Evaluating Machine Learning and Deep Learning Models for Effective Sentiment Analysis: A Focus on Social Media and Mental Health Use Cases

Submitted by Devesh Dhyani to the University of Exeter
as a dissertation for the degree of
Master of Science in August 2024

I certify that all material in this dissertation which is not my own work has been identified and that any material that has previously been submitted and approved for the award of a degree by this or any other University has been acknowledged.

Table of Content

1. Introduction

Sentiment analysis, a crucial aspect of natural language processing, plays a pivotal role in understanding and interpreting the emotions and opinions expressed in textual data. In today's data-driven world, the ability to accurately gauge public sentiment is invaluable for a wide range of applications, including marketing, politics, and social sciences **(Gunasekaran, 2023)**. Over the years, sentiment analysis has evolved significantly, moving from simple keyword-based approaches to sophisticated machine learning algorithms capable of capturing nuanced emotions within text **(Jim et al., 2024).**

Social media platforms such as Twitter, Reddit, and Instagram have emerged as rich sources of user-generated content, offering a wealth of data for sentiment analysis. The immense volume and variety of this data present both substantial opportunities and considerable challenges. Analysing social media data can provide real-time insights into public opinion, customer satisfaction, and emerging trends **(Xu et al., 2022).** However, the informal and unstructured nature of social media language—characterized using slang, emojis, and abbreviations—adds layers of complexity to sentiment analysis, making it a challenging task.

In addition to its traditional applications, sentiment analysis is increasingly being explored within the domain of mental health. The growing prevalence of mental health discussions on social media, particularly on platforms like Twitter, presents a unique opportunity to analyse public sentiment related to issues such as anxiety, depression, and suicidal ideation **(Keles et al., 2019).** By leveraging advanced machine learning techniques, sentiment analysis can be employed to identify and monitor mental health trends, offering insights that could inform public health strategies and interventions. This extension of sentiment analysis into the mental health field highlights its potential to contribute significantly to societal well-being by providing real-time data on public mental health.

Despite the advancements in sentiment analysis techniques, the task of accurately and efficiently analysing the diverse data generated on social media platforms remains complex. Current methods often struggle with the variability and intricacies of social media language, leading to inconsistent sentiment predictions. This complexity is particularly pronounced in the context of mental health, where the language used is often laden with nuance, including expressions of sarcasm, irony, and varying levels of emotional intensity **(Li, 2024).** Existing models may not fully capture these subtleties, resulting in potential misclassification and reduced effectiveness. Addressing these challenges is crucial, as more accurate sentiment analysis in mental health contexts can lead to better detection of mental health issues, more timely interventions, and improved support mechanisms. Enhancing

models to manage the specific challenges of mental health data will provide deeper insights into public sentiment, thereby informing more effective mental health policies and strategies.

The primary objective of this dissertation is to evaluate the effectiveness of various machine learning and deep learning models for sentiment analysis of social media data, particularly in the context of mental health. The research aims to assess which machine learning models are most effective for analysing sentiments related to mental health on social media platforms and to identify the limitations and challenges associated with current sentiment analysis techniques in this context. Furthermore, this study seeks to explore how the findings can be applied in real-world scenarios, such as mental health monitoring and intervention, thus demonstrating the practical applications of sentiment analysis in improving public health responses.

This exploration of sentiment analysis through ML and DL methods offers a promising avenue for addressing the complexities of interpreting social media data. By evaluating the effectiveness of various models, this dissertation aims to contribute to the development of more robust and adaptable sentiment analysis tools. The outcomes of this research will provide valuable insights into the strengths and limitations of current techniques, guiding future advancements in the field. Moreover, by focusing on real-world use cases such as mental health monitoring, this study underscores the potential impact of sentiment analysis in improving public health strategies and responses. As social media continues to grow as a dominant communication channel, the need for effective sentiment analysis becomes increasingly critical, particularly in areas as sensitive and important as mental health.

2. Literature Review

2.1. Evolution of Sentiment Analysis Techniques

Early Approaches in Sentiment Analysis

Sentiment analysis, also known as opinion mining, emerged as a significant research area in the early 2000s. Initially, researchers relied heavily on keyword-based approaches to determine the sentiment of a text. These methods identified positive or negative sentiments based on the presence of specific words. While effective for basic applications, such as analysing customer reviews, these early techniques struggled to interpret nuanced, context-dependent language **(Tan et al., 2023b**). As the limitations of these rudimentary approaches became apparent, researchers sought more advanced methods to improve the accuracy of sentiment analysis. The field began to evolve beyond simple keyword detection, leading to the exploration of more sophisticated techniques that could better handle the complexities of human language **(*Sentiment Analysis and Opinion Mining*, n.d.)**

Transition to Machine Learning Models

The integration of machine learning into sentiment analysis marked a pivotal shift from rule-based systems to data-driven models. Machine learning algorithms, such as Naive Bayes, Support Vector Machines (SVM), and decision trees, significantly enhanced the accuracy and scalability of sentiment analysis. These algorithms learned patterns from labelled datasets, outperforming traditional keyword-based methods **(*Review on Sentiment*, n.d.).** As machine learning techniques advanced, the development of ensemble methods and neural networks further expanded the capabilities of sentiment analysis. These innovations allowed for the processing of large-scale data and more complex linguistic patterns, making sentiment analysis increasingly reliable and applicable across diverse domains, including marketing, finance, and mental health **(Kleppen, 2023).**

2.2. Advances in Machine Learning for Sentiment Analysis

Supervised and Unsupervised Learning in Sentiment Analysis

Supervised learning remains a dominant approach in sentiment analysis, with algorithms such as Logistic Regression, Random Forests, and Neural Networks demonstrating high accuracy in classifying sentiments from text data. Malik and Jain in 2020 conducted a comparative study of machine learning algorithms for social media text analysis, revealing that ensemble methods, particularly Random Forests and Gradient Boosting Machines, often outperform individual classifiers **(Malik & Jain, 2020).** Their study highlights the importance of combining multiple models to achieve higher accuracy, emphasizing the robustness of ensemble approaches in sentiment analysis.

In recent years, unsupervised learning and deep learning techniques have also gained prominence. Deep learning models, particularly Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs), have shown exceptional performance in capturing complex patterns in text, explored deep learning-based sentiment analysis on multilingual data, demonstrating that models utilizing word embeddings and attention mechanisms significantly enhance prediction accuracy **(Shanmugavadivel et al., 2022).** These findings underscore the adaptability and power of deep learning in handling diverse linguistic contexts, making them valuable tools for complex sentiment analysis tasks.

2.3. Sentiment Analysis in the Context of Social Media and Mental Health

Challenges of Sentiment Analysis on Social Media Platforms

Social media platforms present unique challenges for sentiment analysis due to their varying characteristics. Twitter, with its 280-character limit, requires models that can handle short and often ambiguous text. Elmitwalli and Mehegan (2024) analysed sentiment on Twitter using pre-trained models like BERT and GPT-3, finding that these advanced models significantly outperformed traditional techniques in capturing the nuances of Twitter language **(Elmitwalli & Mehegan, 2024).** Their study underscores the importance of utilizing platform-specific models to address the unique challenges posed by different social media environments.

Xu et al. (2022) conducted a systematic review of sentiment analysis across various social media platforms, highlighting the challenges posed by platform-specific characteristics, such as informal language on Twitter and the visual-textual mix on Instagram. They emphasized the need for models tailored to specific platforms to improve the accuracy and effectiveness of sentiment analysis **(Xu et al., 2022b).**

Application of Sentiment Analysis in Mental Health

Recent research has begun exploring the application of sentiment analysis in mental health, reflecting the growing recognition of social media as a valuable source of data for understanding public mental health trends. Studies have shown that sentiment analysis can be particularly useful in detecting mental health conditions like depression and anxiety based on social media activity. For instance, a study analysed Twitter data to identify patterns of anxiety and depression using sentiment analysis, revealing that negative language and first-person pronouns were strongly correlated with depressive states **(Zhang et al., 2023b).** Another study monitored public sentiment during a mental health awareness campaign, demonstrating how sentiment analysis could measure the campaign's impact on public awareness and attitudes **(Saha et al., 2019).**

Deep learning models, especially those using word embeddings and attention mechanisms, have significantly enhanced the accuracy of sentiment predictions in mental health contexts. For example, a BERT-based model was used to analyse tweets related to suicidal ideation, achieving high accuracy in detecting at-risk individuals. These studies highlight the potential of advanced sentiment analysis models in providing timely insights that can inform mental health interventions and policies **(Boonyarat et al., 2024)**

However, challenges remain, particularly in handling sarcasm, irony, and diverse expressions of mental health concerns online. These complexities necessitate continued development of more sophisticated models that can account for these nuances, ensuring that sentiment analysis remains a reliable tool in mental health contexts.

2.4. Theoretical Framework for Sentiment Analysis in Mental Health

Natural Language Processing (NLP) and Machine Learning Integration

Natural Language Processing (NLP) is the foundation of sentiment analysis, with techniques such as tokenization, stemming, lemmatization, and part-of-speech tagging essential for preparing text data for analysis. These methods break down text into manageable components, normalize variations, and identify grammatical structures, making it easier for machine learning algorithms to process the data **(Olaoye & Potter, 2024).** NLP serves as the foundational layer that enables effective processing and interpretation of textual data.

Machine learning paradigms, including supervised, unsupervised, and deep learning approaches, provide the computational framework for building sentiment analysis models. Supervised learning algorithms like Logistic Regression and SVM learn from labelled data to classify sentiments, while unsupervised techniques, such as clustering and topic modeling, uncover hidden patterns in text data without predefined labels **(Sarker, 2021).** Deep learning models, particularly those based on neural networks, have revolutionized sentiment analysis by enabling the extraction of high-level features from text, critical for understanding the subtleties of language.

Statistical Methods and Model Validation

Statistical methods play a crucial role in refining sentiment analysis models. Techniques such as regression analysis and dimensionality reduction are essential for enhancing model performance. Regression analysis helps in understanding the relationships between different features and sentiment outcomes, while dimensionality reduction techniques like Principal Component Analysis (PCA) reduce the complexity of data, enhancing model performance and interpretability **(Bordoloi & Biswas, 2023).** These theoretical foundations create a comprehensive framework for analyzing sentiments in diverse and complex social media data.

2.5. Research Gaps and Future Directions

Identifying Limitations in Current Models.

While previous studies have effectively demonstrated the potential of machine learning and deep learning models in sentiment analysis, particularly on general social media data, this dissertation extends these applications specifically to the domain of mental health. For instance, Elmitwalli and Mehegan (2024) and Xu et al. (2022) have explored sentiment analysis on platforms like Twitter, focusing on pre-trained models like BERT and GPT-3, which excel in capturing nuances in social media language. However, these studies primarily target generic sentiment analysis without delving deeply into the specific challenges associated with mental health contexts. In contrast, my work not only leverages similar advanced models but also fine-tunes them for the complexities of mental health-related content, including the subtleties of expressions tied to anxiety, depression, and suicidal ideation.

Furthermore, existing literature, such as the work of Boonyarat et al. (2024), has showcased the effectiveness of BERT-based models in detecting suicidal ideation from social media data. While their results are promising, this dissertation advances the field by applying a comparative framework across multiple models—including Random Forest, LSTM, and TextCNN—and by retraining these models specifically for mental health datasets. This approach allows for a broader understanding of which models perform best under different conditions, something that is often missing in studies that focus on a single model.

Differentiating Approaches in Mental Health Data Analysis

A significant contribution of this research lies in the nuanced handling of mental health data. Traditional sentiment analysis methods, as reviewed by Malik and Jain (2020) and Shanmugavadivel et al. (2022), often struggle with the informal, highly varied language found on social media platforms. My work addresses this by implementing extensive pre-processing techniques tailored for mental health-related text, such as advanced tokenization and the removal of non-textual elements like emojis, which are prevalent in social media but can obscure sentiment analysis results.

Additionally, my use of hybrid models, such as CNN-LSTM and CNN-GRU, represents a novel approach in this field. These models combine the strengths of convolutional layers for capturing local patterns in text with recurrent layers that excel at modelling sequential data. This combination has proven particularly effective in handling the complex and often evolving expressions seen in mental health discussions, offering a more accurate and contextually aware analysis than what has been reported in earlier works.

Real-World Applications of Sentiment Analysis in Mental Health

This research aims to address these gaps by evaluating the effectiveness of machine learning and statistical models in sentiment analysis, particularly in the context of mental health. By focusing on mental health-related data, this study seeks to identify the most effective approaches for analysing sentiments related to mental health conditions, contributing to the development of more robust sentiment analysis models. These findings will have practical applications in mental health monitoring and intervention, demonstrating the potential of sentiment analysis to improve public health responses.

3. Methodology

3.1. Data Collection

For this study, data collection was a critical step in ensuring the robustness and relevance of the sentiment analysis models developed and evaluated. Two datasets were selected from Kaggle, a widely recognized platform for sharing and accessing high-quality datasets, which were pivotal in addressing the dual objectives of this research: determining the best-performing sentiment analysis model on generic social media data and applying this model to mental health-related data. The use of Kaggle's datasets not only provided a diverse and extensive range of data but also ensured that the datasets had been previously vetted and pre-processed by the community, adding a layer of reliability to the analysis. The significance of data selection in sentiment analysis cannot be overstated, as the quality and relevance of the data directly impact the performance of the models **(García-Hernández et al., 2024).** Furthermore, Kaggle offers datasets that are frequently updated and reflective of current

trends, which is essential for studies focusing on dynamic fields such as social media and mental health.

The first dataset, titled "Twitter Entity Sentiment Analysis," was employed to identify the most effective model for generic sentiment analysis. This dataset consists of a comprehensive collection of tweets labelled with sentiment scores, allowing for the training and evaluation of various machine learning (ML) models. The dataset includes approximately 70,000 labelled messages, providing a diverse range of topics that reflect the informal language often found on social media platforms like Twitter. This diversity and informality present unique challenges that are representative of broader issues in sentiment analysis, making it an ideal dataset for benchmarking the performance of different models. The dataset's entity-level annotation allows for a more nuanced analysis of sentiment directed towards specific subjects, which is crucial in applications like brand monitoring and public opinion analysis.

The second dataset, titled "Sentiment Analysis for Mental Health," was specifically selected to extend the research into the domain of mental health. This dataset includes text data related to mental health issues, such as depression, anxiety, and suicidal ideation, with corresponding sentiment labels. It provides a focused lens on how sentiment analysis can be applied to detect and monitor mental health conditions based on social media activity. The specialized nature of this dataset allows for a deeper exploration of how well sentiment analysis models, initially trained on generic data, can adapt to the nuances of mental health-related language. This adaptability is crucial for ensuring that the models can effectively contribute to mental health monitoring and intervention efforts, particularly in identifying at-risk individuals through their social media interactions.

These datasets form the foundation of the research, enabling a comprehensive evaluation of sentiment analysis models in both general and mental health-specific contexts.

3.2. Data Pre-processing

To ensure the datasets were in optimal condition for analysis, extensive pre-processing was conducted on both the "Twitter Entity Sentiment Analysis" and the "Sentiment Analysis for Mental Health" datasets. This pre-processing phase was crucial for standardizing the data and enhancing the performance of the sentiment analysis models. Pre-processing is essential in sentiment analysis as it reduces noise and irrelevant information, allowing the models to focus on significant aspects of the text, which ultimately improves the accuracy and efficiency of the analysis **(Siino et al., 2024).**

Text Cleaning and Preparation

The raw text data from both datasets underwent a series of cleaning and preparation steps. The first step involved tokenization, where each text entry was broken down into individual tokens or words. Tokenization is a fundamental process in natural language processing (NLP) as it converts text into a form that is easier for machine learning models to analyze and process at the word level **(Nadkarni et al., 2011)**. Following tokenization, normalization was performed to standardize the text. This included converting all text to lowercase to ensure uniformity and removing any unnecessary whitespace that could interfere with model training. These steps are essential in reducing the complexity of the data while preserving its informative content.

Stop Words Removal

Commonly used words that do not contribute significantly to the sentiment of a sentence, known as stop words (e.g., "and," "the," "is"), were removed from the datasets. Stop words are typically removed to reduce the dimensionality of the data, which helps the model focus on the words that carry meaningful sentiment information **(Yadav & Gayen, 2023).** By eliminating these non-informative words, the models can more efficiently identify patterns and relationships within the data relevant to sentiment analysis. This process is widely recognized for its role in improving both the speed and accuracy of machine learning models in various text classification tasks [84].

Stemming and Lemmatization

To further refine the text, stemming and lemmatization were applied. Stemming involves reducing words to their root forms, while lemmatization transforms words to their base or dictionary form. Both processes are designed to consolidate variations of words that share the same meaning, such as "running" and "ran" being reduced to "run". This step is particularly important for ensuring that the models do not treat different forms of the same word as distinct entities, which could lead to inaccuracies in sentiment classification **(De Freitas Santos & Varges, 2023).** Stemming is often used in information retrieval systems to enhance the recall of results by grouping similar words, whereas lemmatization tends to be more precise but computationally expensive. The choice of whether to use stemming or lemmatization depends on the specific requirements of the analysis and the trade-off between precision and computational efficiency.

Emoji and Link Removal

Given the informal nature of social media text, special attention was paid to removing emojis and hyperlinks from the datasets. Emojis, while often carrying significant emotional content, were removed to focus on textual analysis, as their inclusion could complicate the interpretation and consistency of the sentiment analysis. Similarly, hyperlinks were stripped from the text to eliminate noise and potential distractions that do not contribute to sentiment **(Abidin et al., 2024).**

These pre-processing steps were essential in transforming the raw, unstructured text data into a format suitable for machine learning analysis. By standardizing the text, reducing noise, and focusing on the most relevant linguistic features, the datasets were prepared to be effectively utilized in the subsequent model training and evaluation phases.

3.3. Machine Learning and Deep Learning Techniques

In this section, I will provide a detailed explanation of the machine learning and deep learning techniques used in this study. This section will serve as a comprehensive overview of the methods employed, including their theoretical foundations and general implementation, before applying them to specific datasets in subsequent sections.

Overview of Machine Learning Models

- **Logistic Regression**: Logistic Regression is a widely used linear model for binary classification tasks, but it can be extended to multi-class classification problems through methods like One-vs-Rest (OvR). In this study, Logistic Regression was used as a baseline model due to its simplicity and interpretability. The model was trained to predict the sentiment labels of the text data by finding a linear decision boundary that best separates the different classes.
- **Random Forest**: Random Forest is an ensemble learning method that combines multiple decision trees to improve classification performance. Each tree in the forest is trained on a random subset of the data, and the final prediction is made by averaging the predictions of all trees. This model is known for its robustness and ability to handle large amounts of data with high dimensionality. In this study, Random Forest was employed to capture complex interactions between features in the text data.
- **Naive Bayes**: Naive Bayes is a probabilistic model that applies Bayes' theorem with the assumption of independence between features. Despite its simplicity, Naive Bayes is particularly effective for text classification tasks. The model calculates the probability of each sentiment class given the input features (words or tokens) and assigns the class with the highest probability. Due to its efficiency and effectiveness in handling large text datasets, Naive Bayes was included in the model suite.
- **Extra Trees Classifier**: Extra Trees Classifier, or Extremely Randomized Trees, is another ensemble learning method like Random Forest but with more randomness in the choice of splits. This additional randomness can lead to improved generalization performance. The model was included in this study to evaluate whether this increased randomness would yield better performance in sentiment analysis compared to the more traditional Random Forest approach.

Overview of Deep Learning Models:

- **Long Short-Term Memory (LSTM):** LSTM is a type of recurrent neural network (RNN) designed to capture long-term dependencies in sequential data. It mitigates the vanishing gradient problem often encountered in traditional RNNs, making it particularly effective for sentiment analysis. In this study, LSTM networks were used to capture the temporal dependencies and the flow of sentiment within sentences.
- **Recurrent Neural Network (RNN):** RNNs are designed for sequential data, where the order of words in a sentence matters. They maintain a hidden state that captures information from previous time steps. Although less powerful than LSTMs in handling long-term dependencies, RNNs were employed to model the sequential nature of text data.
- **Gated Recurrent Unit (GRU):** GRUs simplify the architecture of LSTMs while retaining their effectiveness. They have fewer parameters, which can lead to faster training times without significant performance loss. GRUs were included to evaluate whether their simpler architecture could match or exceed the performance of LSTMs.
- **Text Convolutional Neural Network (TextCNN):** TextCNN applies convolutional layers to text data to extract local features. It is particularly effective at capturing n-gram features and is less sensitive to text length. In this study, TextCNN was used to identify patterns and key phrases in the text data that contribute to the overall sentiment.
- **CNN-LSTM Hybrid Model:** This model combines CNNs and LSTMs, where CNN layers extract local features that are then passed to LSTM layers to capture long-term dependencies. This combination captures both local and global patterns in the text, making it powerful for sentiment analysis.
- **CNN-GRU Hybrid Model:** Similar to the CNN-LSTM model, this hybrid combines CNNs with GRUs. The simpler GRU architecture, when combined with CNNs, was assessed to determine if it could provide performance advantages over the more complex LSTM-based hybrid model.

Statistical Analysis

In addition to training and testing the models, several statistical techniques were applied to further refine and understand the data and model performance.

- **Dimensionality Reduction:** Dimensionality reduction techniques, such as Principal Component Analysis (PCA), were employed to reduce the complexity of the feature space while retaining the most important information. This step was crucial, particularly for the deep learning models, as it helped mitigate the curse of dimensionality and reduced the risk of overfitting. PCA transformed the high-dimensional text data into a lower-dimensional space, making it more manageable for model training without significant loss of information.
- **Correlation Analysis:** Correlation analysis was conducted to identify relationships between different features and the target sentiment labels. Understanding these correlations helped in feature selection and engineering, allowing the models to focus on the most predictive features . The correlation analysis also provided insights into how different sentiment classes might be related, potentially improving the model's ability to distinguish between similar classes.

- **Confusion Matrix:** The confusion matrix was used to evaluate the performance of the models by providing a detailed breakdown of the predictions. It displayed the true positives, true negatives, false positives, and false negatives for each sentiment class. The confusion matrix was particularly useful for identifying which classes the models struggled with the most, such as confusing positive sentiments with neutral ones or misclassifying sentiments related to mental health conditions. This information was then used to further refine the models and improve their accuracy in critical areas.

By testing a variety of both traditional machine learning and advanced deep learning models, along with applying statistical analysis techniques, this study aimed to identify the most robust and accurate model for sentiment analysis, particularly in the context of mental health monitoring. Each model's performance was thoroughly analysed to determine its strengths and weaknesses, providing insights into the most effective approaches for handling the complexity and nuances of sentiment expressed in social media text data.

