**Stress Identification from Five Different Categories of Reddit Communities using ML**

Abstract:

This project presents a comprehensive dataset comprising lengthy multi-domain social media content extracted from five distinct categories of Reddit communities. The primary objective is to identify and analyze stress patterns within these communities using Natural Language Processing (NLP) techniques. The dataset encompasses diverse linguistic expressions related to stress, providing a rich resource for training and evaluating machine learning models. The project involves data preprocessing, feature extraction, and the application of NLP algorithms to gain insights into stress prevalence across different Reddit community categories. The findings aim to contribute valuable insights into understanding stress dynamics within online social platforms, with implications for mental health and community management. Common algorithms suitable for Natural Language Processing (NLP) tasks like this Logistic regression, KNN, Random forest , Decision Tree, Naive bayes algorithm .The evaluation metric considered includes accuracy score and classification report.

**Objective:**

The objective is to create a dataset from various Reddit communities to identify stress patterns using Natural Language Processing (NLP). This dataset will help understand stress prevalence across different community categories and aid in improving mental health support and community management strategies.

**Dataset:**

This datset includes 2838 rows and 116 feature variables. Each row in the dataset corresponds to a single post or comment from the social media data extracted from the Reddit communities.

The index represents the columns/features present in the dataset as follows:

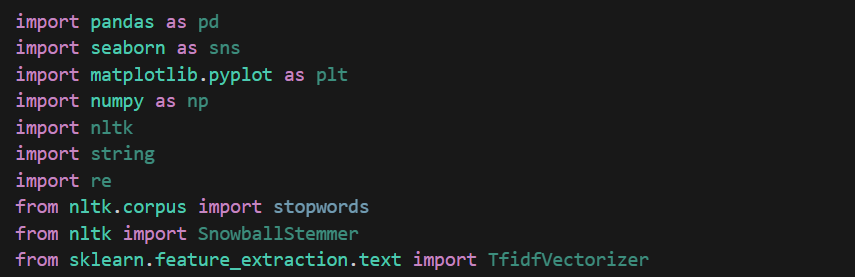
* **subreddit**: The name of the Reddit community where the post/comment was made.
* **post\_id**: The unique identifier for the post/comment.
* **sentence\_range**: The range of sentences within the post/comment that are relevant.
* **text**: The textual content of the post/comment.
* **id**: Identifier for the data entry.
* **label**: Label indicating whether the post/comment is related to stress or not.
* **confidence**: Confidence level associated with the label.
* **social\_timestamp**: Timestamp of the post/comment.
* **social\_karma**: Karma associated with the post/comment.
* **syntax\_ari**: Automated Readability Index (ARI) score indicating the readability of the text.
* **lex\_dal\_min\_pleasantness**: Minimum pleasantness score based on lexical analysis.
* **lex\_dal\_min\_activation**: Minimum activation score based on lexical analysis.
* **lex\_dal\_min\_imagery**: Minimum imagery score based on lexical analysis.
* **lex\_dal\_avg\_activation**: Average activation score based on lexical analysis.
* **lex\_dal\_avg\_imagery**: Average imagery score based on lexical analysis.
* **lex\_dal\_avg\_pleasantness**: Average pleasantness score based on lexical analysis.
* **social\_upvote\_ratio**: Upvote ratio of the post/comment.
* **social\_num\_comments**: Number of comments on the post/comment.
* **syntax\_fk\_grade**: Flesch-Kincaid Grade Level indicating the readability of the text.
* **sentiment**: Sentiment analysis score indicating the sentiment of the text.

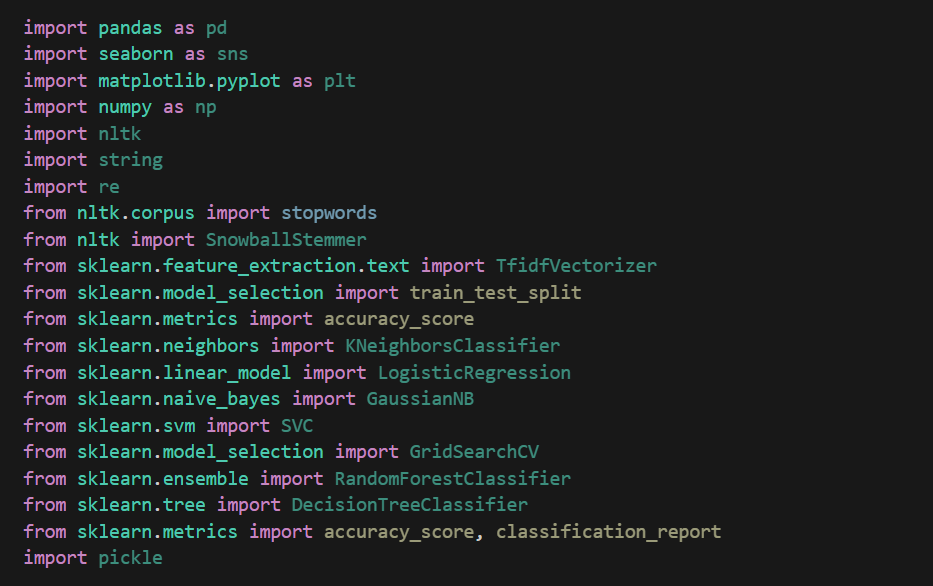
Source of the dataset is from a : Reddit dataset for stress analysis in social media, Given datasets consists of: -

1. dreaddit-train.csv

**Libraries Used**

We have used multiple libraries to perform EDA :-





To load the data for performing data cleaning ,we first need to access the dataset. The code reads the data from a CSV file named "dreaddit-train.csv" into a pandas DataFrame named "df".



