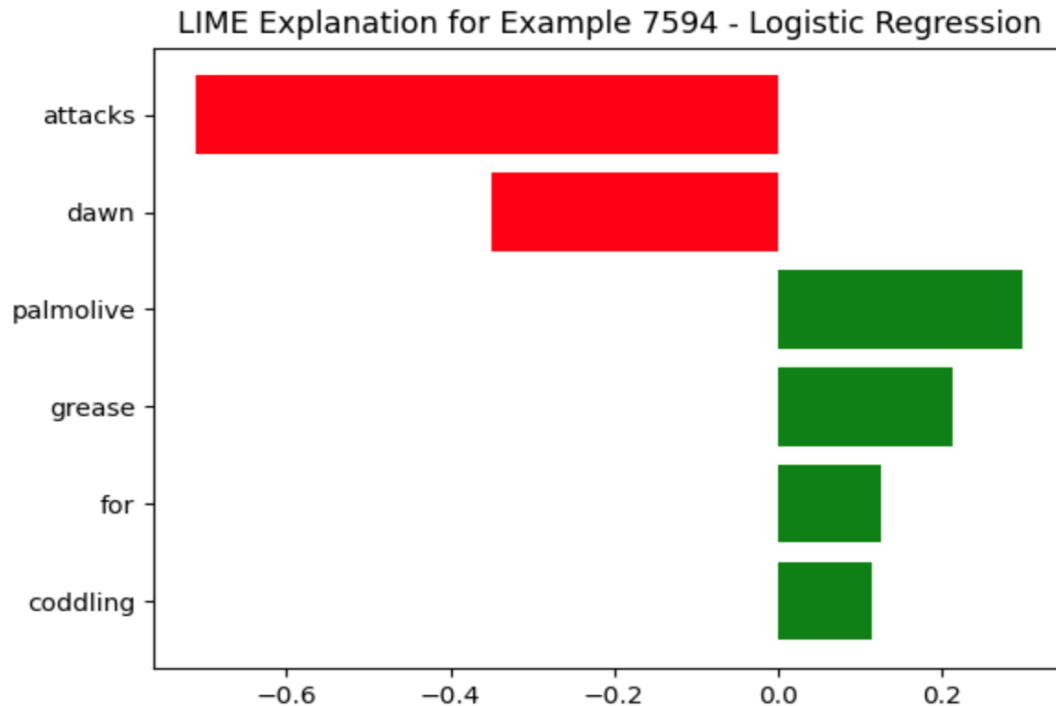
4.3 LIME Explanation Examples



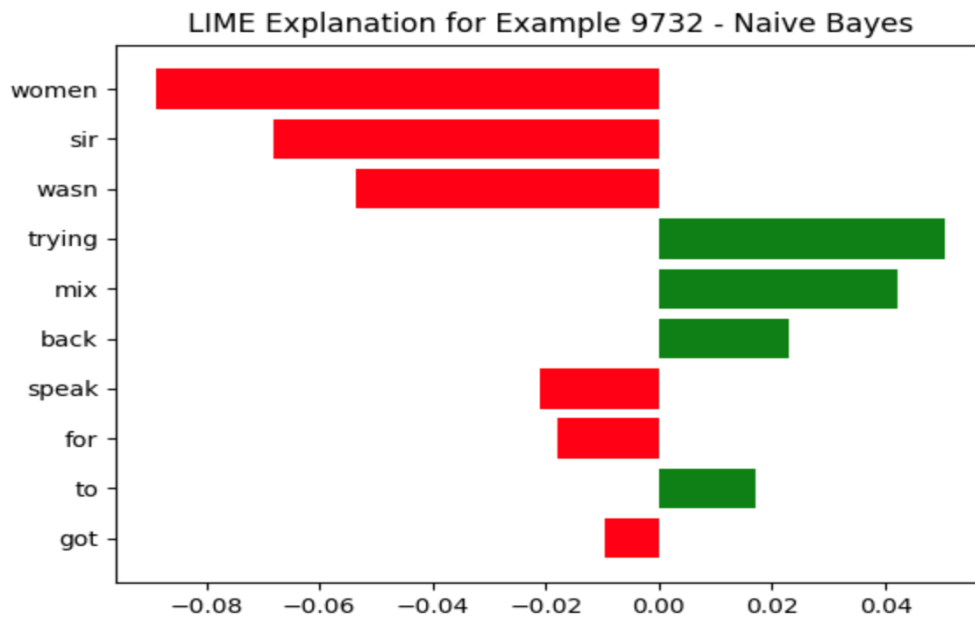
- The LIME explanation graph for a Logistic Regression model shows the weight of each feature in the decision-making process for a specific example.
- Red bars indicate features that contribute to a negative class prediction, while green bars represent features supporting a positive class prediction.
- The length of each bar reflects the strength of each feature's contribution.



