Sentiment Analysis Project Documentation

Introduction

Sentiment analysis is a critical task in natural language processing (NLP) that involves determining the emotional tone behind a series of words. This project focuses on implementing various machine learning and deep learning models to classify text data into three sentiment categories: positive, neutral, and negative. We utilize a dataset consisting of text samples and their corresponding sentiment labels to train, evaluate, and compare different models.

Dataset: Sentiment Analysis Dataset (kaggle.com)

Libraries and Tools

The project utilizes several Python libraries, including:

- Pandas: For data manipulation and analysis.
- NumPy: For numerical computations.
- NLTK (Natural Language Toolkit): For text preprocessing.
- TensorFlow: For building deep learning models.
- Matplotlib and Seaborn: For data visualization.
- Scikit-learn: For machine learning algorithms and evaluation metrics.
- WordCloud: For visualizing the most frequent words in the dataset.
- LIME (Local Interpretable Model-agnostic Explanations): For model interpretability.

Data Exploration

a. Dataset Description

   We used two datasets: train.csv and test.csv. Each dataset contains text samples along with their sentiment labels.

b. Initial Data Analysis
   - Displayed the first few rows of the training and test datasets.
     Function used: *.head()*

   - Checked the data types and non-null counts for each column.
     Function used: *.isna().sum()*

c. Encoding Sentiment Labels

The sentiment labels were encoded as follows:

- Positive: 4
- Neutral: 2
- Negative: 0

Data Preprocessing

a. Handling Missing Values

Missing values in the datasets were handled by dropping rows with NaN values to ensure data integrity and consistency.
Function used: *.dropna()*

b. Text Cleaning

A cleaning function was implemented to:

- Remove square brackets and their contents.
- Remove URLs.
- Remove non-alphabetic characters.
- Remove punctuation.
- Tokenize the text using TweetTokenizer.
- Remove stopwords.
- Lemmatize the tokens.

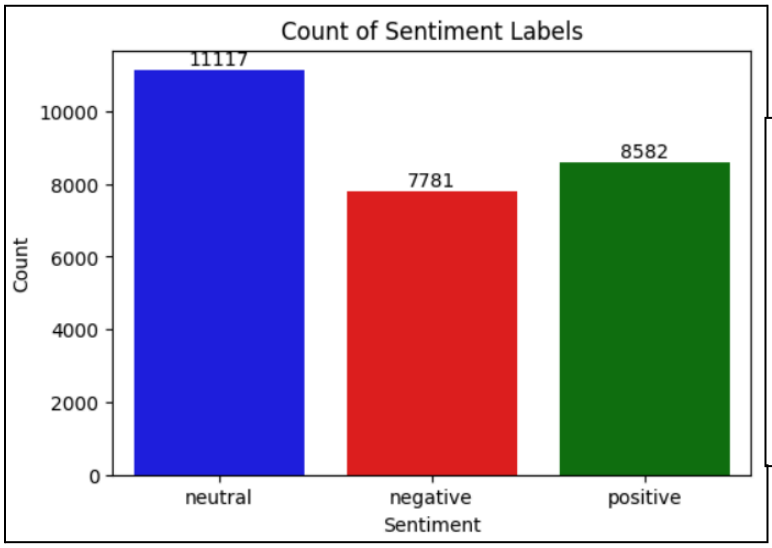
c. Feature Selection

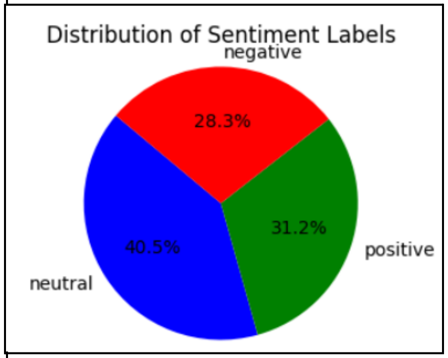
The following features were selected for the models:

- ‘text’
- ‘selected_text’
- ‘sentiment’
- ‘sentiment_encoded’