3.4. Model Training and Validation

The selected models were trained on the pre-processed "Twitter Entity Sentiment Analysis" dataset to identify the best-performing model for generic sentiment analysis. During training, hyperparameter tuning was performed using grid search and cross-validation techniques to identify the optimal settings for each model. Grid search systematically explored combinations of hyperparameters, while cross-validation ensured that the models were robust and generalizable by splitting the training data into multiple folds and validating the performance on each fold. This approach minimizes the risk of overfitting and ensures that the selected model can perform well on unseen data **(Adnan et al., 2022).**

The trained models were then evaluated on the validation set using various performance metrics, including accuracy, precision, recall, and F1-score, to assess their effectiveness. These metrics offer a comprehensive view of the model's performance, with accuracy measuring overall correctness, precision assessing the relevance of positive predictions, recall evaluating the model's ability to capture relevant instances, and F1-score balancing precision and recall **(Shiri et al., 2023b).** The best-performing models were subsequently tested on the test set to further evaluate their generalizability and ensure they could effectively analyse unseen data, which is crucial for real-world applications of sentiment analysis.

Application to Mental Health Dataset

After identifying the best-performing model through generic sentiment analysis on the "Twitter Entity Sentiment Analysis" dataset, the same model was retrained using the "Sentiment Analysis for Mental

Health" dataset. This step was crucial for adapting the model to the specific nuances and challenges of mental health-related text. Retraining the model with this specialized dataset allowed for fine-tuning its parameters and enhancing its sensitivity to the language and patterns unique to mental health discussions, such as expressions of anxiety, depression, and suicidal ideation.

The retraining process involved not only applying the model to the new dataset but also performing additional hyperparameter tuning to ensure that the model was optimized for this domain. This step was vital in refining the model's performance and ensuring that it could accurately classify sentiments within mental health-related content. The model's performance was then evaluated using the same metrics—accuracy, precision, recall, and F1-score—to ensure consistency and comparability with the results from the generic dataset.

By retraining the model on the "Sentiment Analysis for Mental Health" dataset, this study aimed to demonstrate the model's adaptability and reliability in real-world mental health monitoring scenarios. This approach provides valuable insights into how sentiment analysis techniques can be effectively applied in public health contexts, enabling better detection and intervention strategies for mental health issues based on social media data.

3.5. Evaluation Metrics

To comprehensively assess the performance of the machine learning and deep learning models employed in this study, a set of standard evaluation metrics was used. These metrics are essential for understanding how well each model performs in classifying sentiments, particularly in the context of both generic social media data and mental health-related data.

Accuracy: Accuracy is the proportion of correctly classified instances out of the total instances in the dataset. While it provides a general measure of model performance, accuracy alone may not be sufficient, especially in cases where class distributions are imbalanced, as is often the case in sentiment analysis. For instance, in the mental health dataset, where instances of certain sentiments like suicidal ideation may be rarer, accuracy might not fully capture the model's effectiveness **(Sharma et al., 2024).**

Precision: Precision is the ratio of true positive predictions to the total positive predictions made by the model. It measures the model’s ability to correctly identify positive instances, which is crucial in applications where false positives can lead to significant consequences, such as in mental health monitoring **(Boesch, 2024).** For example, incorrectly classifying neutral or non-risky content as indicative of depression could result in unnecessary interventions.

Recall (Sensitivity): Recall is the ratio of true positive predictions to the total actual positives in the dataset. This metric is particularly important in contexts like mental health sentiment analysis, where failing to identify a true positive (such as a sentiment indicative of depression or anxiety) could have serious implications. High recall is essential in ensuring that as many relevant instances as possible are captured by the model **(Islam et al., 2024).**

F1-Score: The F1-Score is the harmonic mean of precision and recall, providing a balanced measure of a model's performance when both precision and recall are important. It is particularly useful in scenarios where there is an uneven class distribution, which is often encountered in sentiment analysis tasks **(Kanstrén, 2023).**

Confusion Matrix: The confusion matrix provides a detailed breakdown of the model's predictions, showing the true positives, false positives, true negatives, and false negatives. This matrix is essential for diagnosing specific areas where the model may struggle, such as confusing positive sentiments with neutral or negative ones (Naidu et al., 2023).

These evaluation metrics were used consistently across both the generic and mental health-specific datasets to ensure comparability of results. By applying these metrics, the study was able to determine the strengths and weaknesses of each model, providing insights into their applicability for real-world sentiment analysis tasks, particularly in the domain of mental health monitoring and intervention.

4. Data Analysis

Exploratory Data Analysis (EDA) serves as a crucial step in understanding the underlying structure of the datasets used in this study. By analysing the distribution of data, the relationships between variables, and the patterns within the data, EDA provides valuable insights that inform the modelling process. This section delves into the key findings from the EDA conducted on both the "Twitter Entity Sentiment Analysis" and "Sentiment Analysis for Mental Health" datasets.

4.1. Distribution of Sentiment Classes:

The first part of the EDA concerned the distribution of sentiment classes within the datasets. Understanding the balance of classes is critical because it directly affects the model's capacity to learn properly. In the "Twitter Entity Sentiment Analysis" dataset, the sentiment classes Negative, Positive, Neutral, and Irrelevant were very evenly distributed, as illustrated in the Figure 1 below.

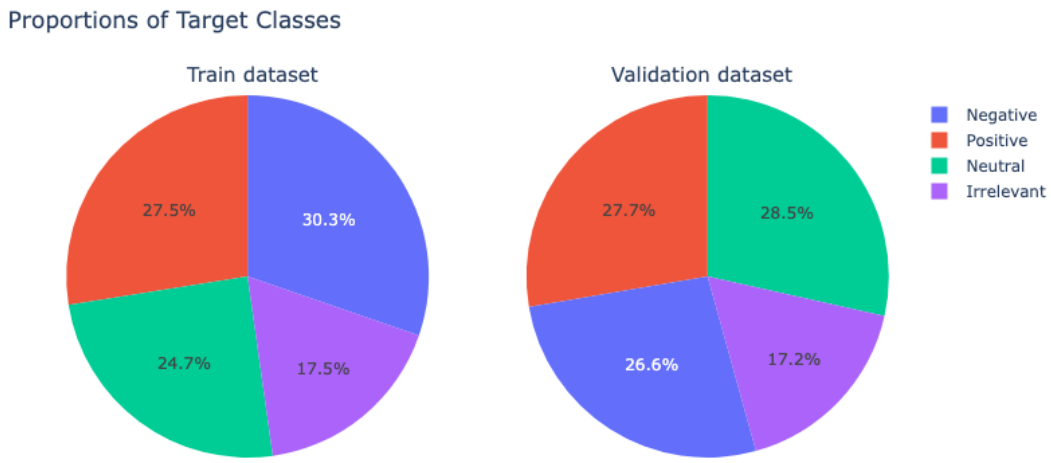


Figure 1: Proportion of Sentiment Classes of Twitter Data

In the training dataset, which consists of 74,682 data points, Negative sentiments comprised 30.3%, Positive sentiments 27.5%, Neutral sentiments 24.7%, and Irrelevant sentiments 17.5%. The validation dataset, with 1,000 data points, displayed a similar distribution with slight variations: Negative (28.5%), Positive (27.7%), Neutral (26.6%), and Irrelevant (17.2%).

The large amount of data used for training plays a crucial role in building a better model. A well-sized training dataset like this allows the model to learn a wide range of patterns, reducing the risk of overfitting and improving its generalization capabilities. With this substantial volume of training data, the model is more likely to capture the nuances of each sentiment class, ultimately leading to more accurate predictions. This relatively balanced distribution, combined with the extensive training data, indicates that the models trained on this dataset would likely perform well across all sentiment classes, minimizing bias toward any class.

For the "Sentiment Analysis for Mental Health" dataset, the sentiment labels were more specialized, reflecting various mental health conditions. The distribution revealed the following counts: 'Normal' sentiments were the most prevalent with 16,351 instances, followed by 'Depression' with 15,404 instances, 'Suicidal' with 10,653 instances, 'Anxiety' with 3,888 instances, 'Bipolar' with 2,877 instances, 'Stress' with 2,669 instances, and 'Personality Disorder' with 1,201 instances. This distribution is depicted in the bar chart below.

The training dataset, consisting of 53,043 data points, provides a robust foundation for the model to learn from these diverse and nuanced sentiment labels. The validation dataset, with 5,000 data points, serves as a critical checkpoint to evaluate the model's performance on unseen data.

Using such a substantial amount of training data is instrumental in building a more accurate and reliable model, particularly when dealing with complex and sensitive topics like mental health. The ample data allows the model to better understand the subtleties between different mental health-related sentiments, enhancing its ability to predict these conditions more effectively.

By ensuring that both the training and validation datasets are well-represented across all sentiment labels, the model can generalize well, ultimately improving its performance in real-world application.This distribution is illustrated in the following bar chart, demonstrating that the dataset covers a wide range of mental health-related sentiments.

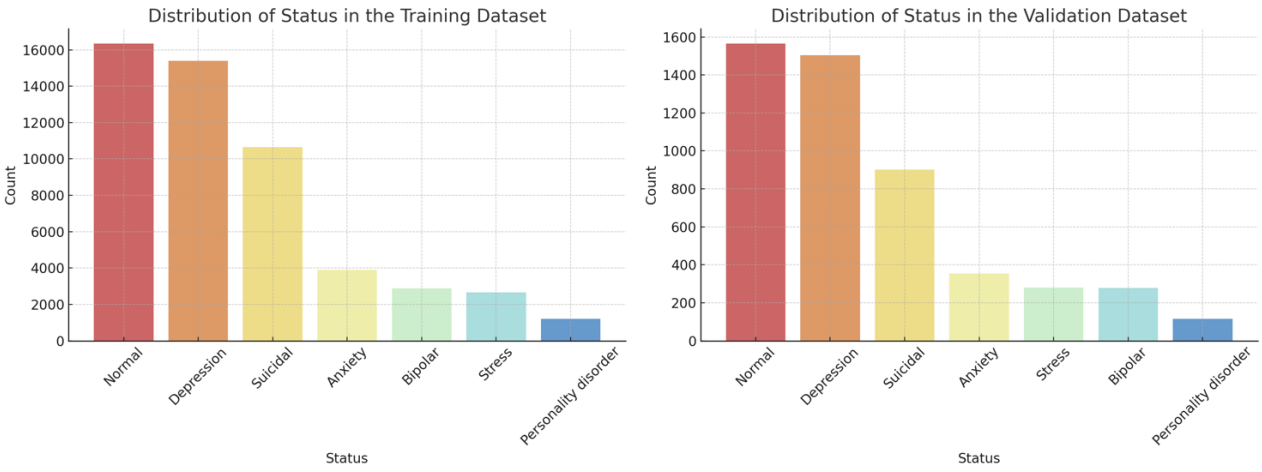


Figure 2: Distribution of Sentiment Classes of Mental Health Data

4.2. Examples of Sentiment Classes in Mental Health Data

To provide a clearer picture of the types of text associated with each sentiment label, here are some examples from the dataset:

- **Anxiety:**
  - "oh my gosh"
  - "trouble sleeping, confused mind, restless heart. All out of tune"
  - "All wrong, back off dear, forward doubt. Stay in a restless and restless place"
- **Normal:**
  - "Gr gr dreaming of ex crush to be my game, God"

    - "wkwkwk what a joke"
    - "Leaves are also standby in front of the PC ... because the office is no longer on leave"
- **Depression:**
    - "I recently went through a breakup and she said she still wants to be friends so I said I can try doing that but when she talks to me about things it always hurts..."
    - "I do not know how to navigate these feelings, not that it's a new feeling by any stretch..."
    - "I will never be good enough, I will always have skin problems and bad teeth and acne."
- **Suicidal:**
    - "I am so exhausted of this. Just when I think I can finally rest, just when I think maybe things are starting to settle, another hurdle comes flying at me..."
    - "I am 20 years old with some good friends, but I am just tired. I had a problem with bullying when I was little, and that hit me hard..."
    - "I do not like play hitting, smacking, striking, hitting or violence of any sort on my person. Do I send out this vibe asking for it from the universe?"
- **Stress:**
    - "It's been tough balancing work, family, and everything else. Sometimes it feels like too much, and I can't keep up with the expectations..."
    - "I keep thinking about everything I have to do, and the list never seems to end. It's overwhelming, and I don't know where to start."
    - "Every day feels like a struggle to keep it all together. I wish I could just take a break from everything and catch my breath."
- **Personality Disorder:**
    - "Is there anyone interested in joining a group for AvPD on Telegram?"
    - "Anyone else have nothing in common with other people? When I see normal people constantly talk and have fun with it, I'm genuinely shocked."

These examples give a snapshot of the type of language and expressions commonly associated with each mental health sentiment label.

4.3. Analysis of Text Length:

Another critical aspect of the EDA was analysing the distribution of text lengths across different sentiment classes. Text length can influence model performance, particularly in natural language processing tasks where longer texts may contain more contextual information but also more noise **(Wankhade et al., 2022).**

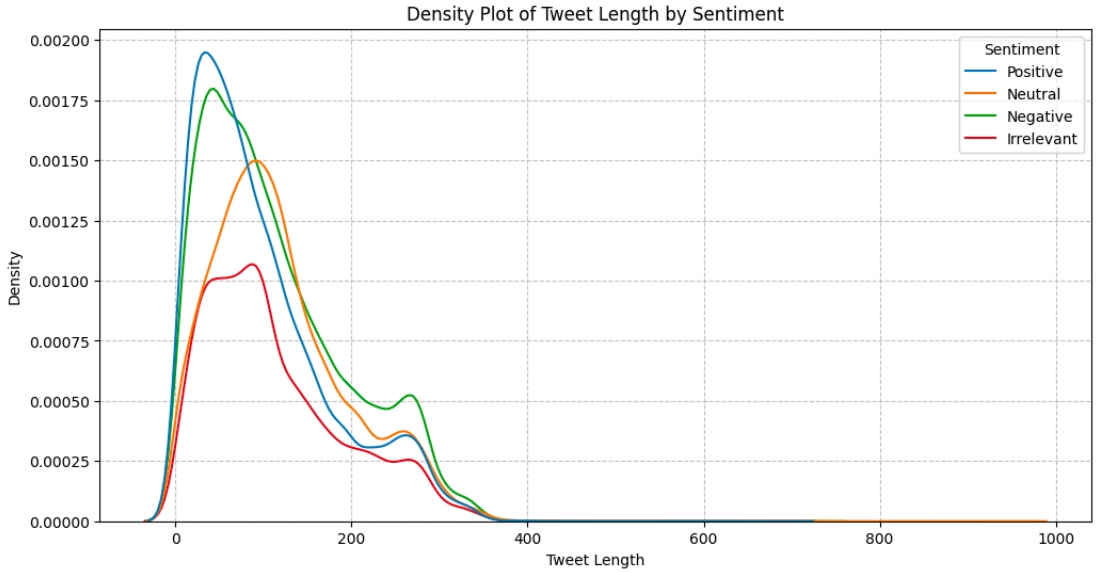


Figure 3: Density Plot of Tweet Length by Sentiment

As observed in Figure 3, 'Positive' and 'Neutral' tweets tend to be shorter, while 'Negative' and 'Irrelevant' tweets exhibit a wider range of lengths. This variation suggests that the models might need to account for the length of the text as a feature, particularly when dealing with longer tweets that might include more complex sentiment expressions.

In contrast, the "Sentiment Analysis for Mental Health" dataset exhibited a different pattern as shown in Figure 4. The distribution of statement lengths varied significantly across the different mental health-related sentiment labels. This pattern suggests that longer statements, particularly those associated with conditions like 'Suicidal,' 'Depression,' and 'Anxiety,' may carry more detailed and complex expressions of sentiment.

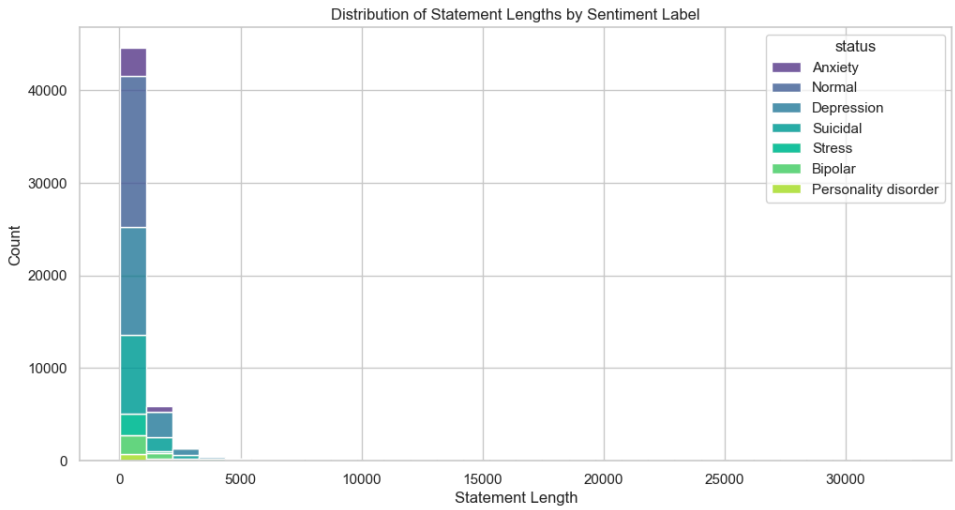


Figure 4: Density Plot of Tweet Length by Sentiment in Mental Health Data

This insight is critical as it may reflect the complexity and depth of expression in discussions related to mental health, suggesting that models may need to be particularly sensitive to the length and depth of content in these contexts.

5. Application of Methods

In this section, the practical application of the machine learning and deep learning techniques outlined in Section 3 is detailed, with a focus on how these methods were tailored to address the research problem using the selected datasets. The challenges encountered during this process and the strategies employed to overcome them are also discussed.

As discussed in Section 3, the primary techniques employed in this study include Logistic Regression, Random Forest, Naive Bayes, LSTM, GRU, and TextCNN. These models were selected for their effectiveness in sentiment analysis tasks, particularly in handling the complexities of social media text. Additionally, techniques such as Principal Component Analysis (PCA) and confusion matrices were utilized to refine and evaluate the models.

5.1. Applications

The methods outlined in Section 3 were systematically applied to both the "Twitter Entity Sentiment Analysis" and "Sentiment Analysis for Mental Health" datasets. The goal was to optimize the models for accurately detecting sentiments, with a particular focus on mental health-related sentiments.

Model Training

Training the models involved several key steps to ensure optimal performance:

- **Logistic Regression and Naive Bayes**: These models served as baselines for comparison. Logistic Regression involved tuning the regularization strength to balance complexity and performance. For Naive Bayes, the model was optimized to handle high-dimensional text data, common in sentiment analysis.
- **Random Forest and Extra Trees Classifier**: These ensemble models required extensive hyperparameter tuning. Parameters such as the number of trees, tree depth, and minimum samples per leaf were optimized to reduce overfitting and improve generalization.
- **LSTM, GRU, and CNN-Based Models**: Training the deep learning models involved selecting the learning rate, batch size, and number of epochs. LSTM and GRU models focused on capturing temporal dependencies, while TextCNN concentrated on extracting local features using convolutional layers.

Parameter Tuning

Parameter tuning was critical to optimize the models:

- **Grid Search and Cross-Validation**: We employed grid search to explore a range of hyperparameters for each model. Cross-validation ensured that the models performed well on unseen data, preventing overfitting.
- **Early Stopping and Regularization in Deep Learning Models**: To avoid overfitting, especially in deep learning models, early stopping was used during training. Regularization techniques, such as L2 regularization and dropout layers, were also implemented to enhance generalization.

5.2. Challenges and Mitigation Strategies

Throughout the application of these methods, several challenges were encountered:

- **Data Imbalance**: In the mental health dataset, class imbalance was a significant issue. Techniques such as oversampling and adjusting class weights were employed to address this.
- **Noisy Data**: Social media data often contains noise in the form of irrelevant information and inconsistent language. Advanced text-cleaning techniques were implemented to reduce this noise.
- **Computational Constraints**: Deep learning models required substantial computational resources. Cloud-based platforms were used to manage these constraints, and model complexity was optimized to maintain performance.

In summary, the methods discussed in Section 3 were effectively applied to the selected datasets. This application highlighted the strengths of each model and the challenges encountered during

implementation. The following section will provide a detailed analysis of the results obtained from these methods and their implications for the study.

6. Results

6.1. Model Performance

This section presents the performance of the various machine learning and deep learning models used in this study for sentiment analysis on both the "Twitter Entity Sentiment Analysis" dataset and the "Sentiment Analysis for Mental Health" dataset. The models were evaluated based on several performance metrics, including accuracy, precision, recall, and F1-score, across different sentiment classes. Below is a summary of the results obtained from the classification reports of each model.

Model	Accuracy	Precision	Recall	F1- Score
Logistic Regression	69.05%	0.69	0.67	0.68
Random Forest	89.86%	0.91	0.89	0.90
Naïve Bayes	63.46%	0.67	0.60	0.60
Extra Tree Classifier	92.55%	0.93	0.92	0.92

Table 1: Machine Learning Models performance

Machine Learning Models:

- **Logistic Regression:** The Logistic Regression model achieved an overall accuracy of 69.05%. While the model performed reasonably well with balanced precision and recall across all sentiment classes, it struggled with the 'Irrelevant' class, where the F1-score was significantly lower compared to other classes. This indicates that Logistic Regression may have limitations in capturing the nuances of less defined or ambiguous sentiment categories.
- **Random Forest:** The Random Forest model significantly outperformed Logistic Regression with an accuracy of 89.86%. The model demonstrated high precision, recall, and F1-score across all classes, particularly excelling in the 'Negative' and 'Positive' classes. However, while the 'Irrelevant' class showed improvement, its performance was still slightly lower compared to

other classes, suggesting that even robust models like Random Forest might struggle with less frequent or ambiguous sentiments.

- **Naive Bayes:** Known for its simplicity, Naive Bayes yielded an overall accuracy of 63.46%. The model showed a marked drop in performance, especially in the 'Irrelevant' and 'Neutral' classes, where the F1-scores were notably lower. This reflects the model's limitations in handling the complexity of sentiment classification compared to more sophisticated models.
- **Extra Trees Classifier:** The Extra Trees Classifier emerged as the top-performing machine learning model with an impressive accuracy of 92.55%. It maintained high precision, recall, and F1-scores across all sentiment classes, demonstrating robustness and reliability in sentiment classification tasks. This performance suggests that Extra Trees' ability to handle large datasets and high-dimensional data gives it an edge over other models.