**Data Cleaning and Preprocessing**

Data cleaning and preprocessing involve several steps to prepare the dataset for analysis and model training to ensure that data is clean and accurate.

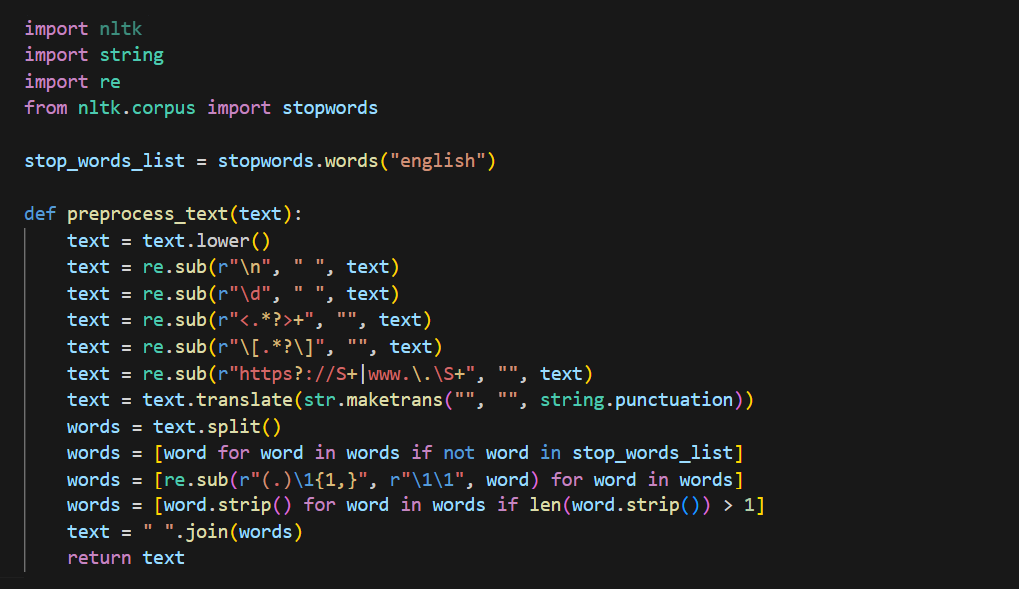
**1 Finding NA and Null values :-**

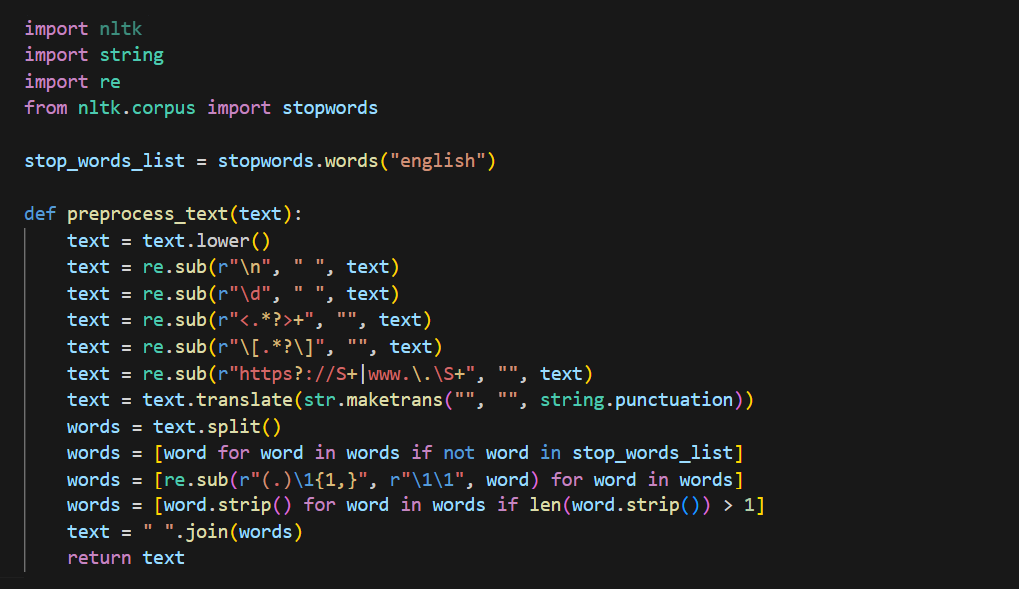
We found no Null or NA values across the data set.



Based on the provided index of columns, we have a dataset with a wide range of features. Focusing on text and label columns for data cleaning and preprocessing, we select only those columns and perform the necessary steps.

Data preprocessing is a crucial step in preparing raw data for analysis and machine learning models. It involves cleaning, transforming, and organizing the data to make it suitable for further processing.

