Advance NLP: Hate Speech detection using Transformers (Deep Learning) - Group Project

by

Team member one:	Team member two:
Michael Udonna Egbuzobi	**Nweke Nonye**
egbuzobi.michael@gmail.com	nonyenweke22@gmail.com
United Kingdom	United Kingdom
University of Wolverhampton	University of Wolverhampton
Data Science	Data Science

Data Glacier Internship Project

Batch LISUM36: 30 July – 30 Oct 24

30/10/2024

Michael & Nonye

TABLE OF CONTENTS

TABLE OF FIGURES

1 Project Overview

The project focuses on developing an advanced hate speech detection model using transformer-based deep learning architectures. Given the rise of online communication, particularly on social media platforms like Twitter, the prevalence of hate speech poses significant risks to social harmony and public discourse. This project aims to create an effective classification model to automatically identify and categorize tweets as hate speech or non-hate speech, contributing to safer online interactions and enabling platforms to take proactive measures against harmful content.

1.2 Problem Description

Hate speech is a form of communication that employs derogatory language to attack or discriminate against individuals based on aspects such as religion, ethnicity, nationality, race, colour, ancestry, or other identity factors. The rapid dissemination of information on platforms like Twitter makes it imperative to detect hate speech efficiently. This project aims to tackle the challenges of identifying subtle and overt forms of hate speech in tweets. By employing advanced transformer-based deep learning models, we seek to enhance the accuracy and efficiency of hate speech detection, ultimately promoting healthier social interactions in online communities.

2 Data Source

The dataset is provided by the company and can be accessed at the following link: https://www.kaggle.com/vkrahul/twitter-hate-speech?select=train_E6oV3lV.csv. It includes labelled Twitter data, with tweets classified as hate speech (label: 1) or non-hate speech (label: 0), specifically prepared for hate speech detection research. For model development, the data was divided into 80% for training and 20% for testing to ensure a reliable evaluation process.

3 Data Pre-processing

Properly cleaning the data is essential in hate speech detection because irrelevant elements, such as URLs, mentions, hashtags, and special characters, can introduce noise and bias, potentially impacting the model's accuracy and interpretability. For this project, we employed

regular expressions (regex) to streamline the data by filtering out non-essential components, ensuring that only meaningful text remained. This included removing URLs (`r"http\S+|www\S+"`), user mentions (`r"@\w+"`), and special characters, which enhanced focus on actual language patterns indicative of hate speech. Additionally, we converted all text to lowercase to standardize the format and reduce redundancy. Figure 3.1 shows this transformation process, highlighting how these steps ensure cleaner, more reliable data, ultimately improving the model's learning and performance.

After these steps, the pre-processed dataset was ready for input into the BERT model for training and evaluation.

```
In [21]: # View the first few rows of the dataset with cleaned text
         print(train_data[['tweet', 'cleaned_tweet']].head())

                                                tweet  \
0   @user when a father is dysfunctional and is s...
1  @user @user thanks for #lyft credit i can't us...
2                                bihday your majesty
3  #model   i love u take with u all the time in ...
4             factsguide: society now    #motivation

                                      cleaned_tweet
0    when a father is dysfunctional and is so sel...
1    thanks for lyft credit i cant use cause they...
2                                bihday your majesty
3  model   i love u take with u all the time in u...
4               factsguide society now    motivation
```

Figure 3.1: Example of Cleaned Data Prepared for Model Training

4 Exploratory Data Analysis

To obtain a comprehensive understanding of data distribution and characteristics, an exploratory data analysis was conducted, encompassing the following key areas:

i. Class Imbalance Analysis identified a significant skew in class distribution, with non-hate speech instances substantially outnumbering hate speech instances, as illustrated in Figure 4.1. Recognizing this imbalance informed subsequent modelling decisions to enhance performance and mitigate bias. We also analyzed the distribution of tweet lengths and observed a left-skewed pattern, further indicating the presence of data imbalance, as shown in Figure 4.2.

ii. Word Cloud Analysis illustrated in Figure 4.3 was performed for each class, revealing frequently occurring words distinctive to hate speech and non-hate speech. This visualization provided valuable insights into thematic content differences across categories. The size of each word correlates with its frequency, allowing for a quick visual interpretation of the language patterns that differentiate the two classes.

iii. Correlation Analysis: As shown in Figure 4.4, a low correlation coefficient (0.051) was observed between tweet length and hate speech, suggesting that tweet length does not significantly predict the presence of hate speech in the dataset.