Model	Accuracy	Estimated Precision	Estimated Recall	Estimated F1 Score
LSTM	84.09%	84.09%	84.09%	84.09%
RNN	78.18%	78.18%	78.18%	78.18%
GRU	78.81%	78.81%	78.81%	78.81%
TextCNN	85.95%	85.95%	85.95%	85.95%
CNN-LSTM Hybrid	84.81%	84.81%	84.81%	84.81%
CNN-GRU Hybrid	84.07%	84.07%	84.07%	84.07%

Table 2: Deep Learning Models performance

Deep Learning Models:

- **LSTM:** The Long Short-Term Memory (LSTM) model achieved an accuracy of 84.09% on the test data. LSTM's ability to capture long-term dependencies in sequential data was evident, as the model performed well across all classes, with strength in the 'Positive' and 'Negative' classes. This model's ability to retain context over longer sequences made it particularly effective in capturing nuanced sentiment shifts within the text.
- **RNN:** The Recurrent Neural Network (RNN) model, although simpler than LSTM, achieved a test accuracy of 78.18%. The performance, while respectable, was lower than that of the LSTM, highlighting the importance of advanced architectures like LSTM in capturing complex patterns in sentiment analysis. RNNs struggled more with maintaining performance across longer sequences, which is where LSTMs excel.

- **GRU:** The Gated Recurrent Unit (GRU) model showed an accuracy of 78.81%, slightly outperforming the RNN but still falling short of the LSTM. GRU's simpler architecture allowed for faster training while still maintaining decent performance. The model's ability to balance efficiency and performance makes it a viable alternative when computational resources are limited.
- **TextCNN:** The Text Convolutional Neural Network (TextCNN) achieved an accuracy of 85.95%, making it one of the top-performing models. The model's ability to capture local features and n-grams in the text contributed to its strong performance, particularly in the 'Negative' and 'Positive' classes. TextCNN's effectiveness in detecting key phrases and local patterns helped it excel in sentiment classification tasks.
- **CNN-LSTM Hybrid:** The CNN-LSTM hybrid model combined the strengths of both CNN and LSTM architectures, resulting in an accuracy of 84.81%. This hybrid approach allowed the model to capture both local and global patterns in the text, leading to balanced performance across all sentiment classes. The combination of convolutional layers with LSTM's sequential processing capabilities provided a comprehensive approach to sentiment analysis.
- **CNN-GRU Hybrid:** The CNN-GRU hybrid model achieved an accuracy of 84.07%, similar to the CNN-LSTM hybrid. While the GRU component provided faster training, the performance was slightly lower than the CNN-LSTM hybrid, highlighting the trade-offs between speed and accuracy in model selection.

The comparison of model performances reveals that both traditional machine learning models and deep learning models offer valuable strengths in sentiment analysis tasks, each excelling in different areas. Random Forest and Extra Trees Classifier, notable for their high accuracy and balanced performance, effectively handle complex interactions in sentiment data, making them reliable choices, especially in ensemble methods. However, while simpler models like Logistic Regression and Naive Bayes are interpretable and efficient, they struggle with the nuanced sentiment distinctions typical in social media data, especially in multi-class scenarios. This underscores the need for more sophisticated models when dealing with complex datasets.

In the realm of deep learning, models such as LSTM, GRU, and CNN-based architectures demonstrate robust capabilities in capturing both local features and long-term dependencies within text, essential for tasks like sentiment analysis in mental health data. LSTM, with its strong performance in tracking sentiment evolution, and hybrid models like CNN-LSTM, which balance local and global feature extraction, emerge as powerful tools for complex sentiment analysis. TextCNN, meanwhile, shines in handling short, impactful social media texts. Ultimately, the choice of model should align with the specific requirements of the task, whether it be accuracy, interpretability, or computational efficiency.

After implementing GridSearchCV to fine-tune the hyperparameters of various models, the Extra Trees Classifier emerged as the best-performing model, achieving an impressive accuracy of 92.55%. This optimization process further reinforced the model's robustness, confirming its ability to handle the variability in sentiment data effectively. The fine-tuning allowed the model to better capture the complex patterns in the data, making it particularly suited for sentiment analysis tasks, especially in

challenging domains like social media and mental health data. This result highlights the importance of not only selecting the right model but also optimizing it to maximize performance.

6.2. Interpretation in the Context of Mental Health Data:

After finding the best model for sentiment analysis, the Extra Trees Classifier was selected and trained with new data to enhance its performance further. In the context of mental health data, this model proved to be particularly effective due to its ability to capture the complexities of sentiment and emotional states reflected in social media content. By fine-tuning the Extra Trees Classifier with grid search and cross-validation techniques, the model was optimized to handle the nuanced and often overlapping emotional categories found in discussions related to mental health. This rigorous approach ensured that the classifier was well-suited to accurately classify sentiments in this sensitive and complex domain.

The results from the Extra Trees Classifier on the "Sentiment Analysis for Mental Health" dataset were particularly impressive, achieving an overall accuracy of **96%**. The detailed performance metrics in Table 3, provide further insights into how the model performed across various sentiment classes:

Sentiment Class	Precision	Recall	F1- Score	Support
Anxiety	0.98	0.99	0.99	348
Bipolar	0.99	0.99	0.99	331
Depression	0.94	0.87	0.90	354
Normal	0.95	0.97	0.96	329
Personality Disorder	1.0	1.0	1.0	324
Stress	1.0	0.99	0.99	317
Suicidal	0.87	0.92	0.89	314

Table 3: Models performance of Mental Health Data

The performance of the Extra Trees Classifier on the "Sentiment Analysis for Mental Health" dataset shows strong results across various sentiment classes. The model achieved high precision, recall, and F1-scores in most classes. For *Anxiety*, the precision and recall were 0.98 and 0.99, respectively, leading to a high F1-score of 0.99. Similarly, *Bipolar* classification was near-perfect, with all metrics at 0.99. The *Depression* class, while slightly lower, still showed strong results with an F1-score of 0.90, indicating some challenges due to overlapping symptoms with other conditions. The model performed well in the *Normal* class, with a precision of 0.95 and recall of 0.97, resulting in an F1-score of 0.96.

Personality Disorder and *Stress* classifications were flawless, with all metrics at 1.00 and 0.99, respectively, highlighting the model's accuracy. However, the *Suicidal* class, while performing well with an F1-score of 0.89, had a slightly lower precision of 0.87, suggesting some difficulty in distinguishing this class with absolute accuracy. Overall, the model demonstrates robust performance, particularly in classes with clearer distinctions.

However, the slight dip in performance for the *Suicidal* class suggests specific challenges in accurately identifying suicidal ideation. One potential reason for this lower performance is the complexity and subtlety of language associated with suicidal thoughts. Unlike other mental health conditions where symptoms may be more directly expressed, suicidal ideation is often communicated indirectly or through coded language, making it more difficult for the model to detect. For example, individuals may express feelings of hopelessness or detachment without explicitly mentioning suicidal thoughts, leading to misclassification. This becomes more evident when examining the confusion matrix, which provides a detailed breakdown of how different sentiment classes were predicted, highlighting potential areas of confusion between similar or overlapping sentiments.

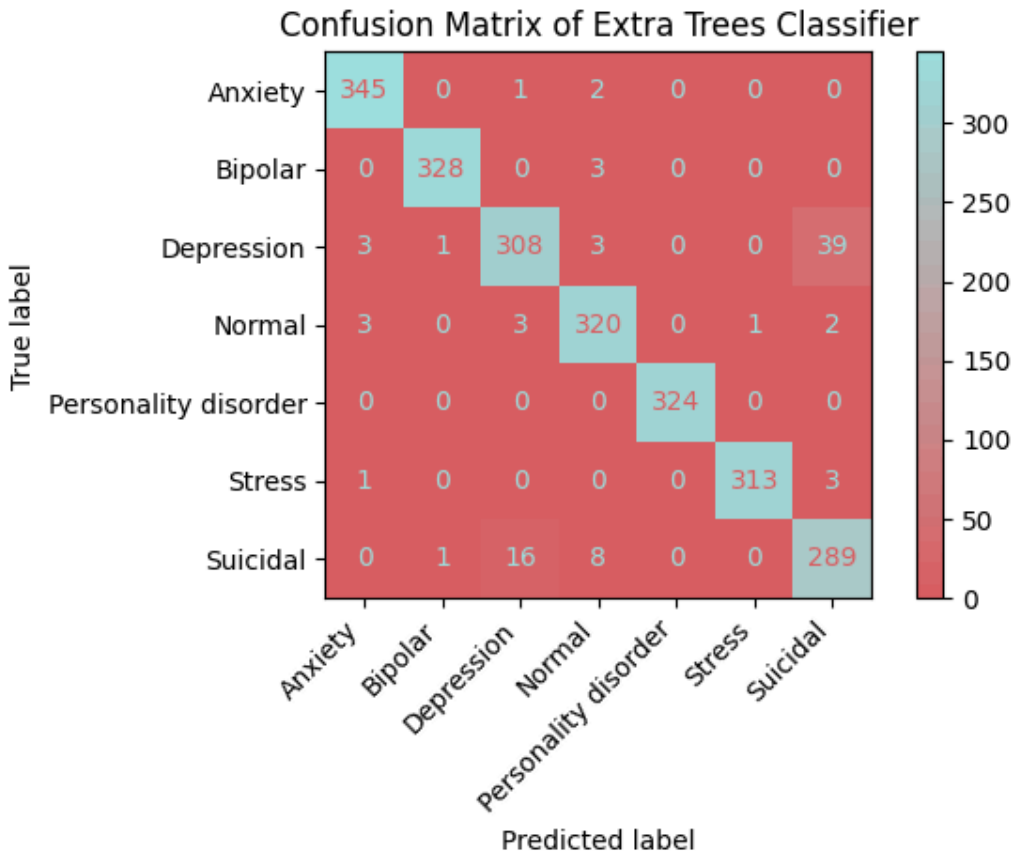


Figure 5: Confusion Metrix Plot of Mental Health Data

The confusion matrix in Figure 5, reveals that the Extra Trees Classifier struggles with accurately predicting instances of the *Suicidal* class, often misclassifying them as *Depression* or *Stress*. This confusion is evident in specific examples from the dataset. For instance, a statement with the true label *Depression* was predicted as *Suicidal*: "Clearly on my mind more than normal. do not really have any motivation to keep on going. Boyfriend mentioned I joke about killing myself a lot recently." Here, the mention of joking about suicide might have led the model to categorize the statement as *Suicidal*, despite the context suggesting a broader depressive state.

Conversely, a statement labelled as *Suicidal* was misclassified as *Depression*: "A couple weeks ago now was my latest attempt to end my own life. For years my mental health was stabilizing, getting better each and every year. Then this year with the pandemic, relapsing depression, my girlfriend breaking up with, and a new stressful job has led me down a very dark rabbit hole that has gotten out of control. Then I finally tried taking my life again." In this case, the detailed context of the individual's history with depression and stress might have influenced the model to predict *Depression* rather than recognizing the suicidal ideation explicitly mentioned.

These examples underscore the difficulty in differentiating between closely related emotional states. The model's tendency to misclassify suicidal ideation highlights a significant challenge in sentiment analysis for mental health: the need to accurately capture the subtle nuances of language that can indicate a shift from depression or stress to suicidal thoughts. This emphasizes the importance of refining the model to incorporate more specific linguistic features or a deeper contextual understanding that can better capture the nuances of suicidal ideation.

Despite this challenge, the model's overall performance across multiple classes remains impressive. Its ability to handle a wide range of emotional categories, such as *Anxiety*, *Bipolar Disorder*, and *Stress*, speaks to its robustness and versatility. This capability is crucial in real-world applications where accurately reflecting the diverse ways individuals express their mental health states can make a significant difference. The consistent performance across various classes demonstrates the model's generalizability, making it a strong candidate for real-world applications in mental health monitoring.

These findings are particularly relevant for early detection and intervention strategies in mental health. Accurately distinguishing between different mental health conditions enables timely and actionable insights for mental health professionals. For instance, identifying high-risk individuals, especially those expressing suicidal thoughts, is critical for preventing crises and providing the necessary support. However, as the confusion matrix and misclassified examples indicate, the model's challenges in classifying suicidal ideation emphasize the need for further refinement. Future research could focus on enhancing the model's sensitivity to the subtle language cues associated with suicide, potentially by incorporating advanced NLP techniques like context-aware embeddings or transformer models. Additionally, integrating domain-specific knowledge from psychology and psychiatry into the model's feature set could further improve its ability to distinguish between closely related conditions.

In summary, while the Extra Trees Classifier shows great promise for sentiment analysis in mental health, its application in sensitive areas such as suicide prevention requires continuous improvement. Ensuring that these models are accurate, interpretable, and reliable in real-world settings is crucial for their successful deployment in mental health monitoring and intervention systems.

6.3. Challenges and Considerations

Despite the strong overall performance of the models in this study, several challenges and considerations must be acknowledged:

1. **Underperformance of Simpler Models:** Simpler models like Logistic Regression and Naive Bayes showed lower accuracy, reflecting their limitations in complex sentiment analysis tasks, particularly in the mental health domain. These models struggle to capture subtle linguistic cues and the nuanced emotional states that are essential for accurate sentiment detection. This underperformance underscores the necessity for more advanced models capable of understanding the intricacies of human emotions expressed through text.
2. **High Computational Demands of Deep Learning Models:** Although deep learning models, such as LSTM, GRU, and CNN, deliver superior accuracy, they come with substantial computational requirements. These models demand significant processing power and extended training times, which may pose challenges in resource-constrained environments or scenarios requiring rapid response times. For instance, the CNN-LSTM hybrid model, despite its effectiveness, requires considerable computational resources, making it less practical for real-time applications.
3. **Challenges in Interpretability:** The interpretability of complex models is another critical consideration, especially in sensitive areas like mental health. Deep learning models, often regarded as "black boxes," can obscure the reasoning behind their predictions, which may hinder their acceptance in clinical settings. In contrast to simpler models like Logistic Regression, where decision-making is more transparent, deep learning architectures can be challenging to interpret, making it difficult for stakeholders to fully trust and understand the model's outputs.
4. **Generalization and Overfitting Risks:** Advanced models, while performing well on training data, may still face the risk of overfitting, particularly in high-dimensional sentiment analysis tasks. Although techniques like cross-validation and grid search help mitigate overfitting, ensuring that these models generalize effectively to new, unseen data remains a crucial concern. For practical deployment, models must maintain their performance across diverse real-world scenarios.

The models developed in this study show significant promise for enhancing sentiment analysis in mental health contexts. However, these challenges highlight the need for continued refinement, particularly in balancing accuracy, interpretability, and computational efficiency. Future efforts should focus on addressing these limitations to develop more accessible, trustworthy, and practical models.

6.4. Implications for Future Research and Applications

The findings from this analysis offer valuable implications for both future research and practical applications in sentiment analysis and mental health monitoring.

1. **Advancing Sentiment Analysis Techniques:** The strong performance of ensemble methods and deep learning models highlights the potential for further advancements in these areas. Future research should explore the integration of cutting-edge techniques, such as attention mechanisms and transformer models like BERT or GPT, which have demonstrated exceptional performance in other NLP tasks. These advanced models could significantly enhance the ability to capture complex linguistic patterns and context-specific nuances, especially in the sensitive domain of mental health.
2. **Developing Specialized Models for Mental Health Data:** Mental health data requires models that can accurately detect subtle and evolving emotional states. Future work should focus on creating specialized models that are trained on datasets reflecting the diverse ways individuals express their mental health experiences online. Emphasizing domain-specific training could improve the detection of emotions such as depression, anxiety, and suicidal ideation, which are often conveyed in indirect or nuanced language.
3. **Expanding Real-Time Monitoring and Applications:** Extending these models to real-time monitoring systems presents significant opportunities. Such systems could analyze social media data to detect early signs of mental health issues, providing crucial support for public health initiatives. By offering timely insights into the mental well-being of populations, these applications could enable more proactive and targeted interventions. Additionally, the ability to monitor shifts in public sentiment, particularly during crises, could lead to better decision-making and resource allocation within mental health services.

These insights emphasize the importance of selecting the right model based on the specific needs of the sentiment analysis task. Whether in general social media analysis or specialized mental health domains, balancing factors like accuracy, interpretability, and computational efficiency will be key to achieving success in future applications.

7. Discussion

7.1. Comparison with Previous Work

In comparing the findings of this study with previous research, several key similarities and differences emerge that contribute to the existing body of knowledge in sentiment analysis, particularly within the domain of mental health monitoring.

Ensemble methods such as Random Forest and Extra Trees Classifier have consistently been shown to outperform simpler models in previous studies on sentiment analysis. This study reaffirms these findings, demonstrating that these models excel at capturing complex feature interactions, which are essential in accurately classifying sentiments from diverse datasets. However, this research also extends prior work by applying these models specifically to mental health-related data, showing that ensemble methods can handle the nuances of this specialized domain with high accuracy.

Similarly, the use of deep learning models, particularly LSTM, GRU, and CNN architectures, has been well-documented in prior research. These models' ability to capture long-term dependencies and local features in text has been validated in numerous studies focusing on general sentiment analysis tasks. This study builds on that foundation by applying these models to mental health data, where sentiments often evolve throughout a statement. The results confirm that LSTM and CNN-based models are effective in this context, but they also highlight the challenges posed by the computational demands and interpretability of these models.

What sets this work apart is its focus on mental health sentiment analysis, a relatively underexplored area compared to other domains like product reviews or political opinion analysis. By demonstrating the effectiveness of advanced models in this context, this study adds valuable insights into how sentiment analysis can be specialized for use in mental health monitoring, thereby contributing to both the fields of sentiment analysis and public health.

7.2. Limitations

While this study offers promising results, several limitations must be acknowledged. One significant limitation is the quality and representativeness of the data. Although the datasets used in this study were carefully selected, they may not fully capture the diversity of language and expression in mental health discussions. Social media data can be noisy and unstructured, which can introduce challenges in model training and validation. Additionally, the reliance on labelled datasets means that the models are only as good as the annotations, which may not always accurately reflect the true sentiment or emotional state of the individuals.

Another limitation is related to the assumptions inherent in the models used. For instance, machine learning models like Logistic Regression and Naive Bayes assume independence between features, which is often not the case in natural language. While more complex models like LSTM and CNN address some of these issues, they introduce other challenges, such as overfitting, especially when dealing with high-dimensional data. Despite using techniques like cross-validation and regularization to mitigate these risks, the potential for overfitting remains a concern.

Computational constraints also played a role in this study. Deep learning models, while powerful, require substantial computational resources and long training times. This limitation can affect the scalability of these models, particularly in real-time applications where quick processing is essential. Additionally, the "black-box" nature of deep learning models presents challenges in interpretability,

which is critical in fields like mental health where understanding the reasoning behind a model's prediction is as important as the prediction itself.

Finally, this study primarily focused on English-language data, limiting its generalizability to non-English speaking populations. Future work should consider multilingual datasets to ensure that the models are applicable across diverse linguistic and cultural contexts.

7.3. Future Work

The findings and limitations of this study suggest several avenues for future research and practical applications in sentiment analysis and mental health monitoring.

One promising direction is the integration of more advanced models, such as transformers and attention mechanisms. Models like BERT and GPT have revolutionized NLP by enabling better context understanding and capturing subtle nuances in text. Applying these models to mental health data could further enhance the accuracy and reliability of sentiment classification, particularly in detecting complex emotional states. Future studies should explore the potential of these models, possibly in combination with existing approaches like LSTM and CNN, to create hybrid models that leverage the strengths of each architecture.

Additionally, the development of more specialized models tailored to mental health data is crucial. Given the unique challenges associated with this domain, such as the need for detecting subtle and evolving sentiments, future research should focus on creating models that are explicitly designed for these tasks. This could involve training models on larger and more diverse datasets that better represent the range of expressions found in mental health discussions, including those in non-English languages. Incorporating psychological and linguistic theories into the model development process could also help improve the accuracy and interpretability of the models.

Another important area for future research is improving the interpretability of deep learning models. As mentioned earlier, the "black-box" nature of these models can be a barrier to their adoption in sensitive areas like mental health. Techniques such as explainable AI (XAI) can help make these models more transparent, allowing stakeholders to understand and trust the predictions made by the models. Future work should explore the application of XAI methods to sentiment analysis, particularly in contexts where interpretability is critical.

Finally, the practical application of these models in real-time monitoring systems offers significant potential. Integrating sentiment analysis models into platforms that continuously monitor social media for signs of mental health issues could provide valuable insights for public health initiatives. Future research should focus on developing and testing such systems, ensuring they are scalable, reliable, and able to operate in real-time. This could involve collaboration between computer

scientists, psychologists, and public health professionals to ensure that the systems are both technically sound and ethically responsible.

While this study contributes valuable insights into sentiment analysis and mental health monitoring, it also opens several avenues for future research. By continuing to refine models, improve interpretability, and explore real-world applications, future work can build on these findings to create more effective and impactful tools for understanding and improving mental health.

8. Conclusion

Summary of Findings

This study explored the application of various machine learning and deep learning models to sentiment analysis, with a specific focus on mental health-related data. The key findings reveal that ensemble methods, such as Random Forest and Extra Trees Classifier, consistently outperformed simpler models like Logistic Regression and Naive Bayes, particularly in capturing complex feature interactions. Among these, the Extra Trees Classifier emerged as the best-performing model, demonstrating exceptional accuracy and reliability across both general sentiment analysis and mental health-specific applications. Its ability to handle large datasets and high-dimensional features made it particularly effective in distinguishing nuanced sentiments related to mental health conditions.

Deep learning models, including LSTM, GRU, and CNN architectures, also showed strong performance, especially in handling the sequential and local patterns in text data. The hybrid models combining CNN with LSTM or GRU further enhanced the ability to capture both local and global text patterns, although this improvement came at the cost of increased computational demand and reduced interpretability. These deep learning models proved valuable in detecting subtle emotional cues, which is crucial for real-time monitoring and intervention systems in mental health.

The study's focus on mental health data highlights the potential of these advanced models in providing timely insights into emotional states and mental health conditions. The Extra Trees Classifier demonstrated robust performance in classifying various mental health-related sentiments, making it a strong candidate for applications in mental health monitoring. Its high accuracy and ability to handle the complexity of mental health data underscore its suitability for real-world deployment in detecting and addressing mental health issues.

Overall, the findings contribute to the growing body of knowledge in sentiment analysis by validating the effectiveness of advanced models, especially the Extra Trees Classifier, in specialized domains like mental health. This research underscores the importance of continued development in sentiment analysis techniques to enhance model interpretability and computational efficiency, ultimately improving their applicability and effectiveness in real-world mental health contexts.