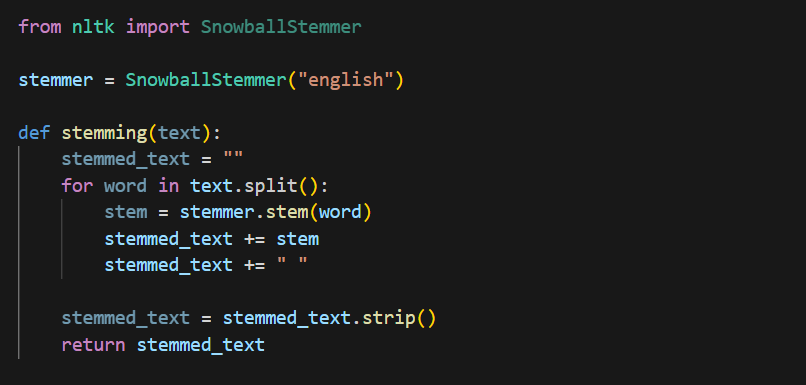




* **Importing Libraries**: The code imports necessary libraries such as **nltk**, **string**, and **re** (regular expressions) to perform text preprocessing tasks.
* **Defining Stopwords**: The code imports stopwords from the NLTK library. Stopwords are common words that are often removed from text data as they do not carry significant meaning.
* **preprocess\_text Function**: This function takes a string of text as input and performs the following preprocessing steps:
  + **Converting to Lowercase**: It converts the entire text to lowercase to ensure consistency in text representation.
  + **Removing Newlines**: It removes newline characters from the text.
  + **Removing Digits**: It removes digits (numbers) from the text.
  + **Removing HTML Tags**: It removes HTML tags (enclosed within **< >**).
  + **Removing URLs**: It removes URLs (starting with **http://**, **https://**, or **www.**) from the text.
  + **Removing Punctuation**: It removes punctuation marks from the text using the **string.punctuation** set.
  + **Tokenization**: It splits the text into individual words (tokens).
  + **Removing Stopwords**: It removes stopwords from the text. Stopwords are common words like "the", "is", "and", etc., which often do not carry significant meaning in the context of text analysis.
  + **Removing Consecutive Characters**: It reduces consecutive characters to two occurrences to handle elongated words (e.g., "loooong" becomes "loong").
  + **Stripping and Joining**: It strips any leading or trailing whitespace from each word and joins the words back into a single string.
  + **Returning Preprocessed Text**: Finally, the function returns the preprocessed text.



After executing this line of code,the “Text” column in dataframe df will contain preprocessed text data,ready for further analysis and modelling.



The provided code defines a function **stemming** that performs stemming on text data using the SnowballStemmer from the NLTK library. Let's break down the function and understand what each part does:

* **Importing Libraries**: The code imports the SnowballStemmer from the NLTK library.
* **Initializing Stemmer**: It initializes a SnowballStemmer object with the language parameter set to "english".
* **stemming Function**: This function takes a string of text as input and performs the following steps:
  + **Tokenization and Stemming**: It tokenizes the text into individual words and then stems each word using the SnowballStemmer. Stemming reduces words to their root or base form.
  + **Concatenation**: It concatenates the stemmed words back into a single string, separated by spaces.
  + **Stripping**: It removes any leading or trailing whitespace from the resulting string.
  + **Returning Stemmed Text**: Finally, the function returns the stemmed text.

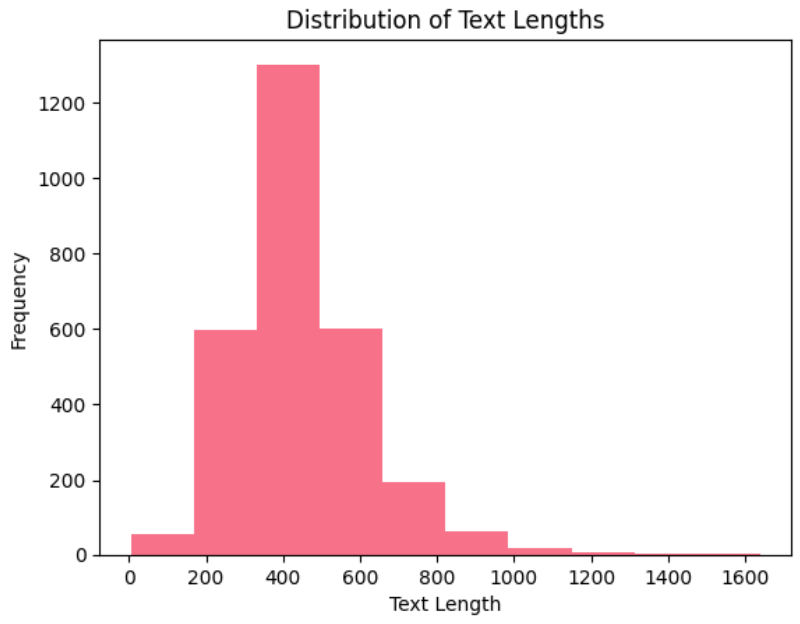
To apply this stemming function to the 'text' column of a DataFrame **df**, you can use the **.apply()** method as follows:



This will stem each element in the 'text' column of the DataFrame **df** and update the column with the stemmed text data.