In [14]:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Visualize the distribution of labels
plt.figure(figsize=(6,4))
sns.countplot(x='label', data=train_data)
plt.title("Label Distribution in Training Data")
plt.show()
```

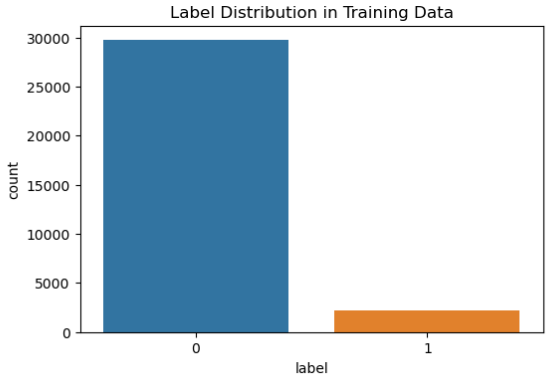

Figure 4.1: Class Imbalance Analysis

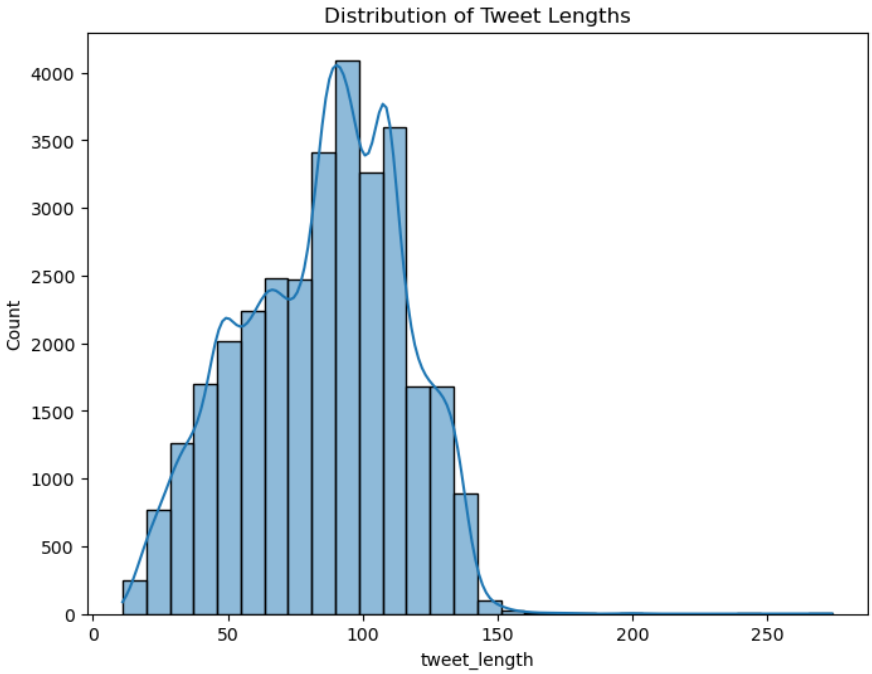

Figure 4.2: Distribution of tweet lengths

```
In [23]:
# Separate the positive (non-hate) and negative (hate speech) tweets
positive_tweets = train_data[train_data['label'] == 0]['cleaned_tweet']
negative_tweets = train_data[train_data['label'] == 1]['cleaned_tweet']

# Combine all positive and negative tweets into single strings
positive_text = ' '.join(positive_tweets)
negative_text = ' '.join(negative_tweets)

# Generate word clouds for each
positive_wordcloud = WordCloud(width=800, height=400, background_color='white').generate(positive_text)
negative_wordcloud = WordCloud(width=800, height=400, background_color='black').generate(negative_text)

# Plot the word clouds
plt.figure(figsize=(10,5))

# Word cloud for positive tweets (non-hate speech)
plt.subplot(1, 2, 1)
plt.imshow(positive_wordcloud, interpolation='bilinear')
plt.title("Word Cloud - Positive Tweets")
plt.axis('off')