Practical Applications

The practical applications of this research extend across multiple industries, including marketing, customer service, public opinion analysis, and, most notably, mental health monitoring. In marketing, the ability to accurately gauge consumer sentiment can guide product development and marketing strategies by identifying shifts in public opinion or customer satisfaction. Similarly, in customer service, sentiment analysis tools can help companies monitor and respond to customer feedback in real-time, improving service quality and customer satisfaction.

In the context of public opinion analysis, sentiment analysis models can be applied to social media data to track trends and sentiments related to political events, social movements, or public health crises. This application is particularly relevant for governments and organizations seeking to understand and respond to public concerns more effectively.

The most significant impact, however, lies in the field of mental health monitoring. By integrating sentiment analysis models into real-time monitoring systems, mental health professionals can identify early signs of distress in individuals or populations, enabling timely interventions and support. This research paves the way for more effective tools in mental health care, contributing to improved outcomes and better public health strategies.

9. Citation

1. *A comprehensive review on sentiment analysis: Tasks, approaches and applications*. (n.d.). Ar5iv. https://ar5iv.labs.arxiv.org/html/2311.11250
2. Abidin, Z., Junaidi, A., & Wamiliana, N. (2024). Text Stemming and lemmatization of Regional Languages in Indonesia: a systematic literature review. *Journal of Information Systems Engineering and Business Intelligence*, *10*(2), 217–231. https://doi.org/10.20473/jisebi.10.2.217-231
3. Adnan, M., Alarood, A. a. S., Uddin, M. I., & Rehman, I. U. (2022). Utilizing grid search cross-validation with adaptive boosting for augmenting performance of machine learning models. *PeerJ Computer Science*, *8*, e803. https://doi.org/10.7717/peerj-cs.803
4. Bayat, S., & Işik, G. (2023). Evaluating the effectiveness of different machine learning approaches for sentiment classification. *Iğdır Üniversitesi Fen Bilimleri Enstitüsü Dergisi*, *13*(3), 1496–1510. https://doi.org/10.21597/jist.1292050
5. Boesch, G. (2024, June 19). *Precision vs. Recall – Full Guide to Understanding Model Output*. viso.ai. https://viso.ai/computer-vision/precision-recall/

6. Boonyarat, P., Liew, D. J., & Chang, Y. (2024). Leveraging enhanced BERT models for detecting suicidal ideation in Thai social media content amidst COVID-19. *Information Processing & Management*, *61*(4), 103706. https://doi.org/10.1016/j.ipm.2024.103706
7. Bordoloi, M., & Biswas, S. K. (2023). Sentiment analysis: A survey on design framework, applications and future scopes. *Artificial Intelligence Review*, *56*(11), 12505–12560. https://doi.org/10.1007/s10462-023-10442-2
8. De Freitas Santos, L., & Varges, D. S. M. (2023). The effect of stemming and lemmatization on Portuguese fake news text classification. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2310.11344
9. Elmitwalli, S., & Mehegan, J. (2024). Sentiment analysis of COP9-related tweets: a comparative study of pre-trained models and traditional techniques. *Frontiers in Big Data*, *7*. https://doi.org/10.3389/fdata.2024.1357926
10. García-Hernández, R. A., Luna-García, H., Celaya-Padilla, J. M., García-Hernández, A., Reveles-Gómez, L. C., Flores-Chaires, L. A., Delgado-Contreras, J. R., Rondon, D., & Villalba-Condori, K. O. (2024). A systematic literature review of modalities, trends, and limitations in emotion recognition, affective computing, and sentiment analysis. *Applied Sciences*, *14*(16), 7165. https://doi.org/10.3390/app14167165
11. Gunasekaran, K. P. (2023). Exploring Sentiment Analysis Techniques in Natural Language Processing: A Comprehensive review. *arXiv.org*. https://doi.org/10.17148/IJARCCE.2019.8126
12. Islam, M. S., Kabir, M. N., Ghani, N. A., Zamli, K. Z., Zulkifli, N. S. A., Rahman, M. M., & Moni, M. A. (2024). "Challenges and future in deep learning for sentiment analysis: a comprehensive review and a proposed novel hybrid approach." *Artificial Intelligence Review*, *57*(3). https://doi.org/10.1007/s10462-023-10651-9
13. Jim, J. R., Talukder, M. a. R., Malakar, P., Kabir, M. M., Nur, K., & Mridha, M. (2024). Recent advancements and challenges of NLP-based sentiment analysis: A state-of-the-art review. *Natural Language Processing Journal*, 100059. https://doi.org/10.1016/j.nlp.2024.100059
14. Kanstrén, T. (2023, September 27). A look at precision, recall, and F1-Score - towards data science. *Medium*. https://towardsdatascience.com/a-look-at-precision-recall-and-f1-score-36b5fd0dd3ec/
15. Keles, B., McCrae, N., & Grealish, A. (2019). A systematic review: the influence of social media on depression, anxiety and psychological distress in adolescents. *International Journal of Adolescence and Youth*, *25*(1), 79–93. https://doi.org/10.1080/02673843.2019.1590851
16. Kleppen, E. (2023, March 3). *What is sentiment analysis?* Built In. https://builtin.com/machine-learning/sentiment-analysis
17. Li, J. (2024). Advances in sentiment Analysis - techniques, applications, and challenges. In *Artificial intelligence*. https://doi.org/10.5772/intechopen.111293

18. Malik, N., & Jain, S. (2020). Comparative Study of Machine Learning Algorithms for Social Media text Analysis. In *Communications in computer and information science* (pp. 223–235). https://doi.org/10.1007/978-981-15-5830-6_19
19. Nadkarni, P. M., Ohno-Machado, L., & Chapman, W. W. (2011). Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, *18*(5), 544–551. https://doi.org/10.1136/amiajnl-2011-000464
20. Olaoye, F., & Potter, K. (2024). Natural language processing and sentiment analysis. *ResearchGate*. https://www.researchgate.net/publication/379025140_Natural_Language_Processing_and_Sentiment_Analysis
21. Saha, K., Torous, J., Ernala, S. K., Rizuto, C., Stafford, A., & De Choudhury, M. (2019). A computational study of mental health awareness campaigns on social media. *Translational Behavioral Medicine*, *9*(6), 1197–1207. https://doi.org/10.1093/tbm/ibz028
22. Sarker, I. H. (2021). Machine learning: algorithms, Real-World applications and research directions. *SN Computer Science*, *2*(3). https://doi.org/10.1007/s42979-021-00592-x
23. *Sentiment analysis and opinion mining*. (n.d.). Google Books. https://books.google.co.uk/books?hl=en&lr=&id=xYhyEAAAQBAJ&oi=fnd&pg=PP1&dq=43+Opinion+Mining+and+Sentiment+Analysis&ots=rlTxOCN6Ft&sig=AL7Vm0GsXJSqRh7rLe0NOq_-XjA#v=onepage&q=43%20Opinion%20Mining%20and%20Sentiment%20Analysis&f=false
24. Shanmugavadivel, K., Sathishkumar, V. E., Raja, S., Lingaiah, T. B., Neelakandan, S., & Subramanian, M. (2022). Deep learning based sentiment analysis and offensive language identification on multilingual code-mixed data. *Scientific Reports*, *12*(1). https://doi.org/10.1038/s41598-022-26092-3
25. Sharma, N. A., Ali, A. B. M. S., & Kabir, M. A. (2024). A review of sentiment analysis: tasks, applications, and deep learning techniques. *International Journal of Data Science and Analytics*. https://doi.org/10.1007/s41060-024-00594-x
26. Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023, May 27). *A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU*. arXiv.org. https://arxiv.org/abs/2305.17473
27. Shiri, F. M., Perumal, T., Mustapha, N., & Mohamed, R. (2023b, May 27). *A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU*. arXiv.org. https://arxiv.org/abs/2305.17473
28. Siino, M., Tinnirello, I., & La Cascia, M. (2024). Is text preprocessing still worth the time? A comparative survey on the influence of popular preprocessing methods on Transformers and traditional classifiers. *Information Systems*, *121*, 102342. https://doi.org/10.1016/j.is.2023.102342

29. Tan, K. L., Lee, C. P., & Lim, K. M. (2023b). A survey of sentiment analysis: Approaches, datasets, and future research. *Applied Sciences*, *13*(7), 4550. https://doi.org/10.3390/app13074550
30. Xu, Q. A., Chang, V., & Jayne, C. (2022). A systematic review of social media-based sentiment analysis: Emerging trends and challenges. *Decision Analytics Journal*, *3*, 100073. https://doi.org/10.1016/j.dajour.2022.100073
31. Xu, Q. A., Chang, V., & Jayne, C. (2022b). A systematic review of social media-based sentiment analysis: Emerging trends and challenges. *Decision Analytics Journal*, *3*, 100073. https://doi.org/10.1016/j.dajour.2022.100073
32. Yadav, C., & Gayen, T. (2023). Stopwords Aware Emotion-Based sentiment analysis of news articles. In *EAI/Springer Innovations in Communication and Computing* (pp. 183–193). https://doi.org/10.1007/978-3-031-28324-6_15
33. Zhang, T., Yang, K., Ji, S., & Ananiadou, S. (2023b). Emotion fusion for mental illness detection from social media: A survey. *Information Fusion*, *92*, 231–246. https://doi.org/10.1016/j.inffus.2022.11.031

10. Appendix (Code)

1 - Importing Libraries

```
import pandas as pd import
numpy as np import
seaborn as sns
import matplotlib.pyplot as plt import
plotly.graph_objects as go
from plotly.subplots import make_subplots from
wordcloud import WordCloud
import matplotlib.pyplot as plt
```

2 - Loading Data

```
column_names=['Tweet_ID','Entity','Sentiment','Tweet_content']

df_train = pd.read_csv("twitter_training.csv",
                       sep=',',names=column_names)


df_val = pd.read_csv("twitter_validation.csv",
                       sep=',',names=column_names)

df_train.head()
```

	Tweet_ID	Entity	Sentiment	\
0	2401	Borderlands	Positive	
1	2401	Borderlands	Positive	
2	2401	Borderlands	Positive	
3	2401	Borderlands	Positive	
4	2401	Borderlands	Positive	

	Tweet_content
0	im getting on borderlands and i will murder yo...
1	I am coming to the borders and I will kill you...
2	im getting on borderlands and i will kill you ...
3	im coming on borderlands and i will murder you...
4	im getting on borderlands 2 and i will murder

```
df_val.head()
```

	Tweet_ID	Entity	Sentiment	\
0	3364	Facebook	Irrelevant	
1	352	Amazon	Neutral	
2	8312	Microsoft	Negative	
3	4371	CS-GO	Negative	
4	4433	Google	Neutral	

```
                                                Tweet_content
0   I mentioned on Facebook that I was struggling ...
1   BBC News - Amazon boss Jeff Bezos rejects clai...
2   @Microsoft Why do I pay for WORD when it funct...
3   CSGO matchmaking is so full of closet hacking,...
4   Now the President is slapping Americans in the...
```

```
# information of the dataset
print(f"Train dataset shape {df_train.shape}")
print(f"Validation dataset shape {df_val.shape}")
```

```
Train dataset shape (74682, 4) Validation
dataset shape (1000, 4)
```

```
# checking nan values print("\t\tTrain
dataset") print(df_train.isna().sum())
print("*"*40) print("\n\n\t\tValidation
dataset") print(df_val.isna().sum())
```

```
                Train dataset


Tweet_ID Entity          0
Sentiment                0
Tweet_content            0
dtype: int64           686
****************************************


                Validation dataset


Tweet_ID Entity          0
Sentiment                0
Tweet_content            0
dtype: int64             0
```

```
# checking duplicated values print("Train
dataset") print(df_train.duplicated().sum())
print("*"*40)
print("\nValidation dataset") print(df_val.duplicated().sum())
```

```
Train dataset 2700
****************************************

Validation dataset 0
```

```
# remove dupicate and nan values
df_train.dropna(inplace=True)
df_train.drop_duplicates(inplace=True)

df_val.dropna(inplace=True)
df_val.drop_duplicates(inplace=True)
```

3 - EDA

```
# Assuming train and validation are your dataframes # Create subplots:

use 'domain' type for Pie charts
fig = make_subplots(rows=1, cols=2, specs=[[{'type':'domain'},
{'type':'domain'}]])

# Train dataset pie chart
fig.add_trace(go.Pie(labels=df_train['Sentiment'].value_counts().index
,
                     values=df_train['Sentiment'].value_counts(),
                     name="Train dataset"),
              1, 1)


# Validation dataset pie chart
fig.add_trace(go.Pie(labels=df_val['Sentiment'].value_counts().index,
                     values=df_val['Sentiment'].value_counts(), name="Validation
                     dataset"),
              1, 2)

# Update titles
fig.update_layout(
      title_text="Proportions of Target Classes", annotations=[dict(text='Train
      dataset', x=0.15, y=1.1,
font_size=15, showarrow=False),
                  dict(text='Validation dataset', x=0.88, y=1.1, font_size=15,
showarrow=False)]
)

# Show figure
fig.show()
```

{"config":{"plotlyServerURL":"https://plot.ly"},"data":[{"domain":
{"x":[0,0.45],"y":[0,1]},"labels":
["Negative","Positive","Neutral","Irrelevant"],"name":"Train
dataset","type":"pie","values":[21698,19713,17708,12537]},{"domain":
{"x":[0.55,1],"y":[0,1]},"labels":
["Neutral","Positive","Negative","Irrelevant"],"name":"Validation

dataset","type":"pie","values":[285,277,266,172]}],"layout":
{"annotations":[{"font":{"size":15},"showarrow":false,"text":"Train
dataset","x":0.15,"y":1.1},{"font":
{"size":15},"showarrow":false,"text":"Validation
dataset","x":0.88,"y":1.1}],"template":{"data":{"bar":[{"error_x":
{"color":"#2a3f5f"},"error_y":{"color":"#2a3f5f"},"marker":{"line":
{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"bar"}],"barpo
lar":[{"marker":{"line":{"color":"#E5ECF6","width":0.5},"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"barpolar"}]," carpet":[{"aaxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"baxis":
{"endlinecolor":"#2a3f5f","gridcolor":"white","linecolor":"white","min
orgridcolor":"white","startlinecolor":"#2a3f5f"},"type":"carpet"}],"ch
oropleth":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"choropleth"}],"contour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"contour"}],"contourcarpet":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"contourcarpet"}],"heatmap":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmap"}],"heatmapgl":[{"colorbar":
{"outlinewidth":0,"ticks":""},"colorscale":[[0,"#0d0887"],
[0.1111111111111111,"#46039f"],[0.2222222222222222,"#7201a8"],
[0.3333333333333333,"#9c179e"],[0.4444444444444444,"#bd3786"],
[0.5555555555555556,"#d8576b"],[0.6666666666666666,"#ed7953"],
[0.7777777777777778,"#fb9f3a"],[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"heatmapgl"}],"histogram":[{"marker":{"pattern":
{"fillmode":"overlay","size":10,"solidity":0.2}},"type":"histogram"}],
"histogram2d":[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2d"}],"histogram2dcontour":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],

[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"histogram2dcontour"}],"mesh3d":[{"colorbar":
{"outlinewidth":0,"ticks":""},"type":"mesh3d"}],"parcoords":[{"line":
{"colorbar":{"outlinewidth":0,"ticks":""}},"type":"parcoords"}],"pie":
[{"automargin":true,"type":"pie"}],"scatter":[{"fillpattern":
{"fillmode":"overlay","size":10,"solidity":0.2},"type":"scatter"}],"sc
atter3d":[{"line":{"colorbar":{"outlinewidth":0,"ticks":""}},"marker":
{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatter3d"}],"scattercarpet":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattercarpet"}],"scattergeo":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergeo"}],"scattergl":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattergl"}],"scattermapbox":
[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scattermapbox"}],"scatterpolar"
:[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolar"}],"scatterpolargl
":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterpolargl"}],"scatterterna
ry":[{"marker":{"colorbar":
{"outlinewidth":0,"ticks":""}},"type":"scatterternary"}],"surface":
[{"colorbar":{"outlinewidth":0,"ticks":""},"colorscale":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],
[1,"#f0f921"]],"type":"surface"}],"table":[{"cells":{"fill":
{"color":"#EBF0F8"},"line":{"color":"white"}},"header":{"fill":
{"color":"#C8D4E3"},"line":
{"color":"white"}},"type":"table"}]},"layout":{"annotationdefaults":
{"arrowcolor":"#2a3f5f","arrowhead":0,"arrowwidth":1},"autotypenumbers
":"strict","coloraxis":{"colorbar":
{"outlinewidth":0,"ticks":""}},"colorscale":{"diverging":
[[0,"#8e0152"],[0.1,"#c51b7d"],[0.2,"#de77ae"],[0.3,"#f1b6da"],
[0.4,"#fde0ef"],[0.5,"#f7f7f7"],[0.6,"#e6f5d0"],[0.7,"#b8e186"],
[0.8,"#7fbc41"],[0.9,"#4d9221"],[1,"#276419"]],"sequential":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]],"sequentialminus":
[[0,"#0d0887"],[0.1111111111111111,"#46039f"],
[0.2222222222222222,"#7201a8"],[0.3333333333333333,"#9c179e"],
[0.4444444444444444,"#bd3786"],[0.5555555555555556,"#d8576b"],

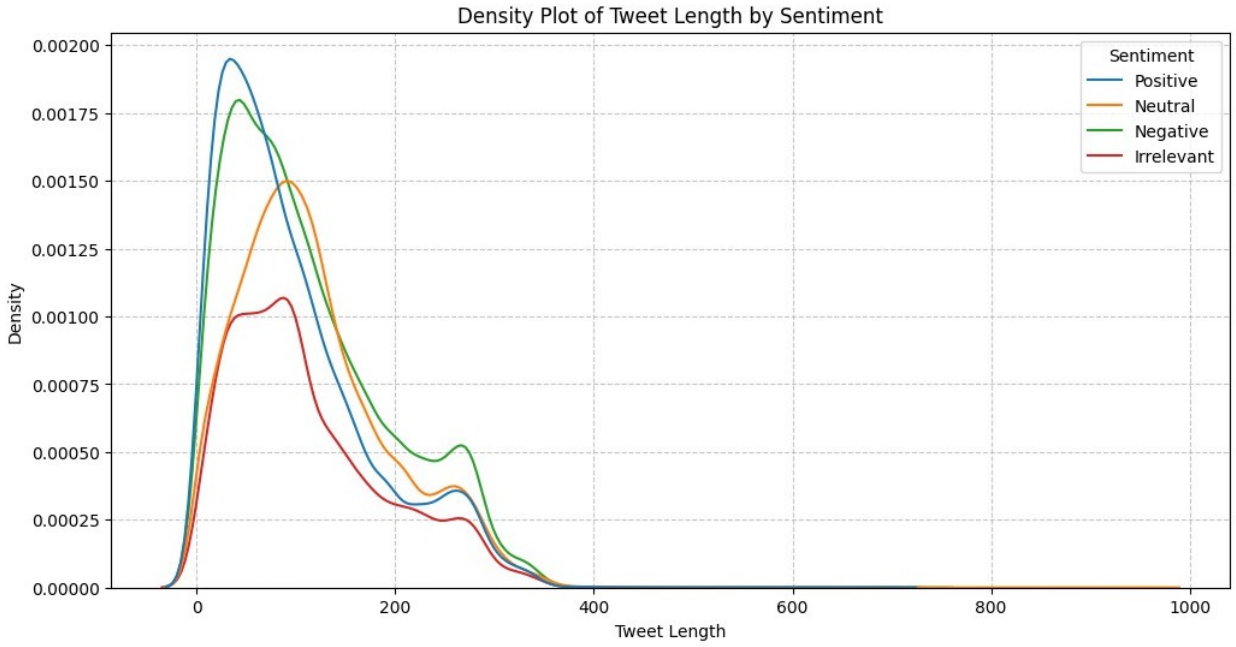
```
[0.6666666666666666,"#ed7953"],[0.7777777777777778,"#fb9f3a"],
[0.8888888888888888,"#fdca26"],[1,"#f0f921"]]},"colorway":
["#636efa","#EF553B","#00cc96","#ab63fa","#FFA15A","#19d3f3","#FF6692"
,"#B6E880","#FF97FF","#FECB52"],"font":{"color":"#2a3f5f"},"geo":
{"bgcolor":"white","lakecolor":"white","landcolor":"#E5ECF6","showlake
s":true,"showland":true,"subunitcolor":"white"},"hoverlabel":
{"align":"left"},"hovermode":"closest","mapbox":
{"style":"light"},"paper_bgcolor":"white","plot_bgcolor":"#E5ECF6","po
lar":{"angularaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF 6","radialaxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"scene":
{"xaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"yaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
,"zaxis":
{"backgroundcolor":"#E5ECF6","gridcolor":"white","gridwidth":2,"lineco
lor":"white","showbackground":true,"ticks":"","zerolinecolor":"white"}
},"shapedefaults":{"line":{"color":"#2a3f5f"}},"ternary":{"aaxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"baxis":
{"gridcolor":"white","linecolor":"white","ticks":""},"bgcolor":"#E5ECF 6","caxis":
{"gridcolor":"white","linecolor":"white","ticks":""}},"title":
{"x":5.0e-2},"xaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"", "title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2},"yaxis":
{"automargin":true,"gridcolor":"white","linecolor":"white","ticks":"", "title":
{"standoff":15},"zerolinecolor":"white","zerolinewidth":2}}},"title":
{"text":"Proportions of Target Classes"}}} import re

# Function to remove URLs
def remove_urls(text):
      url_pattern = re.compile(r'https?://\S+|www\.\S+') return
      url_pattern.sub(r'', text)

# Function to remove emojis
def remove_emojis(text): emoji_pattern
      = re.compile(
            "["
            u"\U0001F600-\U0001F64F"  # emoticons
            u"\U0001F300-\U0001F5FF"  # symbols & pictographs u"\U0001F680-
            \U0001F6FF"  # transport & map symbols u"\U0001F1E0-\U0001F1FF"
            # flags (iOS)
```

```
            u"\U00002702-\U000027B0"  # other symbols
            u"\U000024C2-\U0001F251"
            "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)

# Fill NaN values with an empty string
df_train['Tweet_content'] = df_train['Tweet_content'].fillna('')

# Remove URLs and emojis from the text
df_train['Cleaned_Text'] =
df_train['Tweet_content'].apply(remove_urls).apply(remove_emojis)

# Create the Text_Length column based on the cleaned text
df_train['Text_Length'] = df_train['Cleaned_Text'].apply(len)

# Display the first few rows to verify the new column print(df_train[['Tweet_content',
'Cleaned_Text', 'Text_Length']].head())
```

	Tweet_content	\
0	im getting on borderlands and i will murder yo...	
1	I am coming to the borders and I will kill you...	
2	im getting on borderlands and i will kill you ...	
3	im coming on borderlands and i will murder you...	
4	im getting on borderlands 2 and i will murder ...	