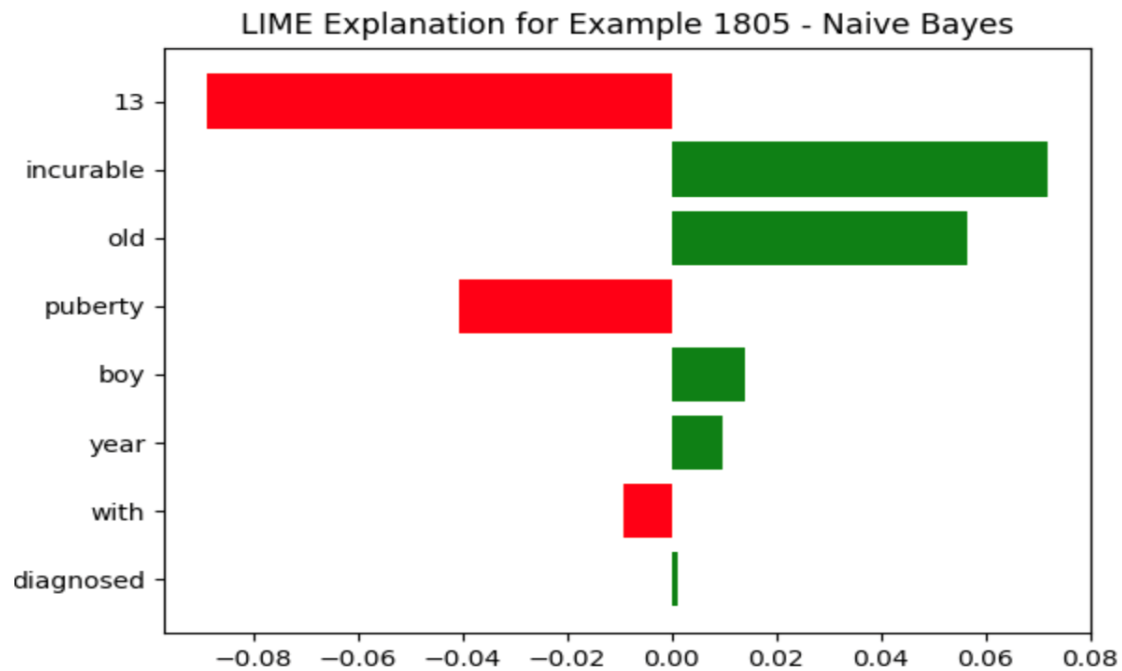
- The LIME graph for Logistic Regression presents the feature impact for example 1805. Here, most features positively influence the model's prediction, represented by green bars.
- The feature '13' negatively influences the prediction, depicted by the red bar.
- The size of the bars indicates the strength of each feature's influence.