**Exploratory Data Anaylsis (EDA)**

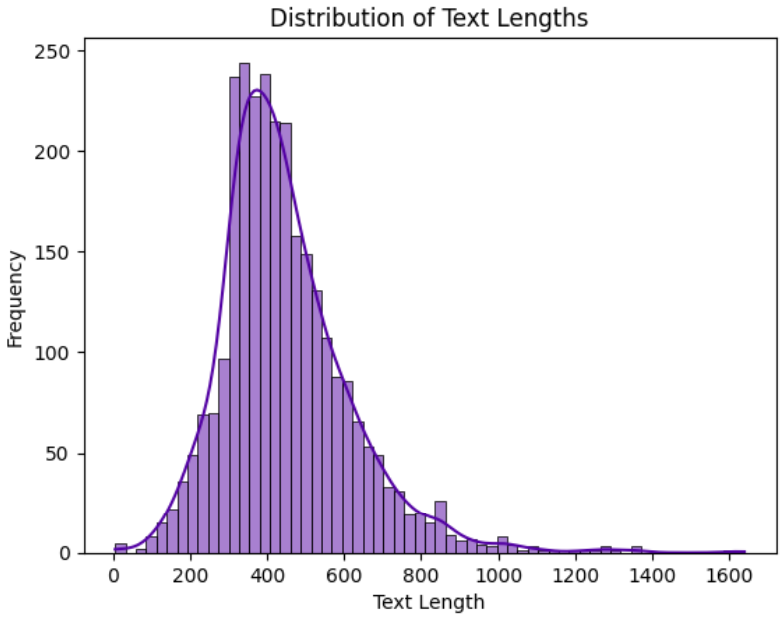
1. Visualize the distribution of text length in dataset.



* The histogram shows the distribution of text lengths.
* The graph shows that the textlength is mostly short and a spike at 400.
* The textlength is measure of how long the texts are in a data.

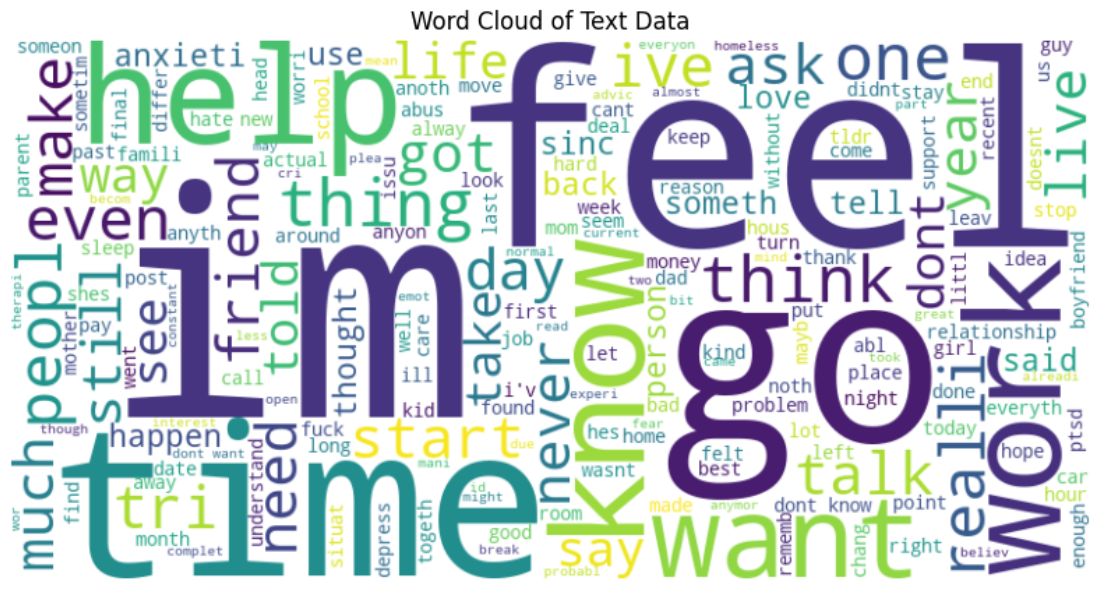
2.KDE curve.

* The KDE curve (if enabled) provides an estimation of the probability density function of the underlying distribution of text lengths.
* It helps to visualize the smooth distribution of text lengths and provides additional insights into the overall shape of the distribution.



3. **Word Cloud of Text Data**:

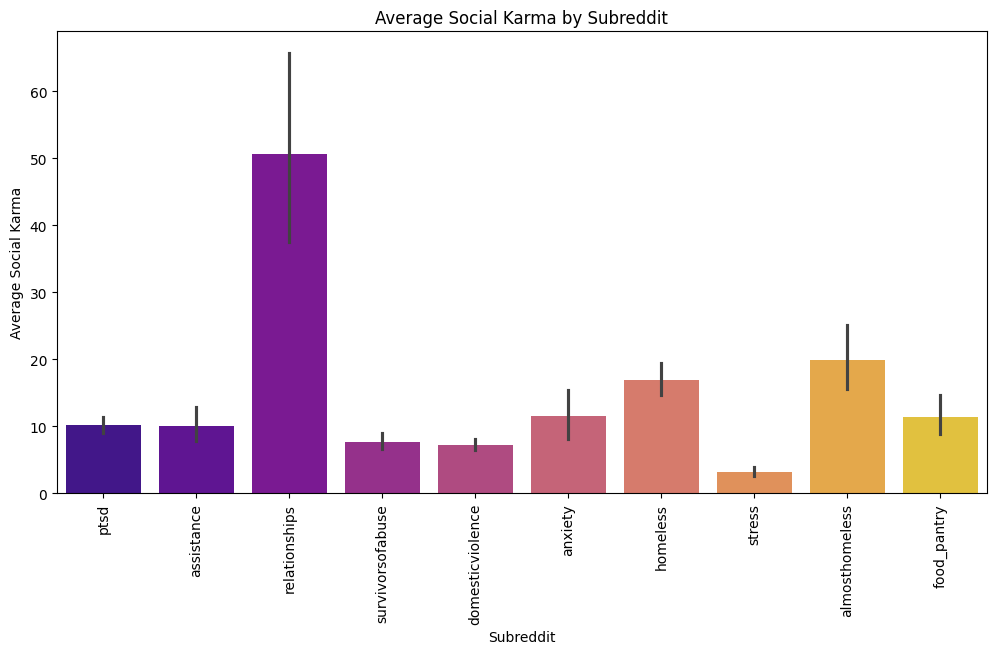
* Create a word cloud to visualize the most common words in the text data.



