**Twitter Sentiment Analysis**

**Problem Statement: Understanding the sentiment behind the tweets**

**Dataset Overview:**

The dataset consists of 74682 entries and 3 features.

**ID:** This column represents a unique identifier assigned to each tweet. It helps in distinguishing one tweet from another within the dataset.

**Topic:** Theme of the tweet.

**Sentiment:** The sentiment or emotional polarity expressed in the tweet.

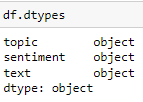
**Text:** It includes the message or statement posted by the Twitter user.

**Imported Necessary Libraries**



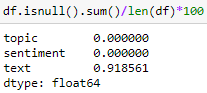
**Exploratory Data Analysis**

**Check for the data types**



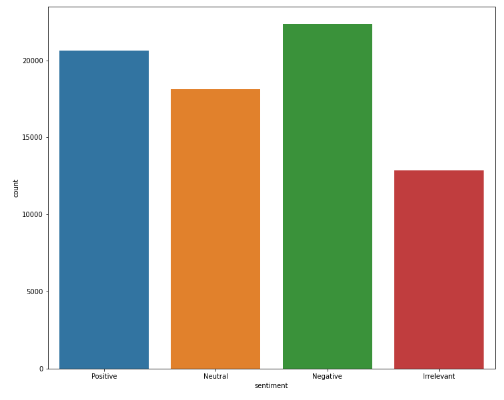
The data types are appropriate for the given features.

**Check for the Null values**



There is 91.8% of the null values present in the text column.

**Distribution of Sentiments in Twitter Data**



**Text Preprocessing**

**-- Drop the null values in the target column**

**Lowercase Conversion**:

text.lower() converts the processed text into lowercase. This normalization step ensures that words are treated consistently regardless of their casing.

**Tokenization**:

text.split() splits the processed text into tokens (words). This step breaks down the text into individual units (tokens) that can be further analyzed or processed.

**Regex Substitution**:

re.sub(r'[^a-zA-Z\s]', '', text, flags=re.I|re.A) removes non-alphabetic characters ([^a-zA-Z\s]) from the text. This includes any characters that are not letters (a-z and A-Z) or whitespace (\s). The re.I flag makes the regex case-insensitive, and re.A ensures ASCII-only matching.

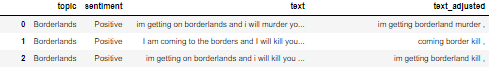
**Stop Words:**

This is a set of stopwords from the NLTK library for the English language. Stopwords are common words (like 'and', 'the', 'is', etc.) that are often filtered out during text preprocessing to focus on more meaningful words.

**Stemming:**

Using Port Stemmer algorithm from NLTK for stemming words, Stemming reduces words to their root or base form. which helps in normalization

-- The preprocessed text stored in a new column called text\_adjusted



**Word to vector:**

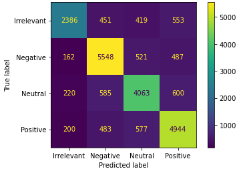
It is a type of word embedding model that transforms words into fixed-length vectors of real numbers.

**Base Model Evaluation**

**Initial Logistic Regression Model**

The base model employed for initial evaluation was a logistic regression model without any preprocessing steps such as data scaling or outlier removal. This approach aimed to establish a benchmark performance to assess subsequent improvements. The model yielded f1\_score for class 0 is 70%, class 1 is 80%, for class 2 is 74%, for class 3 is 77%, suggesting imbalance in the data.

Confusion Matrix:

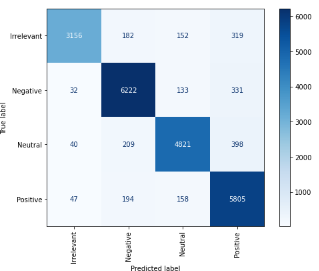


**Ensemble Models:**

**Random Forest Classifier:**

The model employed for evaluation was a random forest model with preprocessing steps such as data scaling. This approach aimed to establish a benchmark performance to assess subsequent improvements. The model yielded a f1\_score for class 0 is 89%, class 1 is 92%, for class 2 is 90%, for class 3 is 89%, suggesting slight imbalance in the data.

Confusion Matrix



**ADA Boost:**

The model employed for evaluation was an ADA boost model with preprocessing steps such as data scaling. This approach aimed to establish a benchmark performance to assess subsequent improvements. The model yielded 31% of f1\_score for class 0 is 18%, class 1 is 52%, for class 2 is 39%, for class 3 is 49%, suggesting imbalance in the data.

Confusion Matrix



**Final Evaluation**

After thorough evaluation, the Random Forest model scaled input data was chosen as the final model. The decision was based on its ability to maintain strong predictive performance while mitigating imbalance issues observed in earlier stages. The model’s performance metrics and stability make it suitable for set attributes.