	Cleaned_Text	Text_Length
0	im getting on borderlands and i will murder yo...	53
1	I am coming to the borders and I will kill you...	51
2	im getting on borderlands and i will kill you ...	50
3	im coming on borderlands and i will murder you...	51
4	im getting on borderlands 2 and i will murder ...	57

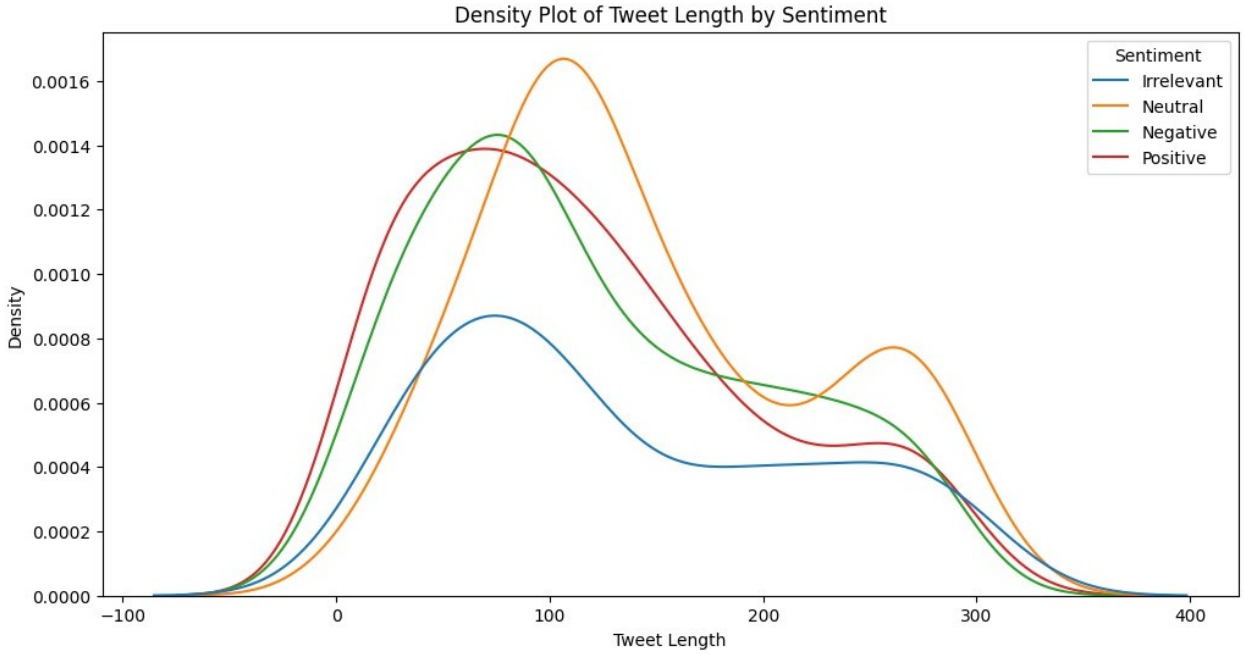
```
# Generate a density plot similar to the provided image
plt.figure(figsize=(12, 6))
sns.kdeplot(data=df_train, x='Text_Length', hue='Sentiment') plt.title('Density
Plot of Tweet Length by Sentiment') plt.xlabel('Tweet Length')
plt.ylabel('Density')
plt.grid(True, linestyle='--', alpha=0.7) plt.show()
```



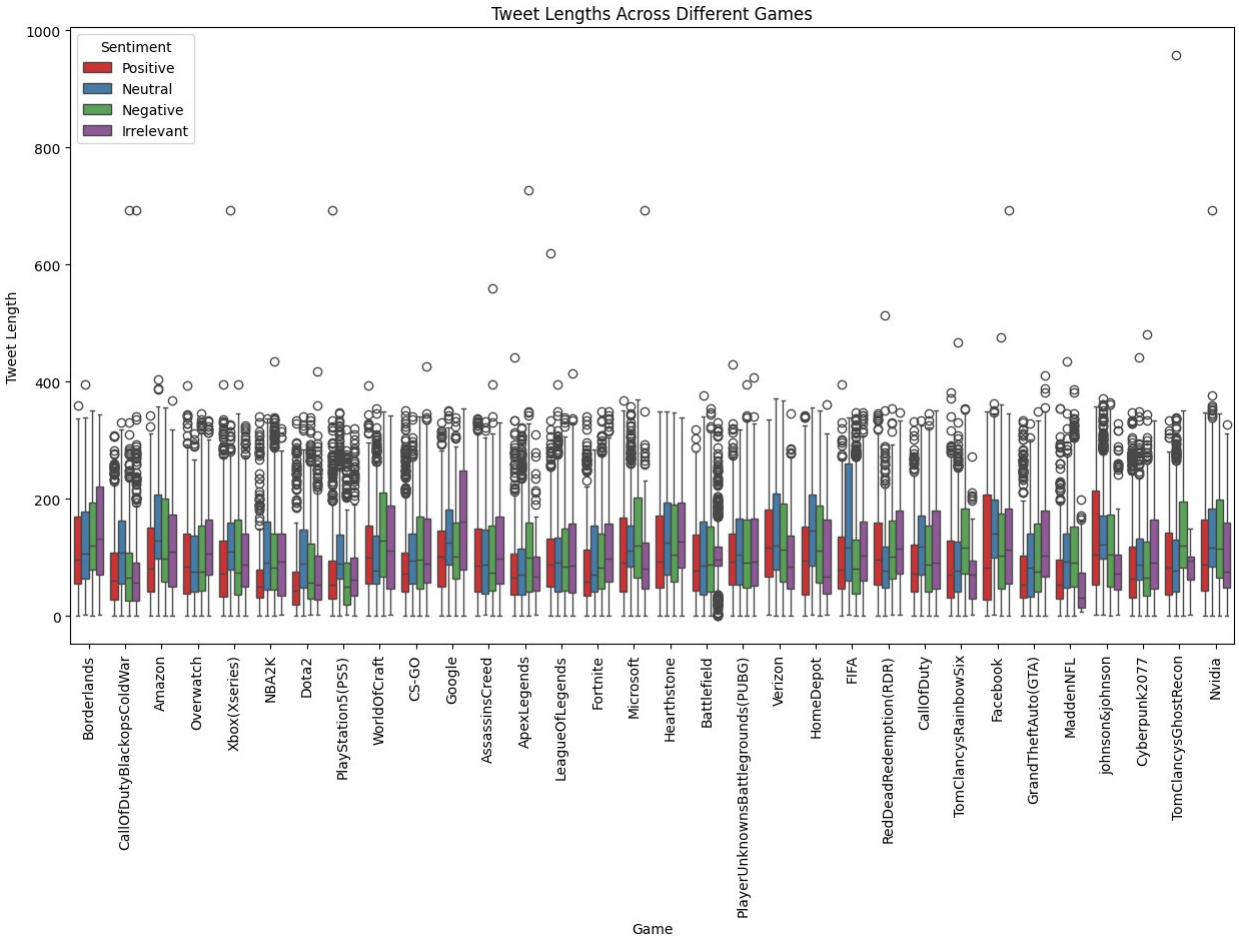

```
df_val['Tweet_content'] = df_val['Tweet_content'].fillna('')

# Remove URLs and emojis from the text
df_val['Cleaned_Text'] = df_val['Tweet_content'].apply(remove_urls).apply(remove_emojis)

# Create the Text_Length column based on the cleaned text
df_val['Text_Length'] = df_val['Cleaned_Text'].apply(len)

# Display the first few rows to verify the new column
print(df_val[['Tweet_content', 'Cleaned_Text', 'Text_Length']].head())



# Generate a density plot similar to the provided image
plt.figure(figsize=(12, 6))
sns.kdeplot(data=df_val, x='Text_Length', hue='Sentiment') plt.title('Density Plot of
Tweet Length by Sentiment') plt.xlabel('Tweet Length')
plt.ylabel('Density') plt.show()
```

```
                                                Tweet_content  \
0   I mentioned on Facebook that I was struggling ...
1   BBC News - Amazon boss Jeff Bezos rejects clai...
2   @Microsoft Why do I pay for WORD when it funct...
3   CSGO matchmaking is so full of closet hacking,...
4   Now the President is slapping Americans in the
```

	Cleaned_Text	Text_Length
0	I mentioned on Facebook that I was struggling ...	242
1	BBC News - Amazon boss Jeff Bezos rejects clai...	109
2	@Microsoft Why do I pay for WORD when it funct...	90
3	CSGO matchmaking is so full of closet hacking,...	71
4	Now the President is slapping Americans in the...	170




```
# Boxplot of tweet lengths across different games
plt.figure(figsize=(15, 8))
sns.boxplot(data=df_train, x='Entity', y='Text_Length', hue='Sentiment',
palette='Set1')
plt.title('Tweet Lengths Across Different Games') plt.xlabel('Game')
plt.ylabel('Tweet Length')
plt.xticks(rotation=90) plt.show()
```




```
def generate_word_cloud(sentiment):
    # Filter the dataset to include only tweets with the specified sentiment
    tweets = df_train[df_train['Sentiment'] == sentiment] ['Cleaned_Text']

    # Combine the text of all tweets with the specified sentiment
    text = ' '.join(tweets)

    # Generate the word cloud
    wordcloud = WordCloud(width=400, height=400,max_font_size=100,
background_color='white').generate(text)

    # Display the word cloud plt.figure(figsize=(10, 5))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.title(f'Word Cloud for {sentiment} Tweets')
    plt.axis('off')
    plt.show()
```

```
# Generate word clouds for Negative, Neutral, and Irrelevant sentiments
generate_word_cloud('Positive')
generate_word_cloud('Negative')
generate_word_cloud('Neutral')
generate_word_cloud('Irrelevant')
```

Word Cloud for Negative Tweets

Word Cloud for Neutral Tweets

4 - Data Processing

```
import pandas as pd

# Function to remove outliers based on IQR
def remove_outliers(df, column): Q1 =
      df[column].quantile(0.25) Q3 =
      df[column].quantile(0.75)
      IQR = Q3 - Q1
      lower_bound = Q1 - 1.5 * IQR
      upper_bound = Q3 + 1.5 * IQR
      return df[(df[column] >= lower_bound) & (df[column] <=
upper_bound)]

# Apply the function to remove outliers in the Text_Length column
df_cleaned = remove_outliers(df_train, 'Text_Length')

# Display the number of rows before and after removing outliers print(f'Number of rows before
removing outliers: {len(df_train)}') print(f'Number of rows after removing outliers: {len(df_cleaned)}')

# save the cleaned dataset to a new CSV file
df_cleaned.to_csv('cleaned_twitter_training.csv',  index=False)
```

```
Number of rows before removing outliers: 71656 Number of
rows after removing outliers: 70871

# Tokenazation and Lemmatization

import spacy
import pandas as pd

# Load the spaCy English model
nlp = spacy.load('en_core_web_sm')

# Function to perform tokenization and lemmatization using spaCy
def tokenize_and_lemmatize(text): doc =
    nlp(text)
    lemmatized_tokens = [token.lemma_ for token in doc] return '
    '.join(lemmatized_tokens)

# Apply the function to the Cleaned_Text column df_train['Processed_Text'] =
df_train['Cleaned_Text'].apply(tokenize_and_lemmatize)

# Display the first few rows to verify the new column
print(df_train[['Cleaned_Text', 'Processed_Text']].head())

# Optionally, save the processed dataset to a new CSV file
df_train.to_csv('processed_twitter_training.csv',  index=False)

                                                Cleaned_Text  \
0   im getting on borderlands and i will murder yo...
1   I am coming to the borders and I will kill you...
2   im getting on borderlands and i will kill you ...
3   im coming on borderlands and i will murder you...
4   im getting on borderlands 2 and i will murder ...

                                              Processed_Text
0   I m get on borderland and I will murder you all ,
1   I be come to the border and I will kill you all ,
2     I m get on borderland and I will kill you all ,
3   I m come on borderland and I will murder you a...
4   I m get on borderland 2 and I will murder you ...

from sklearn.model_selection import train_test_split

# Define the features and target variable
X = df_train['Processed_Text']  # Features (processed text)
y = df_train['Sentiment']                    # Target variable (sentiment)

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Display the sizes of the training and testing sets print(f'Size of training set:
{len(X_train)}') print(f'Size of testing set: {len(X_test)}')
```

```
Size of training set: 57324 Size of
testing set: 14332
```

```
# Vectorize Text Data

from sklearn.feature_extraction.text import TfidfVectorizer

# Assuming X_train and X_test are defined # Step 1:

Initialize the TfidfVectorizer
vectorizer = TfidfVectorizer(max_features=5000)

# Step 2: Fit the vectorizer on the training data and transform the training data into TF-IDF feature
vectors
X_train_tfidf = vectorizer.fit_transform(X_train)

# Step 3: Transform the test data into TF-IDF feature vectors using the already fitted vectorizer
X_test_tfidf = vectorizer.transform(X_test)

# Display the shape of the resulting TF-IDF matrices print(f'Shape of X_train_tfidf:
{X_train_tfidf.shape}') print(f'Shape of X_test_tfidf: {X_test_tfidf.shape}')
```

```
Shape of X_train_tfidf: (57324, 5000)
Shape of X_test_tfidf: (14332, 5000)
```

5 - Machine Learing

```
from sklearn.linear_model import LogisticRegression from
sklearn.neighbors import KNeighborsClassifier from
sklearn.tree import DecisionTreeClassifier from
sklearn.ensemble import
RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier,E
xtraTreesClassifier
from sklearn.linear_model import RidgeClassifier,SGDClassifier from
sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix



# Step 1: Initialize the LogisticRegression model
```

```
model = LogisticRegression(max_iter=1000)

# Step 2: Train the model
model.fit(X_train_tfidf, y_train)

# Step 3: Make predictions on the test set
y_pred = model.predict(X_test_tfidf)

# Step 4: Evaluate the model
accuracy = accuracy_score(y_test, y_pred) report =
classification_report(y_test, y_pred) conf_matrix =
confusion_matrix(y_test, y_pred)

# Print the evaluation results print(f'Accuracy:
{accuracy:.4f}') print('Classification Report:')
print(report)
```

```
Accuracy: 0.6905 Classification
Report:
```

	precision	recall	f1-score	support
Irrelevant	0.68	0.53	0.60	2529
Negative	0.73	0.79	0.76	4383
Neutral	0.68	0.63	0.66	3543
Positive	0.65	0.73	0.69	3877
accuracy			0.69	14332
macro avg	0.69	0.67	0.68	14332
weighted avg	0.69	0.69	0.69	14332

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer from
sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression from
sklearn.svm import SVC
from sklearn.ensemble import RandomForestClassifier from
sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib


# Define a dictionary to store the models
models = {
    'Logistic Regression': LogisticRegression(max_iter=1000),
    #'Support Vector Machine': SVC(),
```

```
    'Random Forest': RandomForestClassifier(), 'Naive
    Bayes': MultinomialNB(),
    'Extra Tree Classifier': ExtraTreesClassifier()
}

# Train and evaluate each model
for model_name, model in models.items():
    print(f'\nTraining {model_name}...') #
    Train the model model.fit(X_train_tfidf,
    y_train)
    # Make predictions on the test set y_pred=
    model.predict(X_test_tfidf) # Evaluate the model
    accuracy = accuracy_score(y_test, y_pred) report =
    classification_report(y_test, y_pred) conf_matrix =
    confusion_matrix(y_test, y_pred) # Print the
    evaluation results print(f'Accuracy: {accuracy:.4f}')
    print('Classification Report:')
    print(report)
```

```
Training Logistic Regression...
Accuracy: 0.6905 Classification
Report:
              precision    recall  f1-score   support

  Irrelevant       0.68      0.53      0.60      2529
    Negative       0.73      0.79      0.76      4383
     Neutral       0.68      0.63      0.66      3543
    Positive       0.65      0.73      0.69      3877

    accuracy                           0.69     14332
   macro avg       0.69      0.67      0.68     14332
weighted avg       0.69      0.69      0.69     14332


Training Random Forest...
Accuracy: 0.8986 Classification
Report:
              precision    recall  f1-score   support

  Irrelevant       0.96      0.82      0.88      2529
    Negative       0.90      0.93      0.91      4383
     Neutral       0.90      0.90      0.90      3543
    Positive       0.87      0.92      0.89      3877

    accuracy                           0.90     14332
   macro avg       0.91      0.89      0.90     14332
weighted avg       0.90      0.90      0.90     14332
```

Training Naive Bayes... Accuracy:
0.6346 Classification Report:

	precision	recall	f1-score	support
Irrelevant	0.77	0.32	0.46	2529
Negative	0.62	0.81	0.70	4383
Neutral	0.68	0.54	0.60	3543
Positive	0.60	0.73	0.66	3877
accuracy			0.63	14332
macro avg	0.67	0.60	0.60	14332
weighted avg	0.65	0.63	0.62	14332

Training Extra Tree Classifier... Accuracy:
0.9255
Classification Report:

	precision	recall	f1-score	support
Irrelevant	0.97	0.87	0.92	2529
Negative	0.93	0.94	0.94	4383
Neutral	0.92	0.93	0.92	3543
Positive	0.90	0.93	0.92	3877
accuracy			0.93	14332
macro avg	0.93	0.92	0.92	14332
weighted avg	0.93	0.93	0.93	14332

```
# Testing diffreent Extra Tree Classifiers Paramenters

from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib

# Assuming X_train_tfidf, X_test_tfidf, y_train, y_test are already defined from the previous steps

# Initialize the Extra Trees Classifier with tuned parameters
extra_trees_model = ExtraTreesClassifier(
    n_estimators=500,                # Number of trees in the forest
    max_depth=None,                  # Maximum depth of the trees, None means
nodes are expanded until all leaves are pure
    min_samples_split=2,             # Minimum number of samples required to split an
internal node
    min_samples_leaf=1,              # Minimum number of samples required to be at a leaf
node
```

```
        random_state=365,                    # Seed for reproducibility
        n_jobs=-1                            # Use all available cores
)

# Train the model
extra_trees_model.fit(X_train_tfidf,  y_train)

# Make predictions on the test set
y_pred = extra_trees_model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred) report =
classification_report(y_test, y_pred) conf_matrix =
confusion_matrix(y_test, y_pred)

# Print the evaluation results print(f'Accuracy:
{accuracy:.4f}') print('Classification Report:')
print(report)
print('Confusion Matrix:')
print(conf_matrix)
```

```
Accuracy: 0.9291 Classification
Report:
                    precision    recall  f1-score   support

   Irrelevant           0.97      0.88      0.92      2529
     Negative           0.94      0.95      0.94      4383
      Neutral           0.92      0.93      0.93      3543
     Positive           0.91      0.94      0.92      3877

     accuracy                               0.93     14332
    macro avg           0.93      0.92      0.93     14332
weighted avg            0.93      0.93      0.93     14332

Confusion Matrix:
[[2224   82   93  130]
 [  27 4148   92  116]
 [  11  102 3302  128]
 [  35   98  102 3642]]
```

```
import pandas as pd
from sklearn.ensemble import ExtraTreesClassifier from
sklearn.model_selection import GridSearchCV
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import joblib

# Define the parameter grid for GridSearchCV
```

```
param_grid = {
      'n_estimators': [100, 200, 300],
      'max_depth': [None, 10, 20, 30],
      'min_samples_split': [2, 5, 10],
      'min_samples_leaf': [1, 2, 4]
}

# Initialize the Extra Trees Classifier
extra_trees = ExtraTreesClassifier(random_state=42, n_jobs=-1)

# Initialize GridSearchCV
grid_search = GridSearchCV(estimator=extra_trees,
param_grid=param_grid, cv=5, n_jobs=-1, verbose=2)

# Fit GridSearchCV to the data
grid_search.fit(X_train_tfidf, y_train)

# Get the best estimator
best_model = grid_search.best_estimator_

# Make predictions on the test set
y_pred = best_model.predict(X_test_tfidf)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred) report =
classification_report(y_test, y_pred) conf_matrix =
confusion_matrix(y_test, y_pred)

# Print the best parameters and the evaluation results print(f'Best
Parameters: {grid_search.best_params_}') print(f'Accuracy: {accuracy:.4f}')
print('Classification Report:')
print(report) print('Confusion
Matrix:') print(conf_matrix)


Fitting 5 folds for each of 108 candidates, totalling 540 fits [CV] END
max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=  57.1s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=  57.2s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=  57.6s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=  57.6s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=  57.3s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time= 2.1min
```

```
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time= 2.1min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time= 2.1min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=  52.4s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time= 2.2min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time= 2.2min

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/joblib/externals/loky/process_executor.py:752: UserWarning:

A worker stopped while some jobs were given to the executor. This can be caused
by a too short worker timeout or by a memory leak.