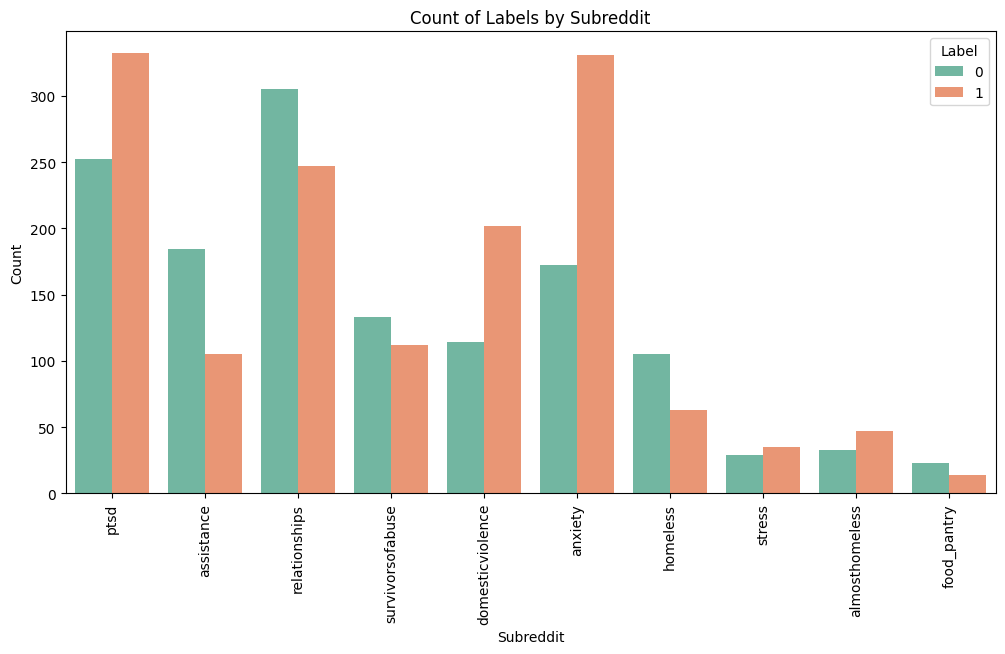
A word cloud is a visual representation of text data where the importance of each word is depicted by its size.

4. **Countplot of Subreddits**:

* Visualize the count of posts for each subreddit

**5.Grouped Bar Plot**:

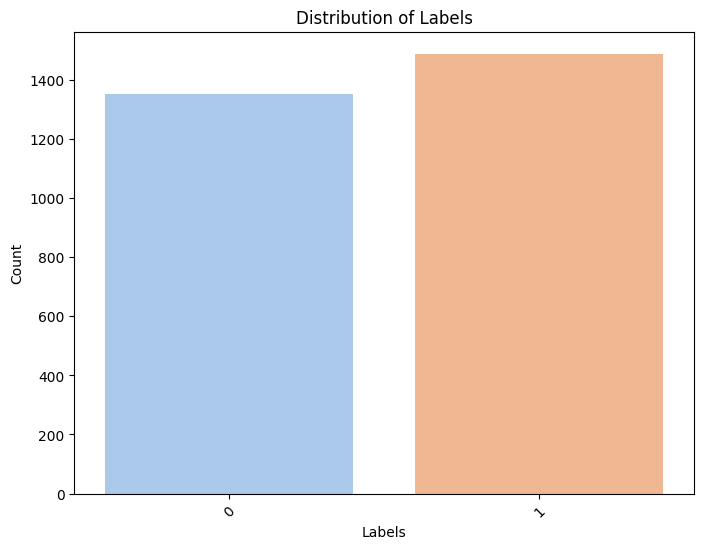
* Show the count of each label within each subreddit using a grouped bar plot.



This shows that maximum number of posts are of subreddits of ptsd, relationship,anxiety.Whereas maximum number of positive(stress) labels are on ptsd and anxiety post, and most of No stress labels are on relationship subreddits.

**Distribution of Label Counts**:

* Visualize the distribution of labels in your dataset.



**Feature Engineering**



This piece of code utilizes the **TfidfVectorizer** from scikit-learn to convert the text data (**X**) into a numerical format suitable for machine learning models. Here's an explanation of each part of the code:

* **Importing TfidfVectorizer**:
  + **from sklearn.feature\_extraction.text import TfidfVectorizer**: This line imports the **TfidfVectorizer** class from the scikit-learn library. **TfidfVectorizer** is used to convert a collection of raw documents (in this case, the text data) into a matrix of TF-IDF features.
* **Initializing TfidfVectorizer**:
  + **tfidf = TfidfVectorizer()**: This line initializes a **TfidfVectorizer** object named **tfidf**. The **TfidfVectorizer** converts a collection of text documents into a matrix of TF-IDF features.
* **Fitting and Transforming Text Data**:
  + **X\_scaled = tfidf.fit\_transform(X)**: This line fits the **TfidfVectorizer** to the text data **X** and transforms it into a matrix of TF-IDF features. The **fit\_transform** method first learns the vocabulary from the text data and then transforms the text data into a sparse matrix representation where each row corresponds to a document (or text sample) and each column corresponds to a unique word in the vocabulary. The values in the matrix represent the TF-IDF score of each word in each document.

