**Introduction**

In today's digital age, social media platforms have become a reflection of people’s mental states, offering a window into their thoughts and emotions. Recognizing patterns of mental distress and suicidal thoughts from social media posts is critical in providing timely support to individuals at risk. The **Mental Wellness** project is focused on detecting suicidal thoughts among social media users and identifying individuals who may be struggling mentally. By leveraging text classification models, this project aims to alert medical professionals to intervene and provide necessary support before any adverse events occur, potentially saving lives.

**1. Null and Duplicate Check**

Before any preprocessing, it’s important to ensure that the dataset is clean. Here, we check for:

* **Null values**: Missing values in text, class, need to be handled (
* **Duplicate entries**: Repeated text instances should be identified and removed, as they can introduce bias in training the model.

**Example:**

If a text value is null or the entire row is duplicated, it should be dropped to avoid inconsistencies in the model.

**There were no Null or Duplicate value in our model.**

**2. Balance of Distribution Check**

For a binary classification problem like suicide vs. non-suicide, **class balance** is critical. Imbalanced classes (e.g., significantly more non-suicide than suicide texts) can cause the model to be biased toward the majority class.

**Example:**

If number of non-suicide entries is higher than suicide, requiring balancing techniques such as **undersampling** the majority class or **oversampling** the minority class (e.g., using SMOTE).

**There were no class imbalance in our Model initially.**

**3. Word Count Column**

We generated the word\_count column to add a numeric feature, representing the number of words in each text. This feature can help understand the length distribution across different classes.

**4. Outlier Removal**

Outliers are texts with unusually high or low word counts compared to the typical text lengths in your dataset. Removing these helps avoid skewed model performance due to extreme values.

**Example:**

Texts with extremely high word counts (e.g., outliers with 1000+ words) or very low word counts (e.g., texts with 1 or 2 words) are removed to maintain consistency.

**We removed entries greater then Percentile 80 and Less then Percentile 20**

**5. Decreasing Difference in Variance of Word Count Between Two Classes**

The **variance of word count** between the suicide and non-suicide classes reflects how varied the lengths of the texts are in each class. A **decreasing difference in variance** indicates that the word count distributions across the two classes are becoming similar.

**Implication in Machine Learning:**

If the word count variance becomes too similar across classes, this feature alone may not help differentiate between the suicide and non-suicide texts effectively. In such cases, additional features (e.g., text sentiment, specific words, or phrase patterns) become more important.

**Before | After**

**Suicide : 36.70 | 20.54**

**Non-Suicide: 46.61 | 21.81**

**6. Text Preprocessing**

Preprocessing the text is crucial for preparing it for modeling. Key steps include:

* **Lowercasing**: Convert all text to lowercase to avoid case sensitivity.
* **Removing Punctuation**: Punctuation is usually irrelevant for classification.
* **Removing Stop Words**: Common words like "the", "is", "in" are removed since they do not provide much meaning.
* **Tokenization**: Text is split into individual words.
* **Stemming or Lemmatization**: Words are reduced to their base form (e.g., "running" → "run").
* **Rejoin Tokens**: After tokenization, the cleaned tokens can be rejoined for specific vectorization techniques.

**Example:**

Original: “Running fast is better.” After preprocessing:

* Lowercase: “running fast is better.”
* Remove punctuation: “running fast is better”
* Tokenize: [‘running’, ‘fast’, ‘is’, ‘better’]
* Lemmatization: [‘run’, ‘fast’, ‘is’, ‘good’]

**7. Label Preprocessing (Binary Category)**

For binary classification, categorical labels like suicide and non-suicide are converted to numerical form (e.g., 0 and 1). This can be done using **LabelEncoder** or manual mapping.

**Example:**

* suicide → 1
* non-suicide → 0

**8. Vectorization**

Text data must be converted into a numerical form that the model can understand.

Common vectorization methods include:

* **Bag of Words (BoW)**: Each word is represented by its frequency in the text.
* **N-grams**: Sequences of N consecutive words are considered features (e.g., unigrams, bigrams, and trigrams).

**Example:**

For Our model:

* **Unigrams**: Single words as features.
* **Bigrams**: Pairs of consecutive words as features (20000 features).
* **Trigrams**: Triplets of consecutive words as features (10000 features).

**9. Model Building: Gaussian Naive Bayes**

We are using **Gaussian Naive Bayes** as our classification model. Naive Bayes works well for text classification because it assumes that features (words or N-grams) are independent, which often holds true in text classification tasks.

**Features:**

1. **Unigrams**: Basic word frequency.
2. **Bigrams with 20000 features**: Word pairs, which may capture some context.
3. **Trigrams with 10000 features**: Word triplets, which provide deeper contextual meaning.

**10. Advantages of the Model**

* **Simplicity**: Naive Bayes is simple and computationally efficient.
* **Works well for text data**: Due to the assumption of feature independence, it can handle text data effectively.
* **Fast and Scalable**: It is quick to train even on large datasets.
* **Handles high-dimensionality well**: The method can manage large feature spaces (like the 20000 bigram features) without overfitting easily.

**11. Disadvantages of the Model**

* **Dropping Rows**: In real-life scenarios, dropping too many rows due to issues like extreme word counts or missing data is impractical. Reducing the size of the dataset may limit the richness and variability of the data available to the model.
* **Assumption of Independence**: Naive Bayes assumes that all features (words) are independent, which might not hold true in complex text structures where word dependencies can be important.
* **Simple Feature Representation**: Using just word count and basic N-grams may limit the model’s ability to capture deeper semantic meaning. More advanced feature engineering (e.g., sentiment analysis or word embeddings) might be necessary.

**Conclusion**

The Gaussian Naive Bayes model implemented in this project has demonstrated strong performance, particularly with the use of unigrams, bigrams, and trigrams, achieving up to 85% accuracy in identifying potential suicidal ideation. Based on the careful preprocessing and analysis of text data, including word count and N-gram features, the model effectively differentiates between suicidal and non-suicidal posts.

In conclusion, the **Mental Wellness** project can significantly contribute to mental health awareness by enabling early detection of individuals at risk, facilitating timely medical intervention, and helping prevent tragic outcomes.