[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=  51.4s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=  51.8s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=  53.1s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time= 3.3min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time= 3.3min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time= 3.3min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=  59.6s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time= 3.4min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time= 3.4min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time= 2.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time= 2.1min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time= 2.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time= 2.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time= 2.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=  43.3s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
```

```
n_estimators=100; total time=  43.3s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=  43.1s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time= 2.9min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=  49.9s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=  50.8s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time= 3.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time= 3.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time= 1.7min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time= 3.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time= 3.0min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time= 1.7min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time= 1.7min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time= 1.7min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=  29.5s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=  30.2s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time= 1.7min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=  29.7s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=  30.1s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=  28.7s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time= 2.5min
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time= 2.5min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=  54.8s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=  55.7s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=  55.7s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time= 2.5min
```

```
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time= 2.4min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=  49.1s
[CV] END max_depth=None, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time= 2.4min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=  46.1s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=  23.5s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=  23.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=  23.8s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=  24.5s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=  25.5s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=  53.7s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=  53.7s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=  53.0s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=  54.0s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=  53.6s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=  22.8s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=  23.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=  23.1s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time= 1.3min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time= 1.3min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
```

```
n_estimators=300; total time= 1.3min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=  21.2s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=  21.5s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time= 1.2min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=  42.6s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=  42.7s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=  42.2s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=  42.0s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=  41.7s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=  15.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=  15.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=  15.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=  15.2s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time= 1.1min
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=  14.6s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time= 1.1min
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=  29.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time= 1.1min
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time= 1.1min
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=  29.4s
[CV] END max_depth=None, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time= 1.1min
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=  30.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=  29.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=  29.9s
```

```
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=  15.0s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=  15.7s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=  14.9s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=  14.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=  14.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=  44.3s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=  43.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=  43.6s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=  42.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=  42.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=  27.5s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=  27.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=  27.5s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=  27.0s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=  27.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=  14.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=  14.0s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=  14.5s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=  41.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=  42.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=  42.3s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=  13.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=  13.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=  40.5s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=5,
```

```
n_estimators=300; total time=  41.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=  27.1s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=  26.8s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=  27.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=  26.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         1.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         1.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         1.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         1.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         1.5s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=  26.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=         2.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=         2.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=         2.9s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=         2.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=         2.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=         3.9s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=         4.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=         4.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         1.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=         3.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=         4.1s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         1.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         1.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         1.4s
```

```
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         1.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         2.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         2.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         2.8s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         2.3s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=  39.8s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         2.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=         3.9s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=         3.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=         4.2s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=  38.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=         4.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         1.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         1.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         1.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         1.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         1.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=         3.6s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         2.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         2.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         2.6s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=  39.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         2.3s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         2.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
```

```
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   3.5s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   4.0s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   4.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   1.3s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=  37.4s
[CV] END max_depth=None, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=  37.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   1.1s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   1.1s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   2.1s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   2.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   2.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   2.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   2.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=   3.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=   3.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   1.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   1.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   1.3s
```

```
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        3.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        3.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        2.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        2.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        2.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        2.6s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        2.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        1.1s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        3.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        3.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        1.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        3.7s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        1.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        3.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        3.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=        2.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=        2.4s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=        2.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=        2.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=        2.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=        1.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=        1.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
```

```
n_estimators=300; total time=   3.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   1.6s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   4.0s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   3.8s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   1.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   1.3s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   3.5s
[CV] END max_depth=10, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   2.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   2.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   2.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   2.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   2.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   1.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   1.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   3.7s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   3.8s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   1.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   3.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   1.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   3.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=   2.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=   2.8s
```

```
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         2.9s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         2.7s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         2.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.2s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.4s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         3.8s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         4.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.1s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         4.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         3.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         3.5s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         2.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         2.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         2.6s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         2.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         2.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         3.8s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         3.8s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         4.0s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         3.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         3.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=         3.3s
[CV] END max_depth=10, min_samples_leaf=4, min_samples_split=10,
```

n_estimators=300; total time= 3.8s

```
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=        3.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=        2.9s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=        2.9s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=        5.4s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=        5.9s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=        5.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=        5.4s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=        5.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=        2.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=        2.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=        2.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=        8.4s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=        7.7s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=        8.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=        2.7s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=        2.7s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=        7.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=        8.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=        5.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=        5.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=        5.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=        5.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=        5.0s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=        2.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
```

```
n_estimators=100; total time=   2.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=   2.4s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=   7.4s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=   2.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=   6.8s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=   2.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=   6.7s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=   7.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=   6.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   4.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   4.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   4.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   4.6s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   4.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   2.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   2.4s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   2.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   6.3s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   6.5s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   6.6s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   2.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   6.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   2.1s
[CV] END max_depth=20, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   6.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   4.1s
```

```
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=        4.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=        4.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=        4.3s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=        3.7s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=        2.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=        2.2s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=        2.2s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        5.8s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        6.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=        2.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        6.5s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=        2.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        5.8s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=        6.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        3.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        4.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        4.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        3.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=        3.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        2.2s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        2.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=        2.3s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        6.3s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=        6.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
```

```
n_estimators=100; total time=   2.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=   6.3s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=   2.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=   5.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=   6.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=   3.8s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=   4.2s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=   3.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=   3.9s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=   3.8s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.2s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   6.1s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   5.8s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   6.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   1.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.0s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   5.5s
[CV] END max_depth=20, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   5.5s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   3.7s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   3.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   3.7s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   3.8s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   3.7s
```

```
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=         2.2s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=         2.1s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=         2.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=         5.5s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=         5.7s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=         5.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=         1.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=         5.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=         1.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=         5.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         3.5s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         3.8s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         3.8s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         3.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         3.7s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         2.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         5.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         2.1s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         5.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         5.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         1.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         2.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         5.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=5,
```

```
n_estimators=300; total time=   5.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=   3.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=   3.7s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=   3.5s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=   3.8s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=   3.6s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=   5.4s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=   5.9s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=   5.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   4.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   4.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   4.0s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=   5.5s
[CV] END max_depth=20, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=   5.5s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   3.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=100; total time=   3.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=   7.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=   7.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=   7.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=   7.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=200; total time=   7.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   3.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   3.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=   4.1s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=  12.1s
```

```
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=  11.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         3.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=100; total time=         3.7s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=  12.7s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=  11.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=2,
n_estimators=300; total time=  11.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         7.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         7.4s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         7.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         7.2s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=200; total time=         7.4s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         3.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         3.4s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         3.7s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=  11.3s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         3.5s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=100; total time=         3.3s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=  10.7s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=  11.3s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=  10.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=5,
n_estimators=300; total time=  10.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         6.5s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         6.9s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=         7.0s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
```

```
n_estimators=200; total time=   6.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=200; total time=   6.7s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   3.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   3.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   3.2s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   9.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=  10.6s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=  10.5s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   3.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=100; total time=   3.1s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   9.8s
[CV] END max_depth=30, min_samples_leaf=1, min_samples_split=10,
n_estimators=300; total time=   9.9s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   6.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   6.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   6.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   6.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=200; total time=   6.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   3.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   3.6s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   3.5s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=   9.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=  10.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=   9.9s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   3.2s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=100; total time=   3.0s
```

```
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=         9.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=2,
n_estimators=300; total time=         9.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=         5.7s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=         6.0s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=         5.8s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=         6.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=200; total time=         5.9s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=         3.2s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=         3.5s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=         3.5s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=         9.5s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=         9.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=         2.8s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=100; total time=         3.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=         9.7s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=         9.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=5,
n_estimators=300; total time=         9.4s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=         6.2s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=         6.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=         5.8s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=         6.3s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=200; total time=         5.9s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=         3.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=         3.2s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
```

```
n_estimators=100; total time=   3.1s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   9.2s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   8.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.6s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   9.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=100; total time=   2.8s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   8.7s
[CV] END max_depth=30, min_samples_leaf=2, min_samples_split=10,
n_estimators=300; total time=   8.9s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   5.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   5.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   4.9s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   5.5s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=200; total time=   5.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   3.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   3.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   3.3s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   8.7s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   7.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   8.5s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   2.7s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=100; total time=   2.8s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   7.9s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=2,
n_estimators=300; total time=   8.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=   5.2s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=   5.8s
```

```
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         5.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         5.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=200; total time=         5.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         3.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         3.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         8.5s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         8.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         3.3s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         8.6s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         2.8s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=100; total time=         2.8s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         8.3s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=5,
n_estimators=300; total time=         8.3s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         5.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         5.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         5.1s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         5.7s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=200; total time=         5.0s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         6.9s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         6.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         6.5s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         4.4s
[CV] END max_depth=30, min_samples_leaf=4, min_samples_split=10,
n_estimators=300; total time=         4.4s
Best Parameters: {'max_depth': None, 'min_samples_leaf': 1, 'min_samples_split': 2,
'n_estimators': 300}
Accuracy: 0.9279 Classification
```

	precision	recall	f1-score	support
Irrelevant	0.97	0.88	0.92	2529
Negative	0.94	0.94	0.94	4383
Neutral	0.92	0.93	0.93	3543
Positive	0.90	0.94	0.92	3877
accuracy			0.93	14332
macro avg	0.93	0.92	0.93	14332
weighted avg	0.93	0.93	0.93	14332

Confusion Matrix:

```
[[2224   85   90  130]
 [  25 4131   98  129]
 [  13   97 3306  127]
 [  40   95  105 3637]]
```

6 - Validation Test

```
# Save the Model and Vectorizer

import joblib

# Save the best model
joblib.dump(best_model, 'extra_trees_model_best.pkl')

# vectorizer
joblib.dump(vectorizer, 'tfidf_vectorizer.pkl') ['tfidf_vectorizer.pkl']

# Load the Model and Vectorizer

import joblib
from sklearn.feature_extraction.text import TfidfVectorizer

# Load the best model and vectorizer
best_model = joblib.load('extra_trees_model_best.pkl') vectorizer =
joblib.load('tfidf_vectorizer.pkl')

# Checking on validation df - #

Preprocessing function
def remove_urls(text):
    url_pattern = re.compile(r'https?://\S+|www\.\S+') return
    url_pattern.sub(r'', text)

def remove_emojis(text): emoji_pattern =
    re.compile(
```

```
            "["
            u"\U0001F600-\U0001F64F"  # emoticons
            u"\U0001F300-\U0001F5FF"  # symbols & pictographs u"\U0001F680-
            \U0001F6FF"  # transport & map symbols u"\U0001F1E0-\U0001F1FF"
            # flags (iOS)
            u"\U00002702-\U000027B0"  # other symbols
            u"\U000024C2-\U0001F251"
            "]+", flags=re.UNICODE)
    return emoji_pattern.sub(r'', text)

def tokenize_and_lemmatize(text): doc =
    nlp(text)
    lemmatized_tokens = [token.lemma_ for token in doc] return '
    '.join(lemmatized_tokens)


# Apply preprocessing to the validation data
df_val['Cleaned_Text'] =
df_val['Tweet_content'].fillna('').apply(remove_urls).apply(remove_emo
jis).apply(tokenize_and_lemmatize)

# Transform the text data using the loaded vectorizer
X_validation_tfidf = vectorizer.transform(df_val['Cleaned_Text'])

# Make predictions using the trained model
y_validation_pred = best_model.predict(X_validation_tfidf)

# Evaluate the model's performance on the validation set
y_validation_true = df_val['Sentiment']
accuracy = accuracy_score(y_validation_true, y_validation_pred) report =
classification_report(y_validation_true, y_validation_pred) conf_matrix =
confusion_matrix(y_validation_true, y_validation_pred)


# Print the evaluation results
print(f'Accuracy on validation data: {accuracy:.4f}') print('Classification Report:')
print(report)
```

```
Accuracy on validation data: 0.9720
Classification Report:
              precision    recall  f1-score   support

  Irrelevant       0.99      0.95      0.97       172
    Negative       0.96      0.98      0.97       266
     Neutral       0.97      0.98      0.97       285
    Positive       0.97      0.97      0.97       277

    accuracy                           0.97      1000
   macro avg       0.97      0.97      0.97      1000
```

weighted avg	0.97	0.97	0.97	1000

```
import pandas as pd
import numpy as np
import re
import nltk
from nltk.corpus import stopwords
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Load data

column_names=['Tweet_ID','Entity','Sentiment','Tweet_content'] data =
pd.read_csv("twitter_training.csv",
                            sep=',',names=column_names)

# Display the first few rows of the dataframe
data.head()
```

	Tweet_ID	Entity	Sentiment	\
0	2401	Borderlands	Positive	
1	2401	Borderlands	Positive	
2	2401	Borderlands	Positive	
3	2401	Borderlands	Positive	
4	2401	Borderlands	Positive	

	Tweet_content
0	im getting on borderlands and i will murder yo...
1	I am coming to the borders and I will kill you...
2	im getting on borderlands and i will kill you ...
3	im coming on borderlands and i will murder you...
4	im getting on borderlands 2 and i will murder ...

```
import re import
nltk
from nltk.corpus import stopwords

nltk.download('stopwords')
stop_words = set(stopwords.words('english'))
```

```
def clean_text(text):
    if isinstance(text, float): return ""
    text = re.sub(r'http\S+', '', text)  # remove URLs text = re.sub(r'@\w+', '', text)
                                                    # remove mentions text
    = re.sub(r'#', '', text)                        # remove hashtags text
    = re.sub(r'\d+', '', text)                      # remove numbers
    text = text.lower()                             # convert to lowercase text =
    re.sub(r'[^\w\s]', '', text)  # remove punctuation text = ' '.join(word for word in
    text.split() if word not in
stop_words)  # remove stopwords
```

```
    return text

data['cleaned_text'] = data['Tweet_content'].astype(str).apply(clean_text)

# Display the first few rows of the cleaned dataframe
data.head()
```

```
[nltk_data] Error loading stopwords: <urlopen error [SSL:
[nltk_data]     CERTIFICATE_VERIFY_FAILED] certificate verify failed:
[nltk_data]     unable to get local issuer certificate (_ssl.c:1006)>
```

```
   Tweet_ID           Entity Sentiment  \
0      2401  Borderlands  Positive
1      2401  Borderlands  Positive
2      2401  Borderlands  Positive
3      2401  Borderlands  Positive
4      2401  Borderlands  Positive
                                                Tweet_content  \
0  im getting on borderlands and i will murder yo...
1  I am coming to the borders and I will kill you...
2  im getting on borderlands and i will kill you ...
3  im coming on borderlands and i will murder you...
4  im getting on borderlands 2 and i will murder ...

                        cleaned_text
0  im getting borderlands murder
1                coming borders kill
2     im getting borderlands kill
3    im coming borderlands murder
4  im getting borderlands murder
```

```
from sklearn.model_selection import train_test_split
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences

# Define max number of words and max sequence length
max_words = 10000
max_len = 100

tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(data['cleaned_text'])
sequences = tokenizer.texts_to_sequences(data['cleaned_text']) padded_sequences
= pad_sequences(sequences, maxlen=max_len)

# Prepare labels (assuming Sentiment is categorical with values like 'Positive', 'Negative', 'Neutral')
label_mapping = {'Positive': 1, 'Negative': 0, 'Neutral': 2}  # Adjust according to your needs
data['label'] = data['Sentiment'].map(label_mapping)
```

```
# Drop any rows with NaN labels
data = data.dropna(subset=['label'])

# Update sequences and labels after dropping NaNs padded_sequences
= padded_sequences[data.index] labels = data['label']

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(padded_sequences, labels,
test_size=0.2, random_state=42)

# Display shapes of training and testing sets
print(X_train.shape, X_test.shape, y_train.shape, y_test.shape) (49353,

100) (12339, 100) (49353,) (12339,)

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout,
Bidirectional
from tensorflow.keras.optimizers import Adam

# Build the LSTM model
model = Sequential()
model.add(Embedding(max_words, 128, input_length=max_len))
model.add(Bidirectional(LSTM(64, return_sequences=True)))
model.add(Dropout(0.5))
model.add(Bidirectional(LSTM(32))) model.add(Dense(1,
activation='sigmoid'))

model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy',
metrics=['accuracy'])

# Display the model summary
model.summary()

# Train the model
history = model.fit(X_train, y_train, epochs=3, batch_size=32, validation_split=0.2,
verbose=1)

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn( Model:

"sequential_11"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━
━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━

━━━━━━━━━━━━━┓
┃ Layer (type)                                    ┃ Output Shape
Param #  ┃
```

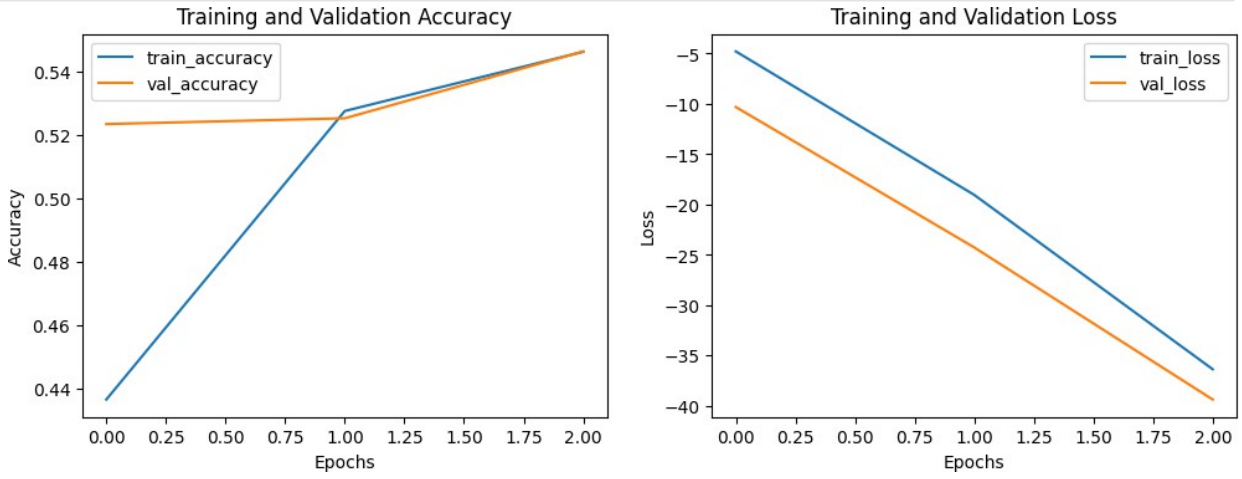
```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━┓
┃ embedding_11 (Embedding)            ┃ ?                          ┃     0
(unbuilt) ┃
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
           ┃ bidirectional_10                ┃ ?             ┃     0
                   (unbuilt) ┃
             ┃ (Bidirectional)                ┃               ┃
                             ┃
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
┃ dropout_7 (Dropout)                 ┃ ?                          ┃     0
(unbuilt) ┃
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
           ┃ bidirectional_11                ┃ ?             ┃     0
                   (unbuilt) ┃
             ┃ (Bidirectional)                ┃               ┃
                             ┃
├──────────────────────────────────┼─────────────────────────┼──────────
──────┤
┃ dense_13 (Dense)                    ┃ ?                          ┃     0
(unbuilt) ┃
└──────────────────────────────────┴─────────────────────────┴──────────
─────┘

 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B) Epoch

1/3
1234/1234 ━━━━━━━━━━━━━━━━━━━━ 103s 82ms/step - accuracy:
0.3859 - loss: -2.1170 - val_accuracy: 0.5236 - val_loss: -10.3079
Epoch 2/3
1234/1234 ━━━━━━━━━━━━━━━━━━━━ 198s 161ms/step - accuracy:
0.5281 - loss: -15.6977 - val_accuracy: 0.5254 - val_loss: -24.2590
Epoch 3/3
1234/1234 ━━━━━━━━━━━━━━━━━━━━ 104s 85ms/step - accuracy:
0.5462 - loss: -32.3638 - val_accuracy: 0.5466 - val_loss: -39.3671

# Evaluate model on test data
loss, accuracy = model.evaluate(X_test, y_test) print(f'Test
accuracy: {accuracy:.4f}')
```

```
386/386 ━━━━━━━━━━━━━━━━━━━━ 8s 21ms/step - accuracy: 0.5482 -
loss: - 40.6241
```

```
Test accuracy: 0.5497
```

```
# Evaluate the model
loss, accuracy = model.evaluate(X_test, y_test, verbose=1) print(f'Test
Accuracy: {accuracy * 100:.2f}%')

# Plot training and validation accuracy/loss
import matplotlib.pyplot as plt

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='train_accuracy')
plt.plot(history.history['val_accuracy'], label='val_accuracy')
plt.title('Training and Validation Accuracy') plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()

plt.subplot(1, 2, 2) plt.plot(history.history['loss'],
label='train_loss')
plt.plot(history.history['val_loss'], label='val_loss') plt.title('Training
and Validation Loss') plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()

plt.show()
```

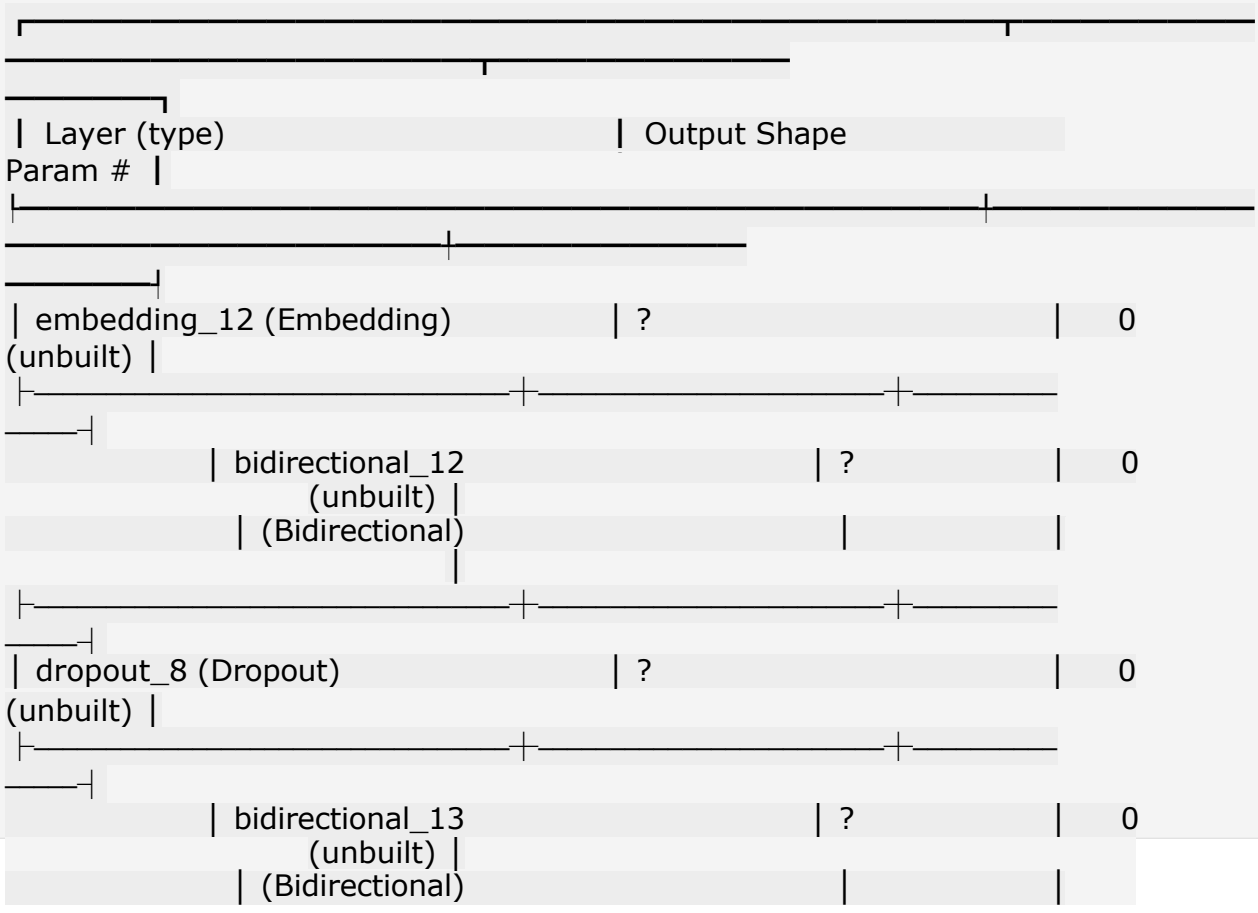
```
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 8s 19ms/step - accuracy: 0.5482 -
loss: - 40.6241
Test Accuracy: 54.97%
```