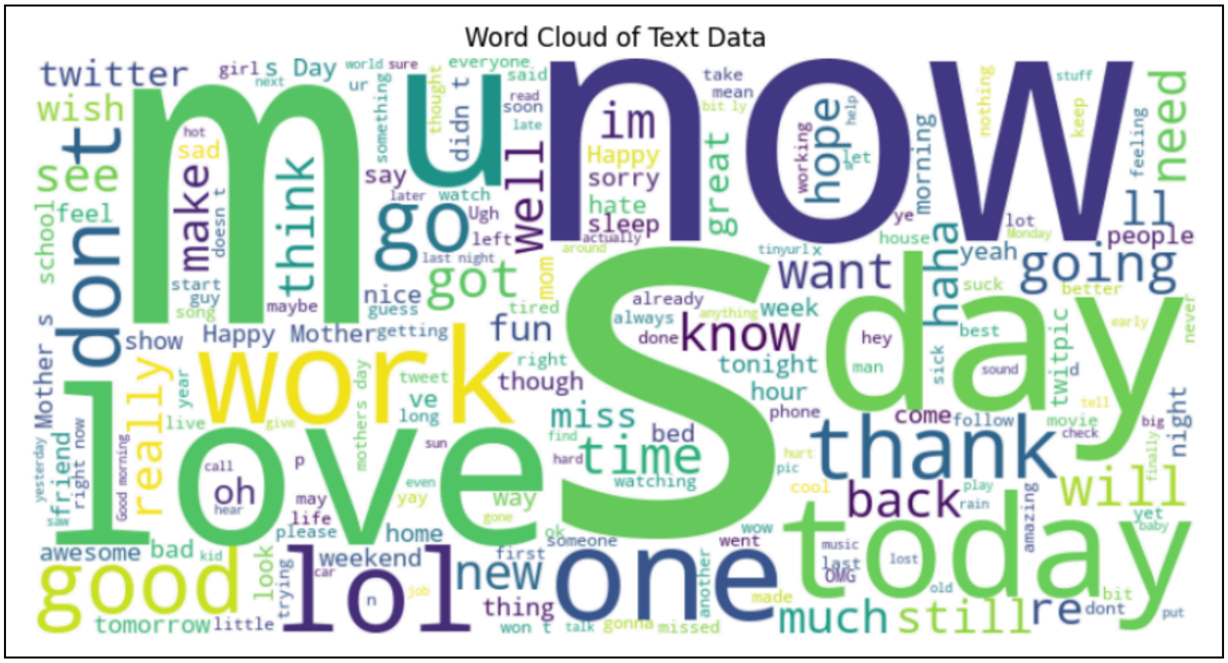
Exploratory Data Analysis

- Plotted the count of sentiment labels using a bar chart and a pie chart to check for class imbalances.