**Model Building and Evaluation**

Model building and evaluation process is iterative ,involving experiments,validation and refinement to develop effective machine learning solutions.

“Label” column is readily encoded as 1 for Stress and 0 for No Stress.

Train\_test\_split:

**train\_test\_split(X\_scaled, y, test\_size=0.3)**: This function is used to split the dataset into random train and test subsets. It takes several parameters:

* **X\_scaled**: The feature matrix, which contains the TF-IDF transformed text data.
* **y**: The target variable, which contains the labels.
* **test\_size**: The proportion of the dataset to include in the test split. Here, it's set to **0.3**, meaning that 30% of the data will be used for testing, and the remaining 70% will be used for training.



**Output Variables**:

* **X\_train**: The training data for features (TF-IDF transformed text data).
* **X\_test**: The testing data for features (TF-IDF transformed text data).
* **y\_train**: The training data for the target variable (labels).
* **y\_test**: The testing data for the target variable (labels).

**Logistic Regression model**

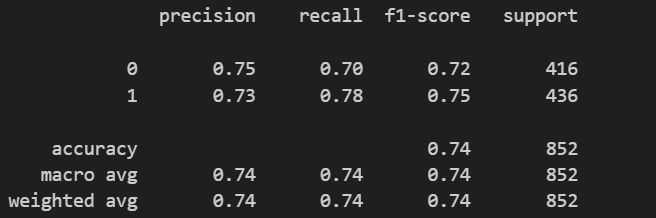
We will check the performance of the logistic regression model on both training and testing data ,helping to access its effectiveness in classifying text as stress or no stress

We use the LogisticRegression() function of the sklearn library to fit the data.

Accuracy on training data: 0.8927

Accuracy on testing data:0.7387

Classification Report



* **Training data accuracy:** The logistic model achieved an accuracy of approximately 89% on the training data.
* **Testing data accuracy:** The logistic model achieved an accuracy of approximately 74% on the testing data.
* **Classification Report:**
* **Precision**: It measures the accuracy of positive predictions. For label 0, 75% of the samples predicted as label 0 were actually label 0, and for label 1, 73% of the samples predicted as label 1 were actually label 1.
* **Recall**: It measures the ability of the label to find all positive instances. For label 0, 70% of the actual label 0 samples were correctly predicted, and for label 1, 78% of the actual label 1 samples were correctly predicted.
* **F1-score**: It is the harmonic mean of precision and recall. It provides a balance between precision and recall. For label 0, the F1-score is 0.72, and for label 1, the F1-score is 0.75.
* **Overall Evaluation:** The observed disparity between training and testing accuracies suggests overfitting, and addressing this issue may involve adjusting model complexity, performing feature selection, tuning hyperparameters, or assessing data quality. Additionally, considering alternative evaluation metrics can provide a more nuanced assessment of the model's performance.

**Best hyperparameters**: {'C': 1, 'penalty': 'l2'}

**Best cross-validation score**: 0.7603293545814716

Testing data accuracy with best hyperparameter ,the logistic regression model achieved an accuracy of 76% on the testing data.

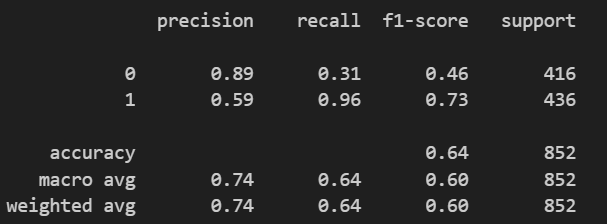
**Naive Bayes Classifier:**

To implement a naive bayes classifier using sci-kit learn,we use the MultinomialNB class for text classification tasks.

Accuracy on training data (Naive Bayes): 0.8066

Accuracy on testing data (Naive Bayes): 0.6431

Classification Report



**Naive Bayes Model Performance**:

* **Training Data Accuracy**: The Naive Bayes model achieved an accuracy of approximately 80% on the training data.
* **Testing Data Accuracy**: On the testing data, the Naive Bayes model achieved an accuracy of approximately 64%.
* **Classification Report:**
* **Precision**: For label0, 89% of the samples predicted as label 0 were actually label 0, while for label 1, 59% of the samples predicted as label 1 were actually label 1.
* **Recall**: For label 0, 31% of the actual label 0 samples were correctly predicted, while for label 1, 96% of the actual label 1 samples were correctly predicted.
* **F1-score**:For label 0, the F1-score is 0.46, and for label 1, it is 0.73. These scores represent the harmonic mean of precision and recall, providing a balanced measure of the model's performance.
* **Overall Evaluation:** The Naive Bayes model achieved decent accuracy on the training data, its performance on the testing data suggests potential issues with overfitting and highlights the need for further evaluation and model refinement to improve generalization performance.

**Best hyperparameters**: {'alpha': 0.1, 'fit\_prior': False}

**Best cross-validation score**: 0.7326354695391314

Testing data accuracy with best hyperparameter ,the Naive Bayes Classifier model achieved an accuracy of 73% on the testing data.