# Word cloud for negative tweets (hate speech)
plt.subplot(1, 2, 2)
plt.imshow(negative_wordcloud, interpolation='bilinear')
plt.title("Word Cloud - Negative Tweets")
plt.axis('off')

plt.show()
```

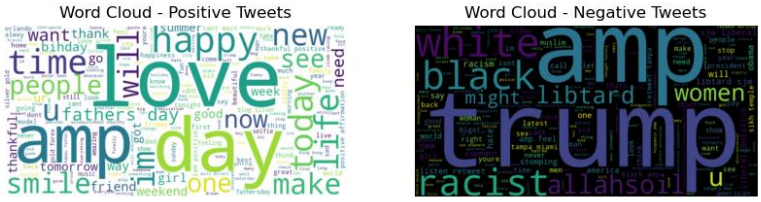

Figure 4.3: Word Cloud Analysis

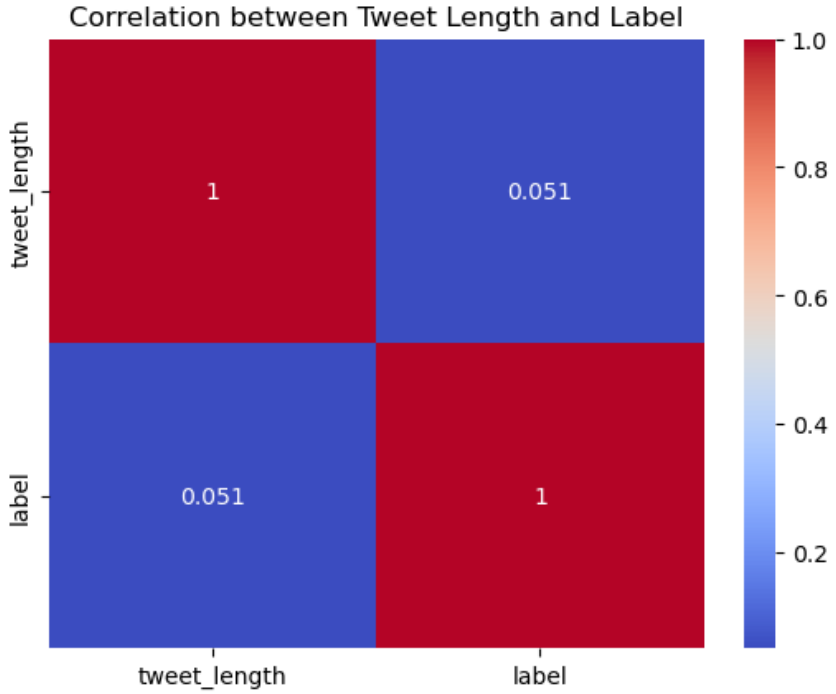

Figure 4.4: Correlation Analysis

5 Transforming the Data

Data transformation is a critical step in preparing the dataset for effective model training and evaluation. The first essential task was **encoding the labels**. In our analysis, hate speech was represented as 1, while non-hate speech was assigned a label of 0. This numerical encoding not only facilitated the processing of the data but also aligned with the requirements of machine learning algorithms, which typically operate on numerical inputs.

To address the inherent **class imbalance,** present in the dataset—where non-hate speech instances significantly outnumbered hate speech instances—class weights were incorporated into the model training process. By applying weights, we ensured that the model would not be biased towards the majority class. Instead, it would allocate more importance to the minority class, thereby enhancing the model's ability to accurately detect hate speech. This strategy is crucial in real-world applications where the costs of false negatives (failing to identify hate speech) can be substantial.

Subsequently, we conducted a **train-test split** of the dataset, allocating 80% for training and 20% for testing. This partitioning was designed to ensure that both sets contained representative samples of each class. A balanced representation is vital for validating the model's performance, as it allows for a more accurate assessment of how well the model generalizes to unseen data.

Through these transformations, we set the groundwork for effective model training, ensuring that the data was appropriately prepared for subsequent phases of the project.

6 Model Building

The model-building phase is a pivotal aspect of our project, as it directly influences the effectiveness of the hate speech detection system. To leverage the powerful capabilities of state-of-the-art natural language processing, we utilized a pre-trained BERT (Bidirectional Encoder Representations from Transformers) model, specifically the **bert-base-uncased** variant. This selection was guided by the model's robust performance in various NLP tasks, coupled with its relatively efficient computational requirements. The uncased nature of this model signifies that it does not differentiate between uppercase and lowercase letters, thereby simplifying the

tokenization process and making it particularly suitable for our dataset, which contains diverse and informal language typical of social media platforms.

A crucial modification made to the pre-trained BERT model was the integration of a custom classifier layer. This layer was designed to produce binary predictions, effectively categorizing tweets as either hate speech or non-hate speech. By placing this classifier on top of the BERT architecture, we harnessed the rich contextual embeddings generated by BERT while allowing the model to specialize in our specific classification task. This two-layer structure enabled the model to leverage pre-existing knowledge while adapting to the nuances of our dataset.

