**Report: Sentiment Analysis for Mental Health Status Classification**

**1. Objective**

The objective of this project is to develop a text classification pipeline that categorizes statements related to mental health into predefined sentiment-based categories. The workflow includes data preprocessing, text normalization, vectorization using TF-IDF, and model training using machine learning techniques.

**2. Data Overview**

* **Dataset**: Sentiment\_analysis\_dataset.csv
* **Content**: The dataset consists of text entries (mental health-related statements) and their associated sentiment labels.

Imported and dropped null values using pandas.

**3. Exploratory Data Analysis**

**Class Distribution**

* A bar plot was generated to visualize the distribution of sentiment labels.
* Observations from the visualization indicated the presence of class imbalance, with some classes appearing significantly more frequently than others.
* This could be used later on to decide whether to under sample or oversample

**4. Text Preprocessing**

Text data underwent the following preprocessing steps to ensure consistency and suitability for machine learning:

* **Lowercasing**: Converted all text to lowercase to reduce dimensionality.
* **Noise Removal**: Eliminated punctuation, numbers, and special characters using regular expressions.
* **Stopword Removal**: Used nltk.corpus.stopwords to remove common English stopwords.
* **Lemmatization**: Applied WordNetLemmatizer to reduce words to their base or dictionary form.

These steps were critical in standardizing the text and reducing irrelevant variation in the data.

**5. Feature Extraction**

TF-IDF vectorization was applied to transform the cleaned text into numerical features:

* **Tool**: TfidfVectorizer from sklearn.feature\_extraction.text.
* **Configuration**:
  + max\_features=4000 (too much caused too much memory usage or slow training)
  + ngram\_range=(1,2)
  + stop\_words='english'

The vectorizer captured both unigrams and bigrams, providing richer representations of the input text.

**LIWC Features: Specific for our use case:**

We decided to use LIWC to extract words necessary for each target variable type, with predefined words we thought fit best for each category. What helped with is that it reduced false positives in the classes with overlapping symptoms like discussed before, we saw a drop of false positives in depression for example from 600 cases to 380 cases.

A computer screen shot of text

AI-generated content may be incorrect.

**6. Model Development**

**Data Split**

* The dataset was split into training and testing sets using an 80/20 ratio via train\_test\_split.

**Model Selection**

* **Model Used**: XGBoost and random forest Regression
* **Justification**: Logistic regression is a strong baseline model for text classification, offering interpretability and efficient training.

**Training and Evaluation**

* Metrics used for evaluation included:
  + **Accuracy**
  + **Precision**

**Recall**

* + **Confusion Matrix**

These metrics were computed using classification\_report and confusion\_matrix from sklearn.metrics.

**7. Results – First run – Random Forest**

* **Accuracy**: Random forest: 74%
* **Class-wise Performance**: Precision and recall scores indicated that the model performed well on the majority class but had reduced performance on underrepresented classes, highlighting the class imbalance issue.
* **Confusion Matrix**: classes with overlapping definition like suicidal and depression had the highest level of misclassification. Anxiety and stress had a similar situation, possibly due them having similar symptoms

**7**. **Results – First run – XGBoost**

* **Accuracy**: 76%
* A graph with numbers and a number of different colored squares

  AI-generated content may be incorrect.

**7. Results – Grid search – XGBoost**

We picked XGBoost as our final model to it, in almost all cases, performing better than RF.  
We ran a grid search to find the best possible parameters over this sample space:  
*'n\_estimators': [50, 100, 150],*

*'max\_depth': [3, 5, 7],*

*'learning\_rate': [0.01, 0.1, 0.2],*

*'subsample': [0.8, 1.0]*

**8. Observations and Recommendations**

**Strengths**

* The preprocessing pipeline effectively normalized diverse text inputs.
* TF-IDF vectorization with bigrams provided meaningful feature representations.
* Logistic regression served as a strong initial classifier.

**Limitations**

* Class imbalance negatively affected performance on minority classes.
* The model may not capture contextual nuances due to the limitations of TF-IDF.
* Slow training: TF-IDF creates a sparse matrix which slowed down training
* Lack of contextual awareness: the model only detects patterns in the English language thus it is restricted to said language and may even get confused if a different way of writing English is used.