```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense,
Dropout, Bidirectional
from tensorflow.keras.optimizers import Adam
```

```
# Build the RNN model with a simple RNN layer
model = Sequential()
model.add(Embedding(max_words, 128, input_length=max_len))
model.add(Bidirectional(SimpleRNN(64, return_sequences=True)))
model.add(Dropout(0.5))  model.add(Bidirectional(SimpleRNN(32)))
model.add(Dense(3, activation='softmax'))  # Adjust output layer for 3 classes

model.compile(optimizer=Adam(learning_rate=0.001),
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

# Display the model summary
model.summary()

# Train the model
history = model.fit(X_train, y_train, epochs=10, batch_size=32, validation_split=0.2,
verbose=1)
```

```
/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn( Model:

"sequential_12"
```



```
│
├─────────────────────────────────┼────────────────────────┼───────────────
──┤
│ dense_14 (Dense)                │ ?                      │     0
(unbuilt) │
└─────────────────────────────────┴────────────────────────┴───────────────
──┘

 Total params: 0 (0.00 B)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 0 (0.00 B) Epoch

1/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46s 36ms/step - accuracy:
0.5132 - loss: 0.9570 - val_accuracy: 0.7685 - val_loss: 0.5853
Epoch 2/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.8405 - loss: 0.4117 - val_accuracy: 0.8066 - val_loss: 0.5198
Epoch 3/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.8952 - loss: 0.2678 - val_accuracy: 0.8208 - val_loss: 0.4930
Epoch 4/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9232 - loss: 0.1875 - val_accuracy: 0.8192 - val_loss: 0.5232
Epoch 5/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9321 - loss: 0.1588 - val_accuracy: 0.8164 - val_loss: 0.5721
Epoch 6/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9257 - loss: 0.1704 - val_accuracy: 0.8225 - val_loss: 0.5698
Epoch 7/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9406 - loss: 0.1376 - val_accuracy: 0.8104 - val_loss: 0.6109
Epoch 8/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9384 - loss: 0.1370 - val_accuracy: 0.8168 - val_loss: 0.6267
Epoch 9/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45s 37ms/step - accuracy:
0.9358 - loss: 0.1496 - val_accuracy: 0.8022 - val_loss: 0.6322
Epoch 10/10
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy:
0.9380 - loss: 0.1457 - val_accuracy: 0.8235 - val_loss: 0.6481
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, SimpleRNN, Dense,
Dropout
```

 # Define RNN model

```
model_rnn = Sequential([
    Embedding(max_words, 128, input_length=max_len),
    SimpleRNN(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(3, activation='softmax')
])

# Compile model
model_rnn.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train model
model_rnn.fit(X_train, y_train, batch_size=32, epochs=5, validation_split=0.2)

# Evaluate model
rnn_score = model_rnn.evaluate(X_test, y_test) print(f'RNN
Test Accuracy: {rnn_score[1]}')
```

```
Epoch 1/5

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy:
0.4207 - loss: 1.0723 - val_accuracy: 0.6439 - val_loss: 0.8071
Epoch 2/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 26s 21ms/step - accuracy:
0.6973 - loss: 0.7150 - val_accuracy: 0.7307 - val_loss: 0.6462
Epoch 3/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 25s 20ms/step - accuracy:
0.8029 - loss: 0.4937 - val_accuracy: 0.7527 - val_loss: 0.6022
Epoch 4/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 28s 23ms/step - accuracy:
0.8285 - loss: 0.4310 - val_accuracy: 0.7565 - val_loss: 0.6082
Epoch 5/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 28s 23ms/step - accuracy:
0.8558 - loss: 0.3660 - val_accuracy: 0.7849 - val_loss: 0.5980
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2s 4ms/step - accuracy: 0.7835 -
loss: 0.5904
RNN Test Accuracy: 0.7811816334724426
```

```
from tensorflow.keras.layers import LSTM

# Define LSTM model
model_lstm = Sequential([
    Embedding(max_words, 128, input_length=max_len),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2), Dense(3,
    activation='softmax')
])
```

```
# Compile model
model_lstm.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train model
model_lstm.fit(X_train, y_train, batch_size=32, epochs=5, validation_split=0.2)

# Evaluate model
lstm_score = model_lstm.evaluate(X_test, y_test) print(f'LSTM Test
Accuracy: {lstm_score[1]}')

Epoch 1/5

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65s 52ms/step - accuracy:
0.5909 - loss: 0.8568 - val_accuracy: 0.7674 - val_loss: 0.5670
Epoch 2/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64s 52ms/step - accuracy:
0.8235 - loss: 0.4412 - val_accuracy: 0.8137 - val_loss: 0.4624
Epoch 3/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 65s 52ms/step - accuracy:
0.8774 - loss: 0.3050 - val_accuracy: 0.8280 - val_loss: 0.4563
Epoch 4/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64s 52ms/step - accuracy:
0.9028 - loss: 0.2387 - val_accuracy: 0.8350 - val_loss: 0.4450
Epoch 5/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 67s 55ms/step - accuracy:
0.9159 - loss: 0.1999 - val_accuracy: 0.8416 - val_loss: 0.4645
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 7s 17ms/step - accuracy: 0.8426 -
loss: 0.4547
LSTM Test Accuracy: 0.8409109115600586

from tensorflow.keras.layers import GRU

# Define GRU model
model_gru = Sequential([
      Embedding(max_words, 128, input_length=max_len),
      GRU(128, dropout=0.2, recurrent_dropout=0.2), Dense(3,
      activation='softmax')
])

# Compile model
model_gru.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])
```

```
# Train model
model_gru.fit(X_train, y_train, batch_size=32, epochs=5, validation_split=0.2)

# Evaluate model
gru_score = model_gru.evaluate(X_test, y_test) print(f'GRU
Test Accuracy: {gru_score[1]}')
```

```
Epoch 1/5

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 69s 55ms/step - accuracy:
0.5968 - loss: 5982.0239 - val_accuracy: 0.7256 - val_loss: 0.6641
Epoch 2/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 64s 52ms/step - accuracy:
0.8074 - loss: 1.5397 - val_accuracy: 0.7672 - val_loss: 0.5737
Epoch 3/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 66s 54ms/step - accuracy:
0.8577 - loss: 0.3662 - val_accuracy: 0.7787 - val_loss: 0.5450
Epoch 4/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71s 57ms/step - accuracy:
0.8792 - loss: 0.4725 - val_accuracy: 0.7838 - val_loss: 0.5363
Epoch 5/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72s 58ms/step - accuracy:
0.8887 - loss: 0.2734 - val_accuracy: 0.7915 - val_loss: 0.5285
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5s 14ms/step - accuracy: 0.7922 -
loss: 0.5158
GRU Test Accuracy: 0.7880703210830688
```

```
from tensorflow.keras.layers import Conv1D, GlobalMaxPooling1D

# Define TextCNN model
model_cnn = Sequential([
        Embedding(max_words, 128, input_length=max_len),
        Conv1D(128, 5, activation='relu'), GlobalMaxPooling1D(),
        Dense(128, activation='relu'),
        Dropout(0.2),
        Dense(3, activation='softmax')
])

# Compile model
model_cnn.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train model
his= model_cnn.fit(X_train, y_train, batch_size=32, epochs=15,
```

```
validation_split=0.2)

# Evaluate model
cnn_score = model_cnn.evaluate(X_test, y_test) print(f'CNN
Test Accuracy: {cnn_score[1]}')

Epoch 1/15

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19s 15ms/step - accuracy:
0.6106 - loss: 0.8323 - val_accuracy: 0.8084 - val_loss: 0.4833
Epoch 2/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 17s 14ms/step - accuracy:
0.8887 - loss: 0.2802 - val_accuracy: 0.8622 - val_loss: 0.3664
Epoch 3/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18s 15ms/step - accuracy:
0.9408 - loss: 0.1364 - val_accuracy: 0.8673 - val_loss: 0.3983
Epoch 4/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 17ms/step - accuracy:
0.9525 - loss: 0.1014 - val_accuracy: 0.8602 - val_loss: 0.4893
Epoch 5/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy:
0.9568 - loss: 0.0914 - val_accuracy: 0.8519 - val_loss: 0.5350
Epoch 6/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy:
0.9587 - loss: 0.0869 - val_accuracy: 0.8584 - val_loss: 0.5331
Epoch 7/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy:
0.9583 - loss: 0.0825 - val_accuracy: 0.8625 - val_loss: 0.5347
Epoch 8/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy:
0.9591 - loss: 0.0781 - val_accuracy: 0.8655 - val_loss: 0.5541
Epoch 9/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy:
0.9600 - loss: 0.0766 - val_accuracy: 0.8615 - val_loss: 0.6181
Epoch 10/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22s 18ms/step - accuracy:
0.9588 - loss: 0.0754 - val_accuracy: 0.8582 - val_loss: 0.6798
Epoch 11/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19s 16ms/step - accuracy:
0.9631 - loss: 0.0688 - val_accuracy: 0.8652 - val_loss: 0.6000
Epoch 12/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 19s 16ms/step - accuracy:
0.9627 - loss: 0.0692 - val_accuracy: 0.8629 - val_loss: 0.6606
Epoch 13/15
```

```
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 17ms/step - accuracy: 0.9633
-
```

```
loss: 0.0696 - val_accuracy: 0.8585 - val_loss: 0.7672 Epoch 14/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21s 17ms/step - accuracy:
0.9620 - loss: 0.0734 - val_accuracy: 0.8599 - val_loss: 0.6857
Epoch 15/15
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 20s 16ms/step - accuracy:
0.9633 - loss: 0.0698 - val_accuracy: 0.8656 - val_loss: 0.7628
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1s 4ms/step - accuracy: 0.8573 -
loss: 0.7632
CNN Test Accuracy: 0.8594699501991272
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D, MaxPooling1D,
LSTM, Dense, Dropout

# Define CNN-LSTM model
model_cnn_lstm = Sequential([
    Embedding(max_words, 128, input_length=max_len),
    Conv1D(128, 5, activation='relu'), MaxPooling1D(pool_size=2),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(3, activation='softmax')
])

# Compile model model_cnn_lstm.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train model
model_cnn_lstm.fit(X_train, y_train, batch_size=32, epochs=5, validation_split=0.2)

# Evaluate model
cnn_lstm_score = model_cnn_lstm.evaluate(X_test, y_test)
print(f'CNN-LSTM Test Accuracy: {cnn_lstm_score[1]}')
```

```
Epoch 1/5

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 50s 40ms/step - accuracy:
0.6034 - loss: 0.8402 - val_accuracy: 0.7888 - val_loss: 0.5276
Epoch 2/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49s 40ms/step - accuracy:
0.8521 - loss: 0.3685 - val_accuracy: 0.8245 - val_loss: 0.4482
Epoch 3/5
```

```
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49s 40ms/step - accuracy:
0.9133 - loss: 0.2083 - val_accuracy: 0.8368 - val_loss: 0.4675
Epoch 4/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 49s 40ms/step - accuracy:
0.9340 - loss: 0.1477 - val_accuracy: 0.8495 - val_loss: 0.4728
Epoch 5/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47s 38ms/step - accuracy:
0.9452 - loss: 0.1181 - val_accuracy: 0.8477 - val_loss: 0.5423
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4s 10ms/step - accuracy: 0.8491 -
loss: 0.5289
CNN-LSTM Test Accuracy: 0.8481238484382629
```

```
from tensorflow.keras.layers import GRU

# Define CNN-GRU model
model_cnn_gru = Sequential([
    Embedding(max_words, 128, input_length=max_len),
    Conv1D(128, 5, activation='relu'), MaxPooling1D(pool_size=2),
    GRU(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(3, activation='softmax')
])

# Compile model model_cnn_gru.compile(loss='sparse_categorical_crossentropy',
optimizer='adam', metrics=['accuracy'])

# Train model
model_cnn_gru.fit(X_train, y_train, batch_size=32, epochs=5, validation_split=0.2)

# Evaluate model
cnn_gru_score = model_cnn_gru.evaluate(X_test, y_test)
print(f'CNN-GRU Test Accuracy: {cnn_gru_score[1]}')
```

```
Epoch 1/5

/Library/Frameworks/Python.framework/Versions/3.11/lib/python3.11/ site-
packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument
`input_length` is deprecated. Just remove it.
  warnings.warn(

1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 52s 37ms/step - accuracy:
0.6076 - loss: 0.9894 - val_accuracy: 0.7708 - val_loss: 0.5843
Epoch 2/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy:
0.8637 - loss: 0.3476 - val_accuracy: 0.8223 - val_loss: 0.4380
Epoch 3/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 46s 37ms/step - accuracy:
0.9208 - loss: 0.1917 - val_accuracy: 0.8389 - val_loss: 0.4568
Epoch 4/5
```

```
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 44s 36ms/step - accuracy: 0.9376 - loss: 0.1492 - val_accuracy: 0.8391 - val_loss: 0.4902
Epoch 5/5
1234/1234 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 39s 31ms/step - accuracy: 0.9464 - loss: 0.1184 - val_accuracy: 0.8385 - val_loss: 0.5416
386/386 ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3s 7ms/step - accuracy: 0.8397 - loss: 0.5155
CNN-GRU Test Accuracy: 0.8407488465309143
```

```
import pandas as pd

# Load the uploaded dataset
file_path = '/Users/deveshdhyani/Documents/Disertation/Final_Draft/data/Combined_Data.csv'
data = pd.read_csv(file_path)

# Display the first few rows of the dataset
data.head()
```

[12] Pytho

	Unnamed: 0	statement	status
0	0	oh my gosh	Anxiety
1	1	trouble sleeping, confused mind, restless hear...	Anxiety
2	2	All wrong, back off dear, forward doubt. Stay ...	Anxiety
3	3	I've shifted my focus to something else but I'...	Anxiety
4	4	I'm restless and restless, it's been a month n...	Anxiety

```
print(data.describe())
```

[4] Pytho

```
         Unnamed: 0
count  53043.000000
mean   26521.000000
std    15312.339501
min        0.000000
25%    13260.500000
50%    26521.000000
75%    39781.500000
max    53042.000000
```

+ Code + Markdown

[ ] Pytho

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report, accuracy_score

# Assume 'data' is already defined and contains the 'statement' and 'status' columns

# Check for missing or invalid values in the 'statement' column
invalid_entries = data['statement'].isnull().sum()

# Drop rows with missing or invalid values in the 'statement' column
data_cleaned = data.dropna(subset=['statement'])

# Extract features and labels again after cleaning
X_cleaned = data_cleaned['statement']
y_cleaned = data_cleaned['status']
```

```python
# Split the cleaned data into training and test sets
X_train_cleaned, X_test_cleaned, y_train_cleaned, y_test_cleaned = train_test_split(X_cleaned, y_cleaned, test_size=0.2, random_state=42)

# Initialize TfidfVectorizer
tfidf = TfidfVectorizer()

# Transform the cleaned text data to TF-IDF features
X_train_tfidf_cleaned = tfidf.fit_transform(X_train_cleaned)
X_test_tfidf_cleaned = tfidf.transform(X_test_cleaned)

# Initialize and train the Extra Trees Classifier on cleaned data
clf_cleaned = ExtraTreesClassifier(n_estimators=100, random_state=42)
clf_cleaned.fit(X_train_tfidf_cleaned, y_train_cleaned)

# Predict on the cleaned test set
y_pred_cleaned = clf_cleaned.predict(X_test_tfidf_cleaned)

# Evaluate the model on cleaned data
accuracy_cleaned = accuracy_score(y_test_cleaned, y_pred_cleaned)
report_cleaned = classification_report(y_test_cleaned, y_pred_cleaned)

accuracy_cleaned, report_cleaned
```

[2] Pytho

```
(0.6704944481351428,
 '              precision    recall  f1-score   support\n\n           Anxiety       0.94      0.43      0.59       755\n           Bipolar       1.00      0.31
```

```python
from imblearn.over_sampling import SMOTE

# Apply SMOTE to the cleaned dataset
smote = SMOTE(random_state=42)
X_tfidf_cleaned = tfidf.fit_transform(X_cleaned)
X_smote, y_smote = smote.fit_resample(X_tfidf_cleaned, y_cleaned)

# Split the augmented data into training and test sets
X_train_smote, X_test_smote, y_train_smote, y_test_smote = train_test_split(X_smote, y_smote, test_size=0.2, random_state=42)

# Initialize and train the Extra Trees Classifier on SMOTE data
clf_smote = ExtraTreesClassifier(n_estimators=50, random_state=42)
clf_smote.fit(X_train_smote, y_train_smote)

# Predict on the SMOTE test set
y_pred_smote = clf_smote.predict(X_test_smote)

# Evaluate the model with SMOTE data
accuracy_smote = accuracy_score(y_test_smote, y_pred_smote)
report_smote = classification_report(y_test_smote, y_pred_smote)

print(accuracy_smote)
print(report_smote)
```

[5] Pytho

```
0.8991302827673616
              precision    recall  f1-score   support
```

```
              Anxiety       0.97      0.97      0.97      3375
              Bipolar       0.99      0.97      0.98      3214
           Depression       0.71      0.69      0.70      3274
               Normal       0.86      0.94      0.90      3325
Personality disorder       1.00      1.00      1.00      3241
               Stress       0.99      0.96      0.98      3209
             Suicidal       0.76      0.76      0.76      3243

             accuracy                           0.90     22881
            macro avg       0.90      0.90      0.90     22881
         weighted avg       0.90      0.90      0.90     22881
```

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.metrics import classification_report, accuracy_score

# Load the uploaded dataset
file_path = '/Users/deveshdhyani/Documents/Disertation/Final_Draft/data/Combined_Data.csv'
data = pd.read_csv(file_path)

# Remove rows with missing 'statement' values
data_cleaned = data.dropna(subset=['statement'])

# Sample the data
sampled_data = data_cleaned.sample(frac=0.1, random_state=42)

# Extract features and labels
X_sampled = sampled_data['statement']
y_sampled = sampled_data['status']

# Combine X and y into a single DataFrame for oversampling
oversampling_data = pd.concat([X_sampled, y_sampled], axis=1)

# Find the maximum class count
max_class_count = oversampling_data['status'].value_counts().max()

# Oversample the dataset
oversampled_data = oversampling_data.groupby('status').apply(lambda x: x.sample(max_class_count, replace=True, random_state=42)).reset_index(drop=True)

# Separate the oversampled data into features and labels
X_oversampled = oversampled_data['statement']
y_oversampled = oversampled_data['status']

# Split the oversampled data into training and testing sets
X_train_over, X_test_over, y_train_over, y_test_over = train_test_split(X_oversampled, y_oversampled, test_size=0.2, random_state=42)

# Vectorize the text data using TF-IDF
tfidf_vectorizer_over = TfidfVectorizer(max_features=1000)
X_train_over_tfidf = tfidf_vectorizer_over.fit_transform(X_train_over)
X_test_over_tfidf = tfidf_vectorizer_over.transform(X_test_over)
```

```python
tfidf_vectorizer_over = TfidfVectorizer(max_features=1000)
X_train_over_tfidf = tfidf_vectorizer_over.fit_transform(X_train_over)
X_test_over_tfidf = tfidf_vectorizer_over.transform(X_test_over)

# Train the Extra Trees Classifier with reduced number of estimators
model_over = ExtraTreesClassifier(n_estimators=250, random_state=42)
model_over.fit(X_train_over_tfidf, y_train_over)

# Make predictions and evaluate the model
y_over_pred = model_over.predict(X_test_over_tfidf)

accuracy_over = accuracy_score(y_test_over, y_over_pred)
classification_rep_over = classification_report(y_test_over, y_over_pred)

accuracy_over, classification_rep_over
```

[3] Pytho

```
/var/folders/z7/w5bn8bvd4gn6ncplr933nsmc0000gn/T/ipykernel_35515/1737633183.py:29: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior i
  oversampled_data = oversampling_data.groupby('status').apply(lambda x: x.sample(max_class_count, replace=True, random_state=42)).reset_index(drop=True)
```

```
(0.9611566681053086,
 '                     precision    recall  f1-score   support\n\n             Anxiety       0.98      0.99      0.99       348\n             Bipolar       0.99      0.99
```

```python
print(classification_rep_over)
```

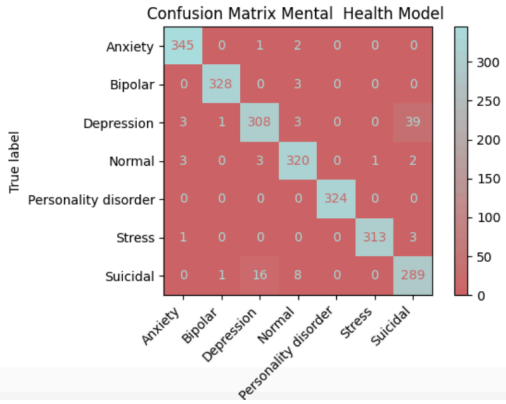
[4] Pytho

```
                     precision    recall  f1-score   support

             Anxiety       0.98      0.99      0.99       348
             Bipolar       0.99      0.99      0.99       331
          Depression       0.94      0.87      0.90       354
              Normal       0.95      0.97      0.96       329
Personality disorder       1.00      1.00      1.00       324
              Stress       1.00      0.99      0.99       317
            Suicidal       0.87      0.92      0.89       314

            accuracy                           0.96      2317
           macro avg       0.96      0.96      0.96      2317
        weighted avg       0.96      0.96      0.96      2317
```

```python
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt
from matplotlib.colors import LinearSegmentedColormap

# Generate the confusion matrix
conf_matrix = confusion_matrix(y_test_over, y_over_pred)

#custom_colors = ['#cc6666', '#dd9966', '#eedd88', '#eeeeaa', '#cceecc', '#aadddd', '#6699cc']

# Define the custom color #f46d43 and create a colormap based on it
custom_cmap = LinearSegmentedColormap.from_list('custom_cmap', ['#cc6666', '#aadddd'])

# Plot the confusion matrix with the custom colormap
plt.figure(figsize=(10, 8))
disp = ConfusionMatrixDisplay(confusion_matrix=conf_matrix, display_labels=model_over.classes_)
disp.plot(cmap=custom_cmap, values_format='g')