In the training process, we leveraged the capabilities of the Transformers framework, which enables efficient model optimization for our hate speech detection task. By using a structured approach for training configurations, we defined our training arguments, specifying an output directory for model checkpoints and setting the number of training epochs to three. The batch size was configured to 16 for both training and evaluation, optimizing the process to balance memory usage and training efficiency. To enhance our training workflow, we incorporated logging to capture metrics every ten steps, and we implemented an evaluation strategy that assesses the model's performance at the end of each epoch. Through this setup, we facilitated a comprehensive and organized training regimen, ensuring that the model was adequately trained and validated throughout the process.

Overall, the model architecture we constructed—anchored by the BERT framework and complemented by a dedicated classifier layer—was meticulously designed to achieve high accuracy in detecting hate speech. Through careful selection of pre-trained models, innovative layer design, and strategic optimization techniques, we established a solid foundation for subsequent training and evaluation stages.

7 Training and Validation

The training and validation phase are a critical component in optimizing our hate speech detection model. During this process, we focused on fine-tuning the BERT architecture specifically for our dataset, allowing the model to adjust the weights in the final layers to enhance its performance on the hate speech detection task. The training configuration included a batch size of 16 and a total of 3 training epochs. We also monitored training progress and

performance through validation at the end of each epoch, which enabled us to observe training loss and validation accuracy. This strategy was essential for detecting any signs of overfitting and ensuring that the model could generalize effectively to unseen data. By systematically evaluating the model after each epoch, we aimed to establish a robust framework for hate speech detection that balances accuracy with generalization, ultimately improving its applicability in real-world scenarios.

8 Evaluation and Metrics

The evaluation of our hate speech detection model was conducted using several key metrics to comprehensively assess its performance. The results were analyzed in terms of **precision**, **recall**, **F1-score**, and **support** for both classes: non-hate and hate speech.

For non-hate speech instances, the model achieved a precision of **0.98**, recall of **0.99**, and an F1-score of **0.98**, based on **5937** samples. In contrast, the hate speech category exhibited a precision of **0.79**, recall of **0.71**, and an F1-score of **0.75**, derived from **456** samples. Overall, the model attained an accuracy of **0.97**, with a total of **6393** samples evaluated. Furthermore, the macro average across both classes resulted in a precision of **0.89**, recall of **0.85**, and F1-score of **0.87**, reflecting a balanced performance across the dataset. The weighted averages indicated robust overall performance, with precision, recall, and F1-scores of **0.97**, **0.97**, and **0.97**, respectively.

To provide a clearer illustration of the model's performance, the evaluation metrics are depicted in **Figure 8.1**, which summarizes the precision, recall, F1-score, and support for each class. Additionally, the confusion matrix results, illustrated in **Figure 8.2**, further elucidate the model's predictive capabilities by showcasing the true positives, true negatives, false positives, and false negatives. These metrics not only highlight the model's strength in identifying non-hate speech but also pinpoint areas for improvement in detecting hate speech, informing future iterations of the model.

In [34]:

```
from sklearn.metrics import classification_report

# Get predictions from the model on the validation dataset
predictions = trainer.predict(val_dataset)
preds = predictions.predictions.argmax(-1)

# Print classification report
print(classification_report(y_test, preds, target_names=['Non-hate', 'Hate']))
```

```
              precision    recall  f1-score   support

    Non-hate       0.98      0.99      0.98      5937
        Hate       0.79      0.71      0.75       456

    accuracy                           0.97      6393
   macro avg       0.89      0.85      0.87      6393
weighted avg       0.97      0.97      0.97      6393
```

Figure 8.1: Evaluation Metrics

In [35]:

```
from sklearn.metrics import confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt

# Generate the confusion matrix
cm = confusion_matrix(y_test, preds)

# Plot the confusion matrix
plt.figure(figsize=(6,4))
sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=['Non-hate', 'Hate'], yticklabels=['Non-hate', 'Hate'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```

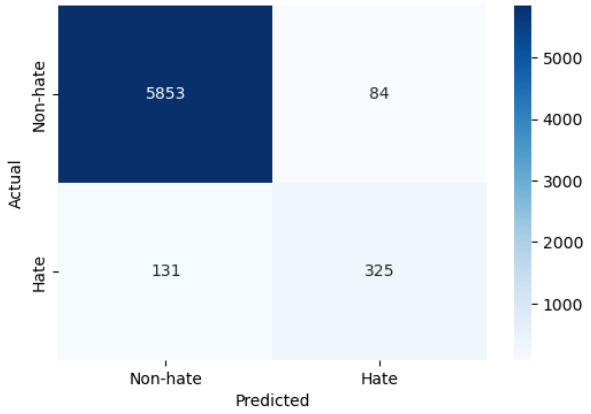

Figure 8.2: Confusion Matrix

9 Comparison of Model Performance

The performance of the Transformer model was compared to Logistic Regression (Figure 9.1) and XGBoost (Figure 9.2). The Transformer model achieved an accuracy of 97%, with F1-scores of 0.98 for non-hate speech and 0.75 for hate speech, demonstrating a balanced capacity to detect both classes. This high performance highlights its effectiveness in capturing the complexities of language within hate speech detection.

Logistic Regression, while accurate (92%), struggled to identify hate speech, achieving a lower precision (0.48) and F1-score (0.61) for this class. Similarly, XGBoost showed limitations in hate speech detection with an accuracy of 91%, precision of 0.43, and F1-score of 0.54. Both models were impacted by class imbalance, as illustrated in the confusion matrices, with a considerable portion of hate speech samples misclassified as non-hate.

The Transformer model's superior results underline the strength of transformer-based deep learning for nuanced text analysis, especially in applications like hate speech detection, as shown in figure 9.3.

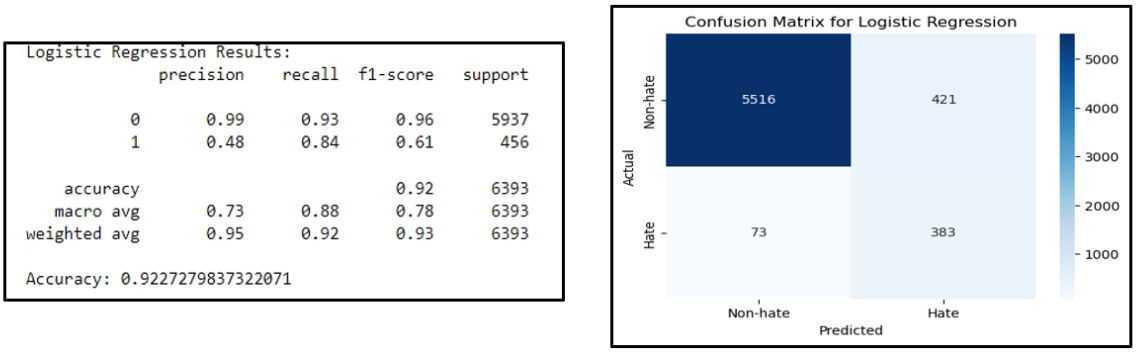

Figure 9.1: Logistic Regression evaluation metrics and confusion matrix

XGBoost Results:

	precision	recall	f1-score	support
0	0.98	0.93	0.95	5937
1	0.43	0.72	0.54	456
accuracy			0.91	6393
macro avg	0.70	0.82	0.74	6393
weighted avg	0.94	0.91	0.92	6393