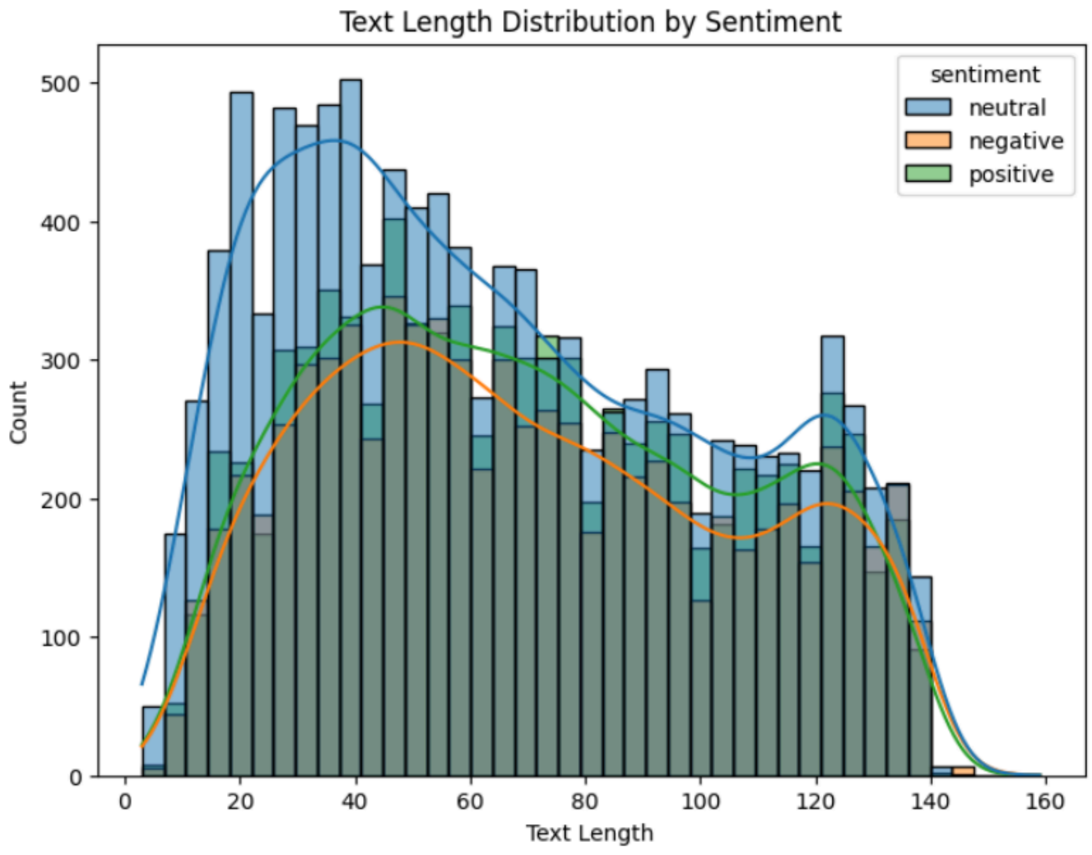




- Generated a word cloud to visualize the most common words in the training data.



- Analyzed the distribution of text lengths across different sentiments.



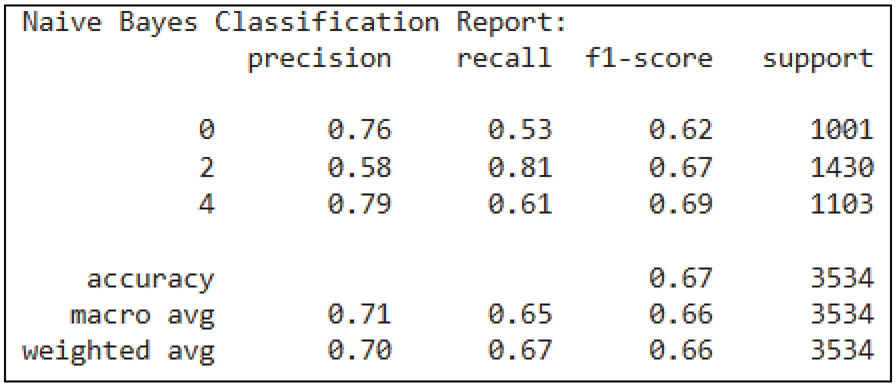
Text Vectorization

The text data was vectorized using the TF-IDF (Term Frequency-Inverse Document Frequency) method to convert the cleaned text into numerical features to implement machine learning models on the data. The vectorizer was fitted on the training data and transformed both the training and test data.

Model Selection and Evaluation

a. Naive Bayes Classifier
   - Implemented a Multinomial Naive Bayes classifier.
   - Evaluated the model using accuracy score and classification report.
   - Hyperparameter Tuning for Naive Bayes - Conducted Grid Search and Random Search to optimize the hyperparameters.

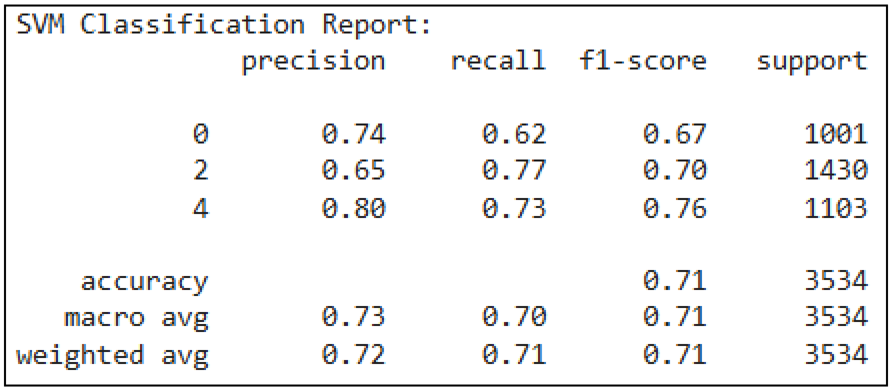
- Evaluated the tuned models and reported the best parameters and accuracy scores.



```
Naive Bayes Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.53      0.62      1001
           2       0.58      0.81      0.67      1430
           4       0.79      0.61      0.69      1103

    accuracy                           0.67      3534
   macro avg       0.71      0.65      0.66      3534
weighted avg       0.70      0.67      0.66      3534
```

b. Support Vector Classifier (SVC)

- Implemented a linear SVM classifier.
- Evaluated the model using accuracy score and classification report.