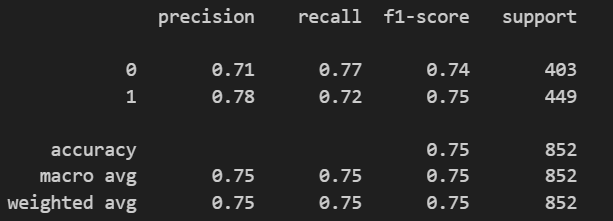
**Support Vector Classifier(SVC):**

To implement SVM (Support Vector Machine) classifier using the **SVC** class from scikit-learn, with a linear kernel specified.

Accuracy on training data (SVC): 0.9446

Accuracy on testing data (SVC): 0.7453

Classification Report:



**Support Vector Classifier Performance:**

* **Training Data Accuracy**: The Naive Bayes model achieved an accuracy of approximately 94% on the training data.
* **Testing Data Accuracy**: On the testing data, the Naive Bayes model achieved an accuracy of approximately 75%.
* **Classification Report:** The higher accuracy on the training data compared to the testing data suggests that the model may be overfitting to the training data. While the accuracy on the testing data is lower than on the training data, an accuracy of 75% still suggests that the model performs reasonably well on unseen data.

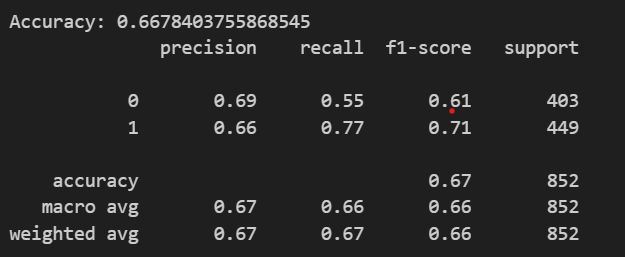
**KNN Classifier:**

We will check performance of KNN classifier model on both the data,and collect necessary insights.

Accuracy on training data (KNN): 0.78

Accuracy on testing data (KNN): 0.67

Classification Report:



**KNN Classifier Performance**

* **Training Data Accuracy**: The KNN model achieved an accuracy of approximately 78% on the training data.
* **Testing Data Accuracy**: On the testing data, the KNN model achieved an accuracy of approximately 67%.
* **Classification Report:** **Precision**:For class 0, the precision is 0.69, indicating that out of all instances predicted as class 0, 69% were correctly classifiedFor class 1, the precision is 0.66, indicating that out of all instances predicted as class 1, 66% were correctly classified.
* **Recall**:For class 0, the recall is 0.55, indicating that the model correctly identified 55% of all actual class 0 instances.For class 1, the recall is 0.77, indicating that the model correctly identified 77% of all actual class 1 instances.
* **F1-score**:For class 0, the F1-score is 0.61, which is the harmonic mean of precision and recall for class 0.For class 1, the F1-score is 0.71, which is the harmonic mean of precision and recall for class 1.
* **Overall Evaluation**:The model's performance on the testing data is still reasonable, but there might be room for improvement. Consider exploring other evaluation metrics, conducting further feature engineering, or experimenting with different values for the hyperparameters to improve generalization.

**Best hyperparameters**: {'n\_neighbors': 11}

**Accuracy**: 0.68

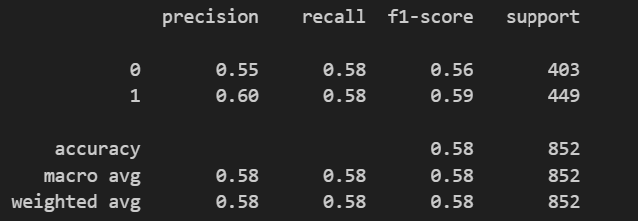
**Decision Tree Classifier**

We will check the performance of the decision tree classifier model on both training and testing data ,helping to access its effectiveness in classifying text as stress or no stress.

Accuracy on training data : 0.9994

Accuracy on testing data : 0. 5786

Classification Report:



**Decision Tree Classifier Performance**

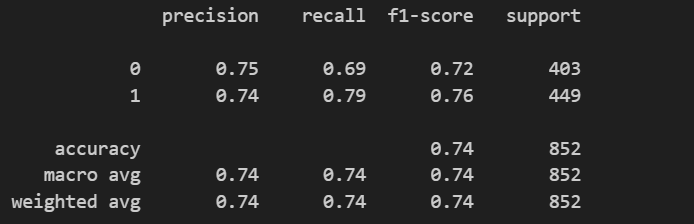
* **Training Data Accuracy**: The Decision tree model achieved an accuracy of approximately 58% on the training data.
* **Testing Data Accuracy**: On the testing data, the Decision tree model achieved an accuracy of approximately 100%.
* **Classification Report:**
* **Precision**:For class 0, the precision is 0.55, indicating that out of all instances predicted as class 0, 55% were correctly classified.For class 1, the precision is 0.60, indicating that out of all instances predicted as class 1, 60% were correctly classified.
* **Recall**:For class 0, the recall is 0.58, indicating that the model correctly identified 58% of all actual class 0 instances.For class 1, the recall is 0.58, indicating that the model correctly identified 58% of all actual class 1 instances.
* **F1-score**:For class 0, the F1-score is 0.56, which is the harmonic mean of precision and recall for class 0.For class 1, the F1-score is 0.59, which is the harmonic mean of precision and recall for class 1.
* **Overall Evaluation:** While achieving 100% accuracy on the testing data may seem ideal, it raises concerns about the reliability of the model. Such high accuracy, especially on unseen data, is often indicative of overfitting. It's essential to carefully examine the model's performance and consider techniques such as cross-validation, regularization, or pruning to prevent overfitting and improve generalization to unseen data.