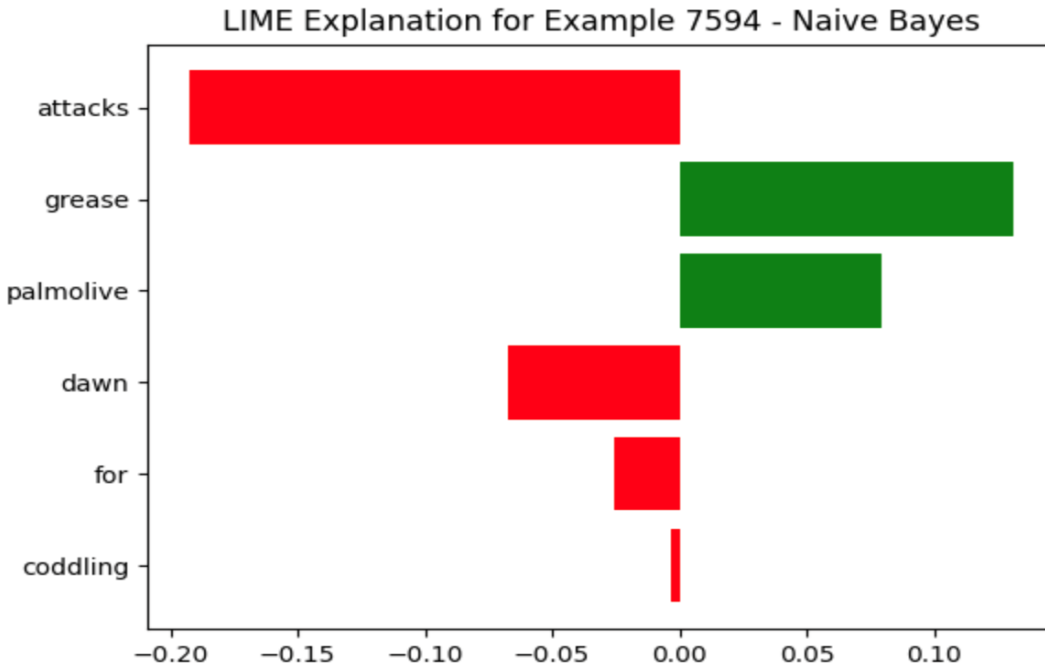
- The LIME graph for Logistic Regression illustrates the positive and negative contributions of various features for Example 7594.
- The feature 'attacks' has the most significant negative impact on the prediction, while 'palmolive' and 'grease' positively contribute, with 'palmolive' having the largest positive effect.
- This visual explanation helps understand the model's reasoning for this instance.



- The LIME explanation for the Naive Bayes model shows how each feature influences the prediction for a specific example.
- Features represented by red bars negatively impact the predicted class, while those in green positively affect it.
- The magnitude of these bars indicates the strength of the influence each feature has on the model's prediction.



- In the LIME explanation for the Naive Bayes model, features like 'old' and '13' negatively impact the prediction, as shown by the red bars.
- In contrast, features such as 'puberty' and 'year' have a positive influence, indicated by green bars.
- The size of each bar represents the feature's impact strength on the model's prediction for example 1805.



- The LIME visualization for the Naive Bayes model on Example 7594 demonstrates how individual features sway the prediction.
- The feature 'attacks' has a strong negative influence, while 'grease' contributes positively.
- Features 'palmolive' and 'dawn' also impact the model's prediction, with 'palmolive' showing negative and 'dawn' a lesser negative influence, contrasting their impact in the Logistic Regression model.

4.4 Text Summarization Performance

The dataset is tokenized using the T5 (Text-to-Text Transfer Transformer) model tokenizer, a state-of-the-art model for sequence-to-sequence tasks. The model used for headline generation is a fine-tuned version of t5-small. Specifically, it is based on the JulesBelveze/t5-small-headline-generator model, which is pre-trained for headline generation using the tldr_news dataset.

The T5 model is fine-tuned on sarcastic news articles using the AdamW optimizer with a specified learning rate. The training process involves multiple epochs, and a linear learning rate schedule is employed to gradually adjust the learning rate during training. The performance of the trained model is evaluated on the validation set using the ROUGE metric. ROUGE scores are tracked over epochs to observe the model's convergence and identify potential areas for improvement.