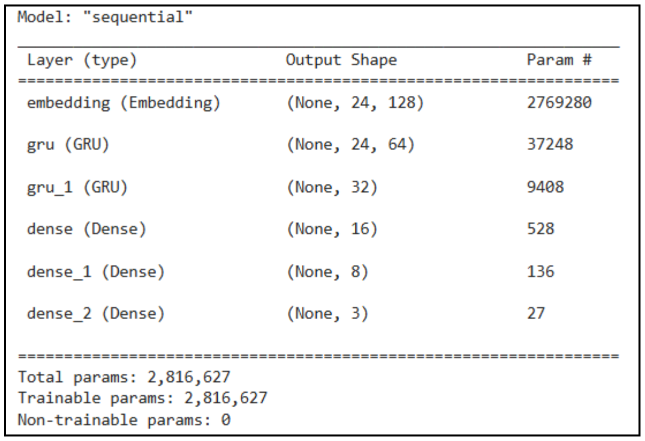


```
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.62      0.67      1001
           2       0.65      0.77      0.70      1430
           4       0.80      0.73      0.76      1103

    accuracy                           0.71      3534
   macro avg       0.73      0.70      0.71      3534
weighted avg       0.72      0.71      0.71      3534
```

c. GRU Model

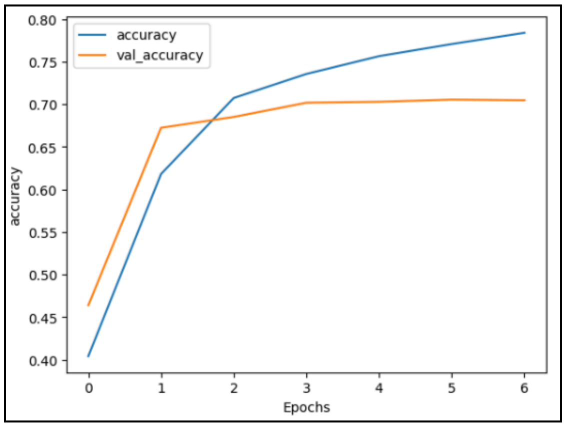
A Gated Recurrent Unit (GRU) model was implemented with the following architecture:

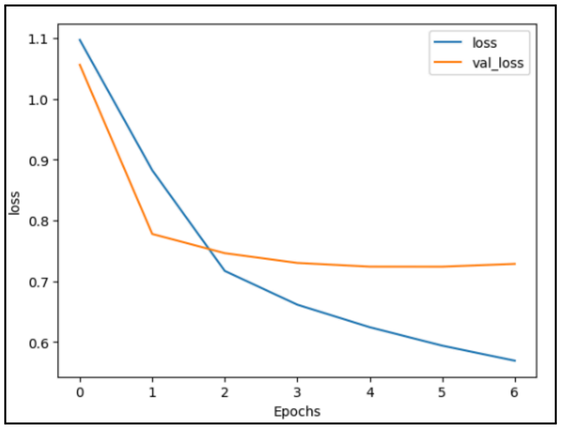
- Embedding layer.
- Two GRU layers.
- Dense layers with ReLU activation.
- Output layer with softmax activation.