Testing data accuracy after hyperparameter tuning:0.6232

**Random Forest Classifier:**

We will check the performance of the decision tree classifier model on both training and testing data ,helping to access its effectiveness in classifying text as stress or no stress.

Classification Report:



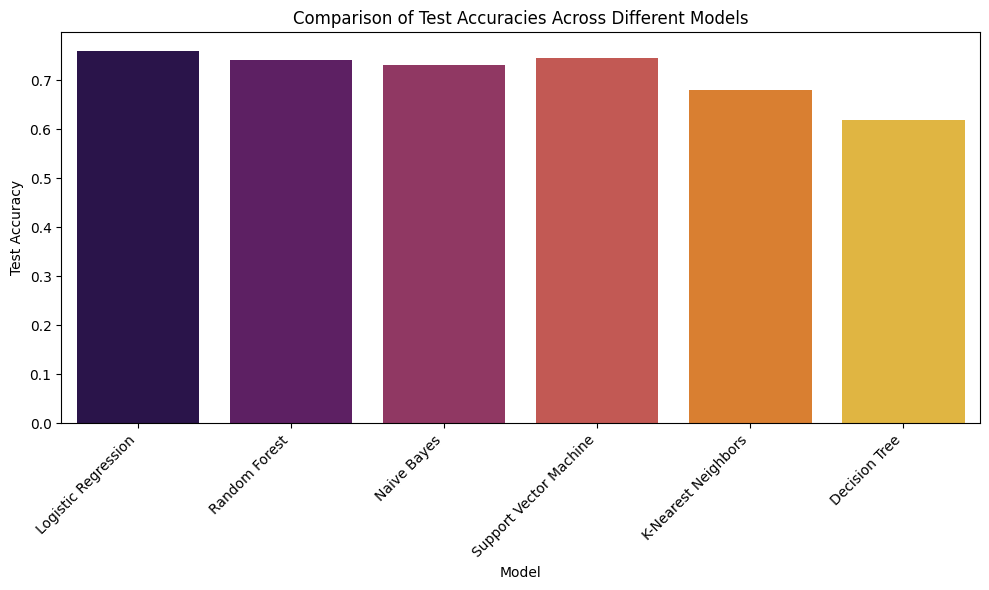
Accuracy on training data : 0.9994

Accuracy on testing data : 0. 7417

**Random Forest Performance**

* **Training Data Accuracy**: The Decision tree model achieved an accuracy of approximately 74% on the training data.
* **Testing Data Accuracy**: On the testing data, the Decision tree model achieved an accuracy of approximately 100%.
* **Classification Report: Precision**:For class 0, the precision is 0.75, indicating that out of all instances predicted as class 0, 75% were correctly classified.For class 1, the precision is 0.74, indicating that out of all instances predicted as class 1, 74% were correctly classified.
* **Recall**:For class 0, the recall is 0.69, indicating that the model correctly identified 69% of all actual class 0 instances.For class 1, the recall is 0.79, indicating that the model correctly identified 79% of all actual class 1 instances.
* **F1-score**:For class 0, the F1-score is 0.72, which is the harmonic mean of precision and recall for class 0.For class 1, the F1-score is 0.76, which is the harmonic mean of precision and recall for class 1.
* **Overall Evaluation**: The Random Forest classifier demonstrates reasonable performance in terms of accuracy, precision, recall, and F1-score. However, further optimization or exploration of other models may be necessary to improve performance further, depending on the specific requirements and constraints of the problem.

**Comparision and Conclusion:**



**1.Logistic Regression:**

Accuracy on training data: ~89%

Accuracy on testing data: ~76%

Balanced performance across classes with reasonable precision, recall, and F1-score for each class.Overall, logistic regression demonstrates good generalization performance.

**2.Random Forest:**

Accuracy on training data: 100%

Accuracy on testing data: ~74%

Random forest exhibits overfitting, with perfect accuracy on training data but lower accuracy on testing data.

**3.Naive Bayes:**

Accuracy on training data: ~80%

Accuracy on testing data: ~73%

Good performance with balanced precision and recall across classes, but slightly lower than logistic regression.

**4.SVC:**

Accuracy on training data: ~94%

Accuracy on testing data: ~74%

Good performance with balanced precision and recall across classes, but slightly lower than logistic regression.

**5.K-Nearest Neighbours (KNN):**

Accuracy on training data: ~78%

Accuracy on testing data: ~68%

Lower performance compared to logistic regression,accuracy on test data is still reasonable.

**6.Decision Tree:**

Accuracy on training data: ~100%

Accuracy on testing data: ~62%

Significant overfitting evident, with perfect accuracy on training data but lower accuracy on testing data.

Conclusion:

Logistic Regression,SVC and Naive Bayes demonstrate the best generalization performance with good accuracy and balanced precision/recall across classes.

KNN, Decision Tree, and Random Forest models exhibit overfitting, leading to lower accuracy and imbalanced performance across classes, especially on the testing data.

Logistic regression, offers a good balance between simplicity and performance, making it a suitable choice for this classification task.