# Rotate the x-axis labels for better alignment
plt.xticks(rotation=45, ha='right')
plt.title('Confusion Matrix Mental  Health Model ')

# Display the plot
plt.tight_layout()
plt.show()

# Save the confusion matrix as an image (optional)
# plt.savefig('/path_to_save/confusion_matrix.png')
```

[15]

```
<Figure size 1000x800 with 0 Axes>
```



```
r == 'Suicidal'))
```

```
ion') & (misclassified_data['Predicted Label'] == 'Suicidal') )| ((misclassified_data['True Label'] == 'Suicidal') & (misclassified_data['Predicted Label'] == 'Depression')
```

[18] Pythc

...

	True Label	Predicted Label	Statement
10034	Suicidal	Depression	I do not know why I am posting this, especially on my main account, whether it is to find comfort with potentially like-minded people, to just throw it out into the anonymous void that is the internet, or just to vent further but I truly feel in my heart if I had a gun or maybe a rope, just some means where I KNOW I would be dead while also not risking the lives of other people like I potentially would be if I jumped off the nearest bridge, I would be gone. My urge is strong, stronger than it is ever been, I feel hopeless and unable to get help, but I have also failed enough in life and the last thing I want to do is fail at ending it too.I used the Lifeline Chat, and I know some people have had bad experiences with both the hotline and the chat, but I do think the person I got was trying to help as best they could, and maybe I will try to use the resources they presented me with to try and get on the road to recovery, I just do not know what that road even is anymore nor where it might take me.I guess the Sticky Notes version of my life is: Grew up in a somewhat dysfunctional family. Mom died young, after I graduated, and it messed me up, Dad developed Dementia and I was essentially forced to take care of him with little support until I finally broke down and he had to go to a nursing home. Pretty isolated from the rest of the family with few remaining friends. I was in a good position at work I guess but my life is literally my miserable job then nothing else meaningful and I am finding it harder to tough through my mental blocks and have been missing work, dying up my bank account and am close to homeless. I truly do not know nor understand what exactly is wrong with me. No car (Never learned to actually drive) and have to walk everywhere. Therapy would cost money but juggling therapy with my job so I could afford it seems impractical in my position. Maybe I am just too lazy to make it work? I dunno. I could go on but feel I have rambled long enough.Thanks for reading... Whatever this is. If I had a gun right now, I would be dead.
4401	Depression	Suicidal	I have no motivation to do things after work or on weekends anymore. Whenever I get home from work, the next work day cannot come quick enough. Whenever the weekend starts, I cannot wait for it to finish.I use to hate work and look forward to free time. But I have found a job I like and it seems to have shown me how much I dislike my personal life/free time. Talking to friends helps, but I cannot take all of their free time up. How do I enjoy going to work and look forward to coming home? I look forward to work because it distracts me from my personal life.
4347	Depression	Suicidal	Just felt like I needed to get that off my chest. I am grateful for the lack of suicidal ideation today, but I have been feeling ugh since yesterday. Just been so busy on top of long work shifts (12 hours). So, yeah, thanks for letting me yell this into the void. Needed it. Feeling like I want to die, but not feeling suicidal.
4680	Depression	Suicidal	Clearly on my mind more than normal. do not really have any motivation to keep on going. Boyfriend mentioned I joke about killing myself a lot recently
3717	Depression	Suicidal	I lay in bed for hours to the point of back pain. I get up for work 30 minutes before I have to clock in. I have no motivation to do anything on my days off, I literally stare at my computer screen, and have to force myself to do something. The bare minimum. More and more I find myself wanting to end it. Would that be so bad? I ask myself. I am lonely, always broke, hopeless and disgusting looking, and half blind. I know nobody could ever actually want me. I have a decent paying job but am always playing catch up financially.In the end I resolve not to kill myself, though I do not want to continue living. I would not be upset if something killed me. No I think Id be grateful. I am always tired.
4366	Depression	Suicidal	No matter what I do I only feel like I can sort of distract myself from it, but then I will remember that it is there it is just always there
11281	Suicidal	Depression	I do not know why I am posting this, especially on my main account, whether it is to find comfort with potentially like-minded people, to just throw it out into the anonymous void that is the internet, or just to vent further but I truly feel in my heart if I had a gun or maybe a rope, just some means where I KNOW I would be dead while also not risking the lives of other people like I potentially would be if I jumped off the nearest bridge, I would be gone. My urge is strong, stronger than it is ever been, I feel hopeless and unable to get help, but I have also failed enough in life and the last thing I want to do is fail at ending it too.I used the Lifeline Chat, and I know some people have had bad experiences with both the hotline and the chat, but I do think the person I got was trying to help as best they could, and maybe I will try to use the resources they presented me with to try and get on the road to recovery, I just do not know what that road even is anymore nor where it might take me.I guess the Sticky Notes version of my life is: Grew up in a somewhat dysfunctional family. Mom died young, after I graduated, and it messed me up, Dad developed Dementia and I was essentially forced to take care of him with little support until I finally broke down and he had to go to a nursing home. Pretty isolated from the rest of the family with few remaining friends. I was in a good position at work I guess but my life is literally my miserable job then nothing else meaningful and I am finding it harder to tough through my mental blocks and have been missing work, dying up my bank account and am close to homeless. I truly do not know nor understand what exactly is wrong with me. No car (Never learned to actually drive) and have to walk everywhere. Therapy would cost money but juggling therapy with my job so I could afford it seems impractical in my position. Maybe I am just too lazy to make it work? I dunno. I could go on but feel I have rambled long enough.Thanks for reading... Whatever this is. If I had a gun right now, I would be dead.
3941	Depression	Suicidal	I always only care for myself and the only time I get along with my siblings is if I need a favor.Its hard to be happy and well I am always isolating myself from everyone.My brother in law plays with my siblings something I do not do.Makes me feel bad but in the mental state I am in I do not care about socializing with anyone.Just want to lay down all day do nothing.I am not responsible like I should be and get mad when I have to do things and do not like getting told to do things.My brother lives by the he way if your wondering why I am mentioning him.I do not care about how anyone in my family Just myself and brother in law always makes sure everyone is okay.I just want to move out because hes basically my replacement and I feel like I because problems Anyways.I feel upset and mostly mad all the time but want to move out but have no job. My brother in law is a better brother to my siblings than I am
4685	Depression	Suicidal	I am really scared to tell him because what if he will take it badly or something. I have know him for around 13 years and I really do not want to lose him to something stupid like this. We never talk about sensitive stuff like this or anything I have never told anyone irl about this before. How do I tell one of my closest friends that I used to cut
10779	Suicidal	Depression	This motherfucker has only been working here for 1 week. I tried to give him tips on how to work efficiently and to make future work easier for both him and others, but no. Hes incredibly arrogant and stubborn. Apparently hes been spreading rumors that I take 40 minutes breaks while I was on my off days. This fucking Piece of filth is going after my job. My fucking livelihood. What I use to pay my bills. Hes been here one week. Hes been getting into it with other workers too. But you are NOT GOING TO GO AFTER MY INCOME. I WILL NOT SIT QUIETLY AND LET SOME FUCKING NEWBIE RUIN MY LIFE. THIS PIECE OF SHIT HAS NO GOD DAMN CLUE WHAT I have BEEN THROUGH AT THIS PLACE. HOW HARD I FUCKING WORK AT THIS GOD FORSAKEN PLACE. Sorry. But I have been holding that in all day ever since I found out. Thankfully my supervisor knows how hard I work, and trusts me. All I try to

EDA

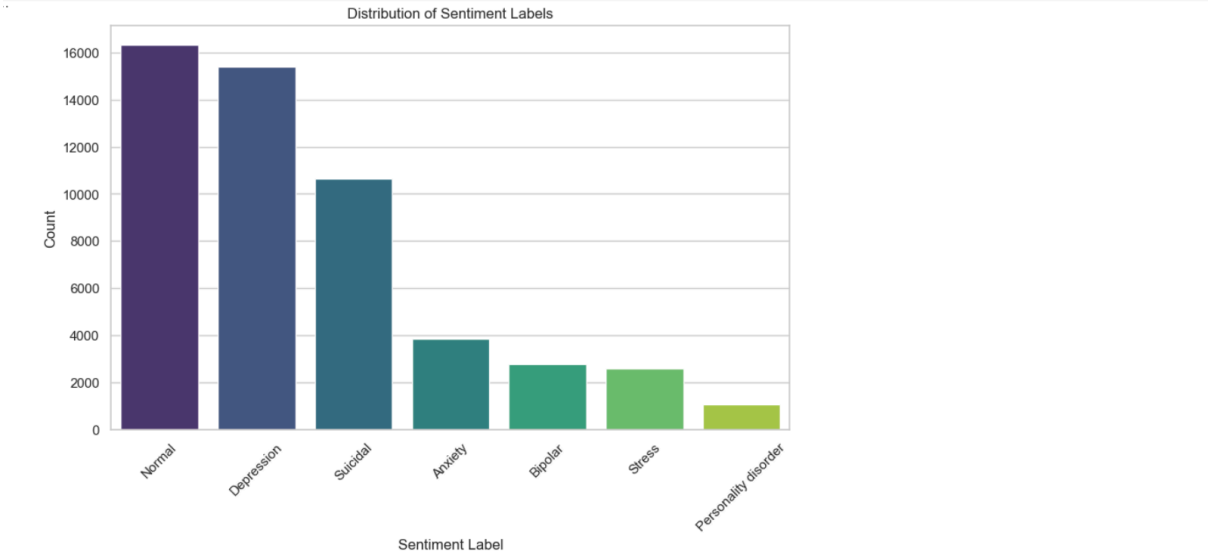
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
file_path = '/Users/deveshdhyani/Documents/Disertation/Final_Draft/data/Combined_Data.csv'
data = pd.read_csv(file_path)

# Dropping the unnecessary column 'Unnamed: 0'
data_cleaned = data.drop(columns=['Unnamed: 0'])

# Dropping rows with missing 'statement' values
data_cleaned = data_cleaned.dropna(subset=['statement'])

# Calculate the length of each statement
data_cleaned['statement_length'] = data_cleaned['statement'].apply(len)
```

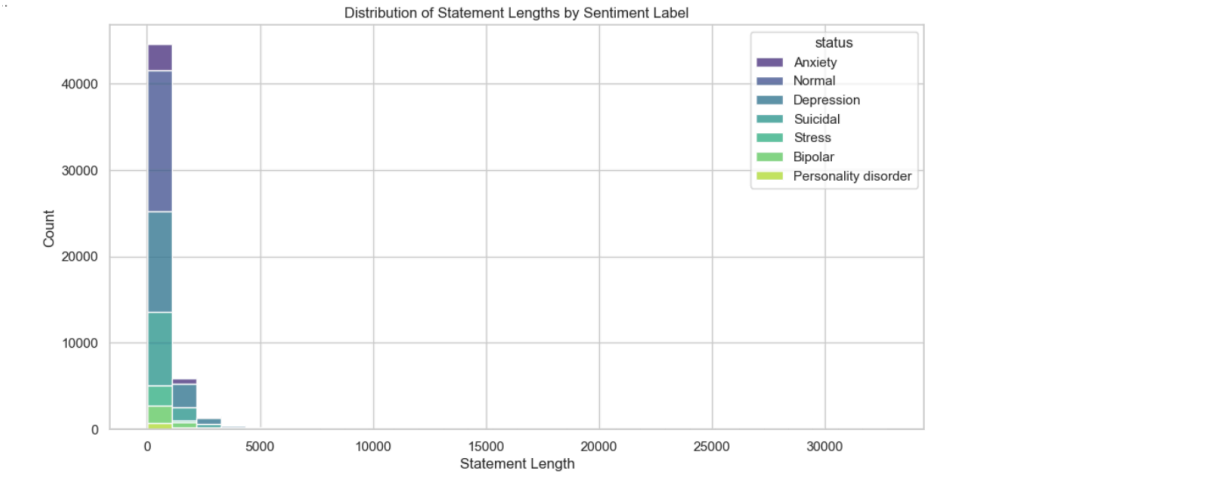
[1] Pyth

```
# Set the style for the plots
sns.set(style="whitegrid")

# Plot 1: Distribution of Sentiment Labels
plt.figure(figsize=(10, 6))
sns.countplot(data=data_cleaned, x='status', order=data_cleaned['status'].value_counts().index, palette='viridis')
plt.title('Distribution of Sentiment Labels')
plt.xlabel('Sentiment Label')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()

# Plot 2: Distribution of Statement Lengths by Sentiment Label
plt.figure(figsize=(12, 6))
sns.histplot(data=data_cleaned, x='statement_length', hue='status', multiple='stack', palette='viridis', bins=30)
plt.title('Distribution of Statement Lengths by Sentiment Label')
plt.xlabel('Statement Length')
plt.ylabel('Count')
#plt.xlim(0, 6000)
plt.show()
```

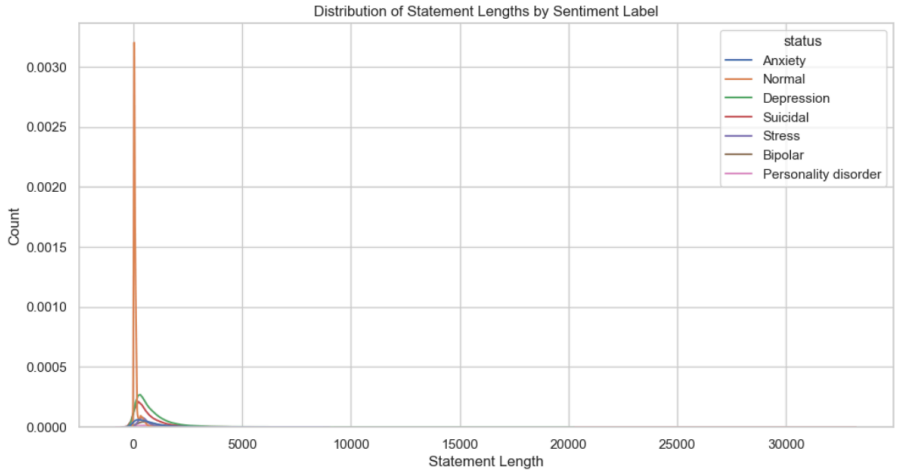
[16] Pyth

```
/var/folders/z7/w5bn8bvd4gn6ncplr933nsmc0000gn/T/ipykernel_66815/2442914217.py:6: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(data=data_cleaned, x='status', order=data_cleaned['status'].value_counts().index, palette='viridis')
```


Distribution of Sentiment Labels
Count
16000
14000
12000
10000
8000
6000
4000
2000
0
Normal
Depression
Suicidal
Anxiety
Bipolar
Stress
Personality disorder
Sentiment Label


Distribution of Statement Lengths by Sentiment Label
status
Anxiety
Normal
Depression
Suicidal
Stress
Bipolar
Personality disorder
Count
40000
30000
20000
10000
0
0
5000
10000
15000
20000
25000
30000
Statement Length

```
# Plot the histogram with a limit on the x-axis
plt.figure(figsize=(12, 6))
sns.kdeplot(data=data_cleaned, x='statement_length', hue='status')
plt.title('Distribution of Statement Lengths by Sentiment Label')
plt.xlabel('Statement Length')
plt.ylabel('Count')
plt.xlim(0, 10000)  # Set x-axis limit to 10,000
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
from collections import Counter
import nltk
import string

# Load the dataset
file_path = '/Users/deveshdhyani/Documents/Disertation/Final_Draft/data/Combined_Data.csv'
data = pd.read_csv(file_path)

# Dropping the unnecessary column 'Unnamed: 0'
data_cleaned = data.drop(columns=['Unnamed: 0'])

# Dropping rows with missing 'statement' values
data_cleaned = data_cleaned.dropna(subset=['statement'])

# Download NLTK stopwords
nltk.download('stopwords')
nltk.download('punkt')

# Define stop words and punctuation
stop_words = set(stopwords.words('english'))
```

```
nltk.download('stopwords')
nltk.download('punkt')

# Define stop words and punctuation
stop_words = set(stopwords.words('english'))
punctuation = set(string.punctuation)

# Preprocessing function to clean the text
def preprocess_text(text):
    # Convert to lowercase
    text = text.lower()
    # Tokenize the text
    words = word_tokenize(text)
    # Remove stop words and punctuation
    words = [word for word in words if word not in stop_words and word not in punctuation]
    return words

# Apply preprocessing to the 'statement' column
data_cleaned['processed_statement'] = data_cleaned['statement'].apply(preprocess_text)
```

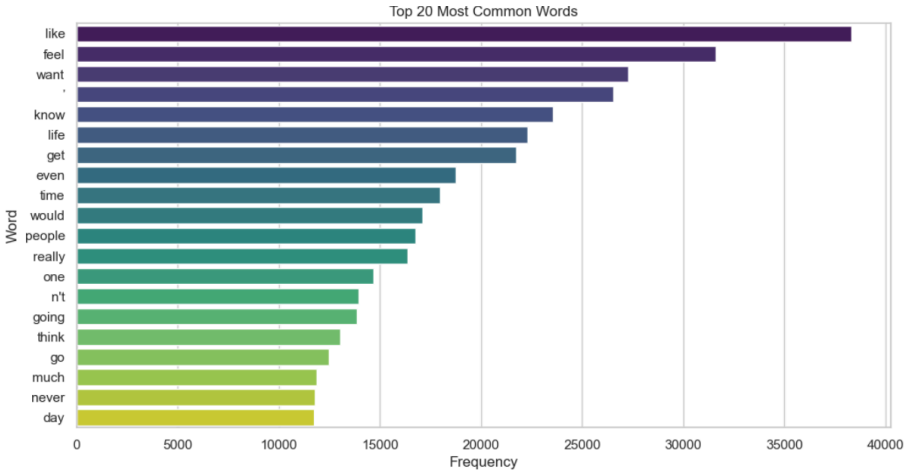
[2]

```
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/deveshdhyani/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/deveshdhyani/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
# Flatten the list of words into a single list
all_words = [word for statement in data_cleaned['processed_statement'] for word in statement]

# Calculate word frequencies
word_freq = Counter(all_words)

# Get the 20 most common words
common_words = word_freq.most_common(20)

# Convert to DataFrame for easier plotting
common_words_df = pd.DataFrame(common_words, columns=['word', 'frequency'])

# Plot the most common words
plt.figure(figsize=(12, 6))
sns.barplot(x='frequency', y='word', data=common_words_df, palette='viridis')
plt.title('Top 20 Most Common Words')
plt.xlabel('Frequency')
plt.ylabel('Word')
plt.show()
```

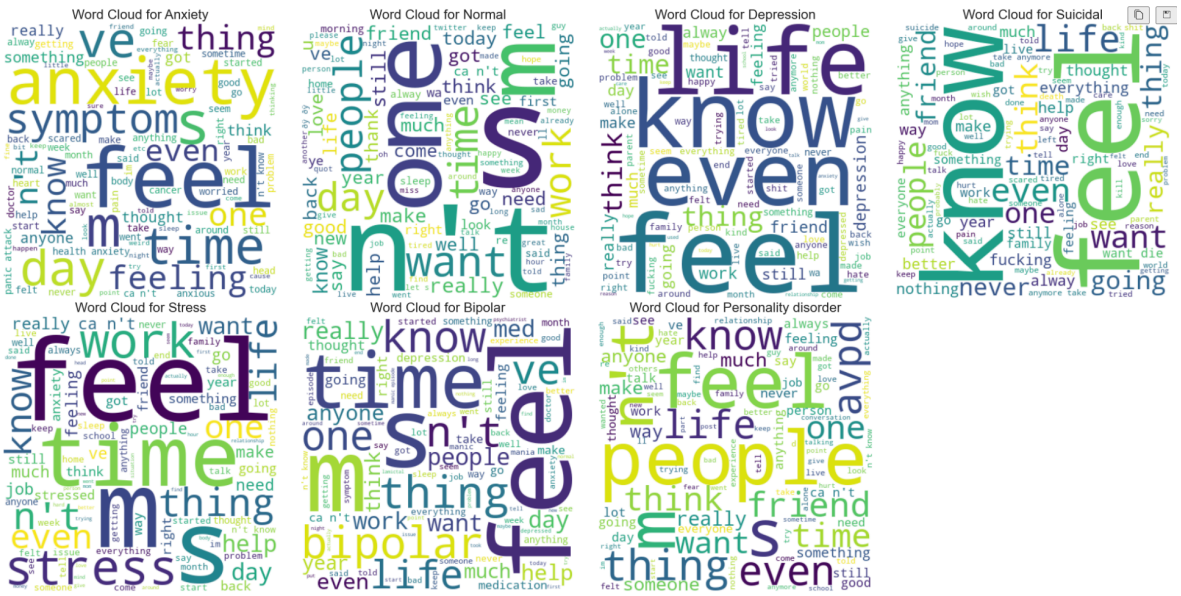
[3]

```
/var/folders/z7/w5bn8bvd4gn6ncplr933nsmc0000gn/T/ipykernel_87336/45518105.py:15: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same effect.

  sns.barplot(x='frequency', y='word', data=common_words_df, palette='viridis')
```




```
# Function to generate a word cloud for a given text
def generate_word_cloud(text):
    return WordCloud(width=400, height=400, background_color='white', max_words=100, colormap='viridis').generate(text)

# Generate word clouds for each sentiment label
labels = data_cleaned['status'].unique()

# Prepare to plot in a grid
fig, axes = plt.subplots(nrows=2, ncols=4, figsize=(20, 10))

# Flatten axes for easy iteration
axes = axes.flatten()

# Generate and plot word clouds for each label
for i, label in enumerate(labels):
    label_text = ' '.join([' '.join(statement) for statement in data_cleaned[data_cleaned['status'] == label]['processed_statement']])
    wordcloud = generate_word_cloud(label_text)

    axes[i].imshow(wordcloud, interpolation='bilinear')
    axes[i].axis('off')
    axes[i].set_title(f'Word Cloud for {label}', fontsize=16)

# Adjust layout and remove any unused subplots
for j in range(len(labels), len(axes)):
    fig.delaxes(axes[j])

plt.tight_layout()
plt.show()
```

```
plt.show()
```



```
status_counts = data['status'].value_counts()
```

Pyth

```
import matplotlib.pyplot as plt

# Define the custom colors
custom_colors = ['#cc6666', '#dd9966', '#eedd88', '#eeeeaa', '#cceecc', '#aadddd', '#6699cc']


# Plotting the bar chart for the entire dataset's status distribution
plt.figure(figsize=(10, 6))
plt.bar(status_counts.index, status_counts.values, color=custom_colors)
plt.xlabel('Status')
plt.ylabel('Count')
plt.title('Distribution of Status in the Entire Dataset')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

Pyth