Group Members: Anita Gee, Siwei Guo, Yingzhu Chen, Nemo (Sorachat) Chavalvechakul

Group Name: Mental Health Warriors

Data Cleaning and Text Data Preprocessing:

1. Deletion of Duplicate Rows/Columns:

- In our dataset, identifying and removing duplicate entries was paramount to ensuring the integrity and quality of our data. Duplicate rows can lead to biased model training by over-representing certain observations, which might result in overfitting. A systematic approach was employed to scan the dataset for any redundancies, utilizing hashing techniques to ensure efficiency and accuracy in the detection process. Once identified, these duplicates were removed to maintain the uniqueness of our dataset, thereby enhancing the reliability of our subsequent analyses.

2. Text Data Preprocessing:

- The textual data within our dataset required extensive preprocessing to transform raw text into a more analyzable and standardized format. The following steps were meticulously implemented:
    - **Stop Word Removal: We did not remove stop words since they capture feelings and emotions.** For example, “I feel so lost” carries more meaning than “feel lost”. We can also see from the word clouds that there are not many stop words that are in bold so it does not seem to impact the data.
    - **Case Normalization:** All text was converted to lowercase to treat words such as "The" and "the" as identical, reducing the complexity of the data.
    - **Punctuation Removal:** All punctuation marks were removed to prevent the model from misinterpreting them as part of the words.
    - **Stemming:** Words were reduced to their base or root form, simplifying the vocabulary of our models and enhancing the processing speed.
    - **Whitespace Removal:** Extra spaces, including tabs and newline characters, urls and special characters, were eliminated to ensure consistent data entry formats and create cleaner data.

These preprocessing steps were critical in preparing the textual data for further analysis and modeling. By standardizing the text, we not only improve the efficiency of our algorithms but also enhance their ability to discern patterns and make predictions based on linguistic content. This foundational work is essential for the robust performance of our subsequent NLP (Natural Language Processing) tasks, aiming to accurately classify mental health statuses from textual data.

3. Outlier treatment and Remove columns not needed or useful for modeling

During this process, we evaluated the distribution of statement lengths and identified potential outliers using the interquartile range. While we calculated bounds for outliers, we intentionally kept all data points, including extreme values, to preserve critical emotional context.

We decided to keep the outliers because sentiment analysis often involves extreme reviews, such as very short or very long statements, which can carry strong emotions that are valuable for classification. These extreme statements might provide unique insights that are especially useful when working with non-transformer models. However, for transformer-based models, we can shorten the word length input or use a filtered dataset (with outliers removed) to reduce computational time without significantly affecting the model's performance.

We focused on keeping only the relevant data for modeling, ensuring that the dataset contained the necessary features. While we did not remove any specific columns in this process, we did drop duplicate entries based on the 'statement' column to ensure that each text entry was unique. This step is important as it helps maintain the integrity of the dataset by eliminating redundancy, making it more manageable for the model's learning process.

4. Label encoding to codify categorical variables into numbers

This is when the statements are split into words/tokens. Tokenization is essential in NLP as it breaks text into smaller units (tokens), making it processable by models like Naïve Bayes, XGBoost, and Transformers.

This step was used to convert categorical target variables into numerical values. This is a crucial step for machine learning models. Most machine learning algorithms, including XGBoost, Naïve Bayes, and Support Vector Machines, require numerical input, so encoding the target variable is essential for compatibility with these models. By using `LabelEncoder`, each unique category in the target variable (`y`) was mapped to a numerical value. In other words, we are using this step to transfer our data into a language that the computer can understand.

For example, categories like "positive," "neutral," and "negative" might be encoded as 2, 1, and 0, respectively. This transformation ensures that the model can effectively process categorical data and make predictions, as models typically struggle with non-numeric data.

5. Tokenization

Tokenization is a fundamental step in natural language processing (NLP), where text is split into smaller units, typically words or tokens, to prepare it for modeling. In this case, We applied NLTK's word_tokenize() to split statements into individual tokens, ensuring a granular analysis of linguistic patterns. Following this, Porter stemming was implemented to reduce inflectional forms—for example, converting "running" to "run" and "happily" to "happi."

Although stemming can sometimes produce non-dictionary terms (e.g., "fli" from "flies"), it still effectively compressed lexical variants, reducing feature space dimensionality. More importantly, stop words were retained after analysis revealed their contextual importance in mental health narratives (e.g., "I don't feel" vs. "feel").

Tokenization is essential because it transforms unstructured text into a structured format that models can process. This step allows us to convert raw text into a series of meaningful components (tokens), which are crucial for tasks like text classification or sentiment analysis. Tokenization helps to standardize the input text, improving the model's ability to detect patterns and relationships between words.

6. TF-IDF

We used the Frequency-Inverse Document Frequency (TF-IDF) method to convert the tokens into numerical features, capturing the importance of each word in relation to the entire document set. By using both unigrams and bigrams, this method captures a broader range of textual information, allowing the model to understand phrase-level context.

The TfidfVectorizer was configured with a maximum of 50,000 features, ensuring that the most discriminative terms were prioritized while maintaining computational efficiency. This transformation resulted in a sparse matrix where words like "self-harm" were assigned higher weights compared to more common terms like "feel," highlighting significant but rare terms.

In addition to TF-IDF features, we incorporated engineered numerical features, such as character counts and sentence lengths, which help capture structural patterns within the text. As previously mentioned, we noticed a pattern between statement length and emotions severeness. We want to use this combination of linguistic and numerical features to improve model accuracy by providing a more holistic representation of the data.

7. Oversampling

Class imbalance occurs when the distribution of categories in the target variable is skewed, which can lead to biased predictions from the model. By oversampling the minority class, we ensured that the model would be exposed to more examples of the underrepresented class, thereby improving its ability to learn and predict both classes effectively.

To address severe class imbalance (e.g., 16,343 "normal" vs. 1,077 "personality disorder" instances) problems in our dataset, we applied random oversampling to the training data. This technique duplicated minority-class samples rather than synthesizing new data, ensuring authentic representation of rare emotional states. After oversampling, all seven classes achieved parity in sample counts (12,000+ instances each), enabling the model to learn distinctive patterns without majority-class bias.

This approach helps improve performance metrics such as recall and F1-score, particularly for the minority class, leading to more robust and accurate model predictions. More importantly, oversampling was applied only to the training set post-train/validation split to prevent data leakage—validation and test sets retained original distributions for realistic performance evaluation.