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 24, 128)           2769280

 gru (GRU)                   (None, 24, 64)            37248

 gru_1 (GRU)                 (None, 32)                9408

 dense (Dense)               (None, 16)                528

 dense_1 (Dense)             (None, 8)                 136

 dense_2 (Dense)             (None, 3)                 27

=================================================================
Total params: 2,816,627
Trainable params: 2,816,627
Non-trainable params: 0
```

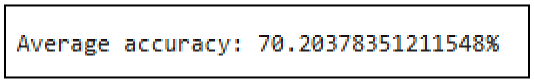
The model was trained and evaluated on the padded text sequences, and its performance was measured in terms of loss and accuracy.





Cross-Validation

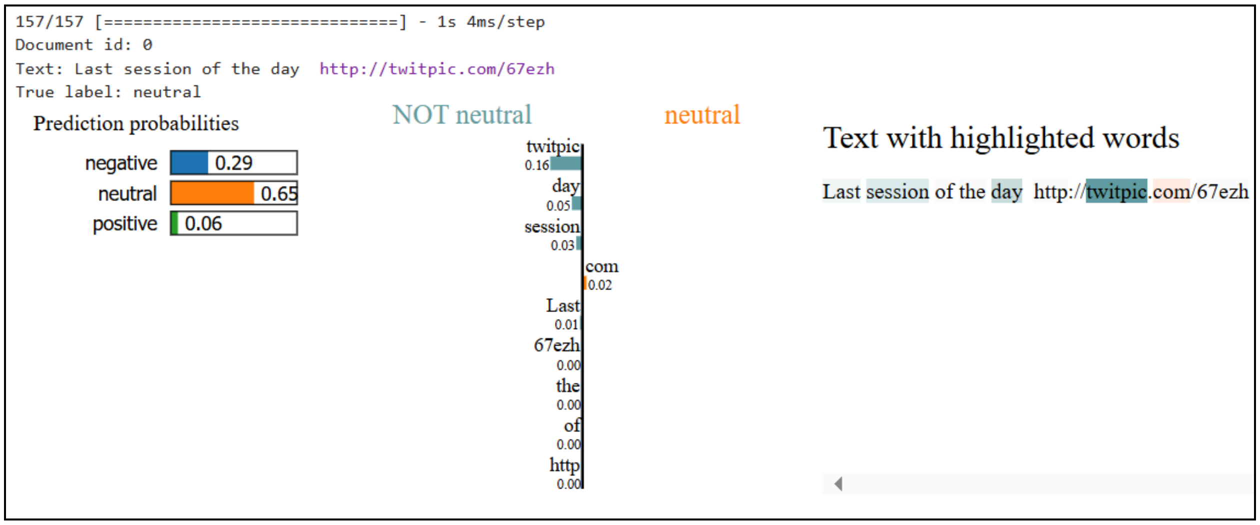
Performed k-fold cross-validation (k=5) to assess the stability and generalization of the GRU model. The average accuracy across the folds was 70.2%.