Accuracy: 0.9111528234005944

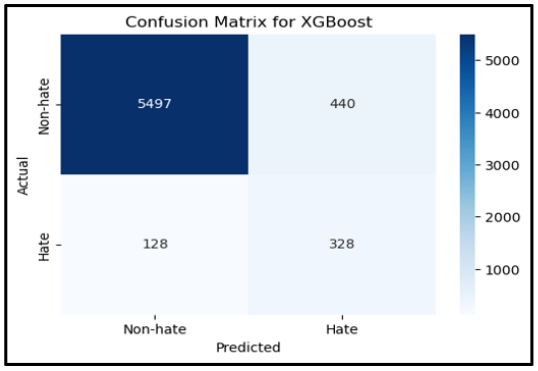

Figure 9.2: XGBoost evaluation metrics and confusion matrix

Model	Accuracy
Transformer	**97%**
Logistic Regression	**92%**
XGBoost	**91%**

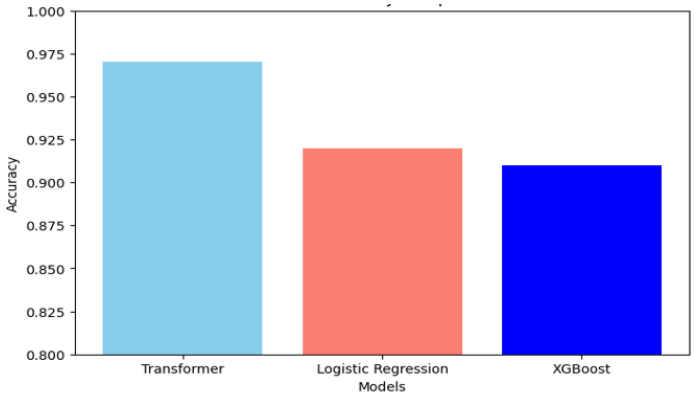

Figure 9.3: Model Accuracy Comparison

10 Model Inference

The final model was evaluated on unseen data to assess its performance in real-world scenarios, as illustrated in Figure 10.1. We conducted inference on a sample set of tweets, ensuring consistency with our previous processes. The BERT tokenizer was employed, along with padding and attention masks, to maintain uniformity in data preparation. Test samples were processed in batches to enhance efficiency and prevent memory overload. The model produced binary predictions, demonstrating accuracy and recall metrics that aligned closely with those obtained during the testing phase.

```
In [42]:
# Create the DataFrame to store predictions
predictions_df = pd.DataFrame({
    'tweet': test_data['cleaned_tweet'],  # Original tweet
    'predicted_label': preds              # Predicted label (0 for Non-hate, 1 for Hate)
})

# Save the predictions to a CSV file
predictions_df.to_csv('test_predictions.csv', index=False)

# Display a few rows to confirm
print(predictions_df.head())
```

```
                                               tweet  predicted_label
0  studiolife aislife requires passion dedication...                0
1    white supremacists want everyone to see the ...                1
2  safe ways to heal your acne    altwaystoheal h...                0
3  is the hp and the cursed child book up for res...                0
4    rd bihday to my amazing hilarious nephew eli...                0
```

Figure 10.1: Model Performance Evaluation on Unseen Data

11 Recommendation

Based on its superior performance, the Transformer model (BERT) is the most suitable for deployment, given its high accuracy and balanced detection capability across both hate and non-hate speech classes. This model demonstrates the strongest potential for reliable real-world applications in hate speech detection, ensuring both precision and consistency.

To further optimize the model's robustness, future enhancements could incorporate data augmentation and advanced fine-tuning techniques. Integrating ensemble methods may also improve prediction stability across diverse text inputs. Additionally, testing on larger datasets, alongside real-time monitoring in production, could refine the model's adaptability and effectiveness in dynamic environments, contributing to safer online spaces.

12 Conclusion

This project demonstrates that transformer models, specifically BERT, are highly effective for hate speech detection in social media data. By leveraging BERT's contextual language understanding, we achieved a balance of high accuracy, precision, recall, and F1-score. This model could serve as a robust tool for hate speech moderation and potentially be integrated into social media platforms to enhance online community safety.

13 Reflections on Learning

Through this project, we gained valuable insights into advanced NLP techniques and deep learning with transformer models. Key learnings included:

- **Data Pre-processing for NLP**: The importance of effective tokenization and managing input sequences for BERT.
- **Model Fine-Tuning**: The challenges and benefits of fine-tuning a pre-trained model for domain-specific tasks.
- **Model Evaluation**: Understanding and implementing classification metrics beyond accuracy, especially for imbalanced datasets.
- **Collaboration**: Working as a team required constant coordination and merging of individual contributions, enriching our project's scope and quality.

References

Analytics Vidhya. (2020). *Text Classification Pytorch | Build Text Classification Model*. [online] Available at: https://www.analyticsvidhya.com/blog/2020/01/first-text-classification-in-pytorch/.

Devlin, J., Chang, M.-W., Lee, K. and Toutanova, K. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. [online] arXiv.org. Available at: https://arxiv.org/abs/1810.04805.

PyTorch (2019). *PyTorch documentation — PyTorch master documentation*. [online] Pytorch.org. Available at: https://pytorch.org/docs/stable/index.html.

Sanjay.M (2018). *Why and how to Cross Validate a Model?* [online] Medium. Available at: https://towardsdatascience.com/why-and-how-to-cross-validate-a-model-d6424b45261f.

Vaswani, A., Brain, G., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., Kaiser, Ł. and Polosukhin, I. (2017). *Attention Is All You Need*. [online] Available at: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.