Baseline score on validation dataset is:

rouge1 : 0.30535865583734484

rouge2 : 0.15140735912807318

rougeL : 0.28014930371528557

rougeLsum : 0.27992954959101823

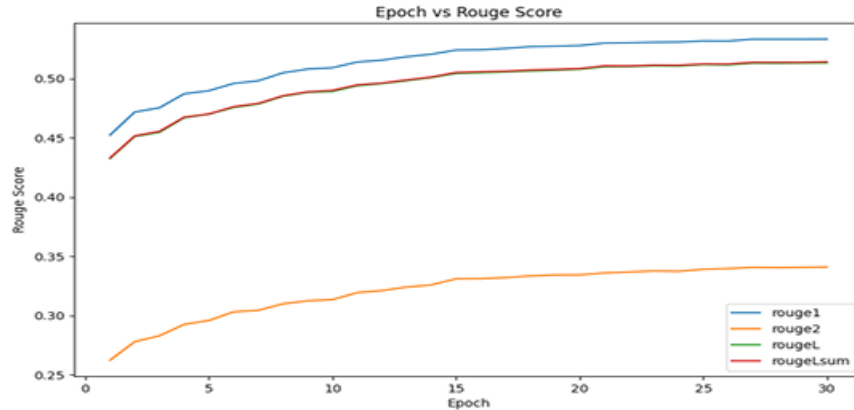


Figure 6: Epoch vs Rouge Score for Summarization model

Here we can see that the ROUGE scores improve as the model is trained over the epochs. The ROUGEScores around 0.50 are considered to be good. Thus, we can say that our text summarization model is appropriately trained. Here we can see that the summary created by the model is pretty close to what the actual headline is.

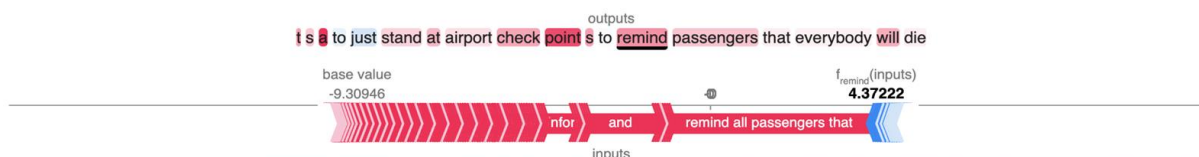
```
'>>> Article: ARLINGTON, VA- Following the release of a report indicating that the agency failed 95 percent of security tests, the Transportation Security ,
Administration announced Tuesday that agents will now simply stand at airport checkpoints and remind all passengers that everybody will eventually die someday. "As ,
part of our new security protocol, TSA agents at every checkpoint will carefully inform each passenger that life is a temporary state and that no man can escape the ,
fate that awaits us all," said acting TSA administrator Mark Hatfield, adding that under the new guidelines, agents will ensure that passengers fully understand ,
and accept the inevitability of death as they proceed through the boarding pass check, luggage screening, and body scanner machines. "Signs posted throughout the ,
queues will also state that death is unpredictable but guaranteed, and a series of looping PA messages will reiterate to passengers that, even if they survive this ,
flight, they could still easily die in 10 years or even tomorrow." Hatfield went on to say that the TSA plans to add a precheck program that will expedite the ,
process for passengers the agency deems comfortable with the ephemeral nature of life.'
```

```
'>>> Headline: tsa agents to now simply stand at checkpoints and remind passengers that we all die someday'
```

```
'>>> Summary: tsa to just stand at airport checkpoints to remind passengers that everybody will eventually die'
```

Figure 7: Sample Summarization using model

Understanding how the model summarized the text is a very complex task. We tried to use SHAP to interpret the summarized text and how the model interpreted the words.



Following the release of aa report indicating that the agency failed 95 percent of security tests, the Transportation Security Administration announced Tuesday that agents will now simply stand at airport checkpoints and remind all passengers that everybody will eventually die someday. "As part of our new security protocol, TSA agents at every checkpoint will carefully inform each passenger that life is aa temporary state and that no man can escape the fate that awaits us all," said acting TSA administrator Mark Hatfield, adding that under the new guidelines, agents will ensure that passengers fully understand and accept the inevitability of death as they proceed through the boarding pass check, luggage screening, and body scanner machines. "Signs posted throughout the queues will also state that death is unpredictable but guaranteed, and aa series of looping PA messages will reiterate to passengers that, even if they survive this flight, they could still easily die in 10 years or even tomorrow." Hatfield went on to say that the TSA plans to add aa precheck program that will expedite the process for passengers the agency deems comfortable with the ephemeral nature of life.