```
Average accuracy: 70.20378351211548%
```

Model Interpretability

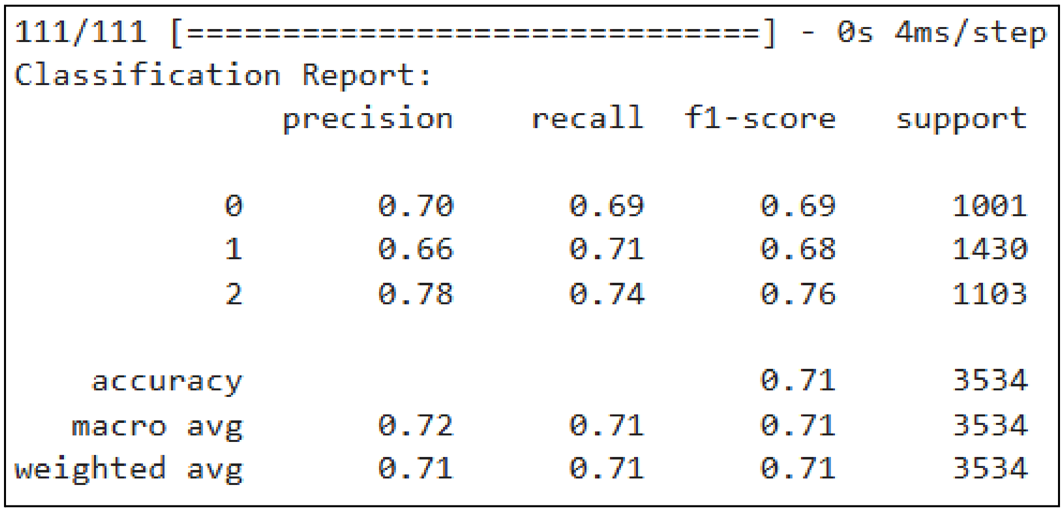
Utilized LIME to interpret the GRU model's predictions. A sample text from the test set was analyzed to understand the model's decision-making process.



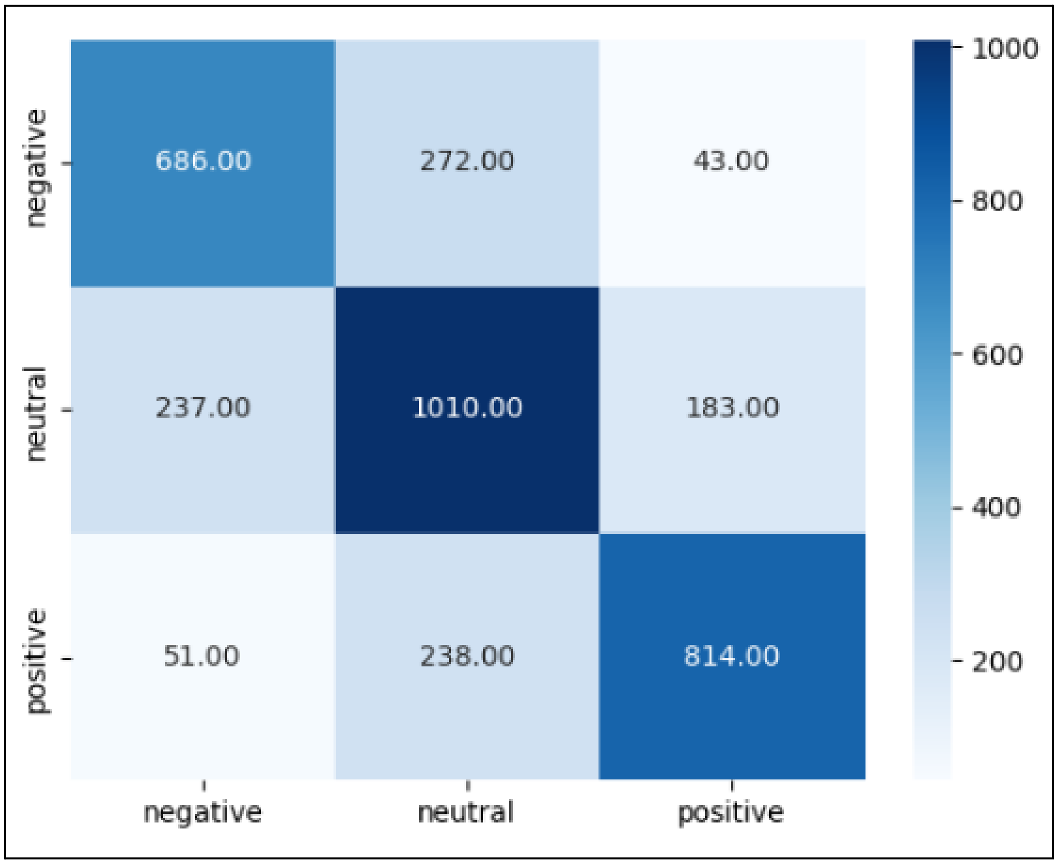
```
157/157 [==============================] - 1s 4ms/step
Document id: 0
Text: Last session of the day  http://twitpic.com/67ezh
True label: neutral
```

Final Evaluation

The GRU model's performance was further evaluated using a confusion matrix and classification report to provide a detailed breakdown of its predictive capabilities.

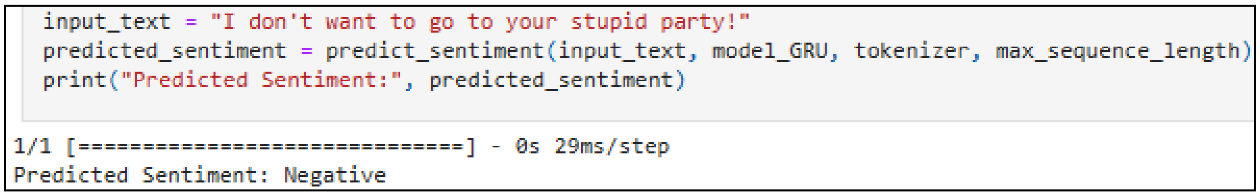


```
111/111 [==============================] - 0s 4ms/step
Classification Report:
              precision    recall  f1-score   support

           0       0.70      0.69      0.69      1001
           1       0.66      0.71      0.68      1430
           2       0.78      0.74      0.76      1103

    accuracy                           0.71      3534
   macro avg       0.72      0.71      0.71      3534
weighted avg       0.71      0.71      0.71      3534
```



Predictions on Custom Data

A function was created to predict the sentiment of new, unseen text inputs using the trained GRU model.



```
input_text = "I don't want to go to your stupid party!"
predicted_sentiment = predict_sentiment(input_text, model_GRU, tokenizer, max_sequence_length)
print("Predicted Sentiment:", predicted_sentiment)
```

```
1/1 [==============================] - 0s 29ms/step
Predicted Sentiment: Negative
```

Conclusion

This project successfully demonstrated the application of various machine learning and deep learning techniques for sentiment analysis. The GRU model, enhanced with hyperparameter tuning and interpretability methods, provided a robust solution for classifying text sentiment. Future work could explore more advanced NLP techniques and larger datasets to further improve model accuracy and generalization.