Here, we can see that the word with higher red intensity indicates that this token significantly increased the model's prediction for the specific summary. However, there are some words with blue intensity indicating that these tokens slightly decreased the model's prediction for the specific summary. This suggests that the model identified "airport", "checkpoint", and "remind" as key words for summarizing the article.

4.5 Model Optimization: Hyperparameter Tuning and Overfitting Prevention Strategies

In our sarcasm detection model, we conducted a thorough hyperparameter search focusing on several key parameters. The ones that significantly influenced our model's performance included learning rate, batch size, the number of training epochs, and the maximum sequence length for input text.

- **Learning Rate:** We experimented with rates ranging from $1e-6$ to $5e-5$, observing that smaller rates often led to better generalization, particularly in complex models like BERT combined with MLP.
- **Batch Size:** A range from 16 to 32 was tested to find the sweet spot for our model's memory constraints and learning stability.
- **Epochs:** We limited the number of epochs to a range of 3 to 30, depending on the model complexity, to balance between underfitting and overfitting.
- **Max Length:** The maximum sequence length varied between 120 to 256 tokens, which helped in managing the context window the models could effectively utilize.

To detect and prevent overfitting, we implemented several strategies:

- **Validation Set:** Utilized a separate validation dataset to monitor the model's performance and prevent it from learning noise in the training data.
- **Early Stopping:** Incorporated early stopping in our training process to halt the training when the validation loss stopped improving, thus avoiding overfitting.
- **Regularization:** Applied techniques like dropout in neural network architectures to reduce overfitting by preventing complex co-adaptations on training data.

5. Summary

Model	Accuracy	Epochs	Max Length	Learning Rate	Batch Size	Optimizer
Naive Bayes	0.86					
Logistic Regression	0.94					
BERT + LSTM	0.98	3	128	2e-6	32	AdamW
BERT+MLP	0.98	3	128	5e-5	32	AdamW
RoBERTa	0.76	10	256	1e-5	32	Adam

Table 1: Models Summary

Model	Accuracy	Epochs	Max Length	Learning Rate	Batch Size	Optimizer
t5-small-headline-generator	0.55 (ROUGE)	30	128	2e-5	16	AdamW

Table 2: Summarization Model Summary

6. Conclusion

- **Highest Accuracy:** Our best-performing models were BERT + MLP and BERT + LSTM, each achieving an impressive 0.98 accuracy. This suggests that the combination of BERT's contextual understanding with the nuanced pattern recognition of MLP and LSTM networks is highly effective for sarcasm detection.
- **Transformer Models:** Standalone transformer models like BERT and RoBERTa showcased strong potential, with BERT achieving 0.91 accuracy. However, RoBERTa lagged at 0.76, indicating that model architecture and parameter tuning are critical for optimal performance. The integration of BERT with other neural network architectures has proven to be highly effective for sarcasm detection, outperforming traditional models and even other advanced neural network-based classifiers.
- **Text Summarization:** The trained t5-small-headline-generator scored 0.55 in ROUGE, which is a sign of good summarization model. This indicates room for improvement in the model's ability to generate sarcasm within summaries.

7. Future Scope

- **Model Optimization:** Further refinement of hyperparameters for transformer models, especially RoBERTa, to enhance their sarcasm detection accuracy.
- **Dataset Expansion:** Incorporation of more diverse datasets, including multilingual headlines, to improve the model's robustness and applicability across different languages and cultural contexts.
- **Summarization Improvement:** Development of a more sophisticated approach for the t5-small-headline-generator to improve its performance in generating contextually relevant sarcastic summaries.

8. Percentage of code

According to the formula, the code percentage is **47%**

9. References

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://huggingface.co/docs/transformers/model_doc/roberta

5. <https://www.analyticsvidhya.com/blog/2022/02/a-comprehensive-guide-on-hyperparameter-tuning-and-its-techniques/>
6. <https://towardsdatascience.com/decrypting-your-machine-learning-model-using-lime-5adc035109b5>
7. <https://streamlit.io/gallery>
8. <https://realpython.com/python-web-scraping-practical-introduction/#:~:text=One%20useful%20package%20for%20web,a%20URL%20within%20a%20program>