Part 1: Data Preparation/Feature Engineering

1. Overview

This phase was one of the most critical parts of the project. The primary goal was to process, clean, and transform the raw, unstructured text posts from the Combined Data.csv file into a high-quality, numerical format. This process ensures that our machine learning model, particularly the advanced DistilBERT model, could effectively learn the patterns associated with different mental health statuses. This involved cleaning the text, balancing a highly imbalanced dataset, and using an advanced "WordPiece" tokenization strategy.

2. Data Collection

- **Source:** The dataset used was a single CSV file named Combined Data.csv from Kaggle. This file contained over 52,000 text posts (or 'statements') from users, with each post labeled with a corresponding 'status' (e.g., 'Anxiety', 'Depression', 'Normal', etc.).
- **Preprocessing:** Upon loading the data into a pandas DataFrame, an initial "Unnamed: 0" index column was dropped as it was redundant.

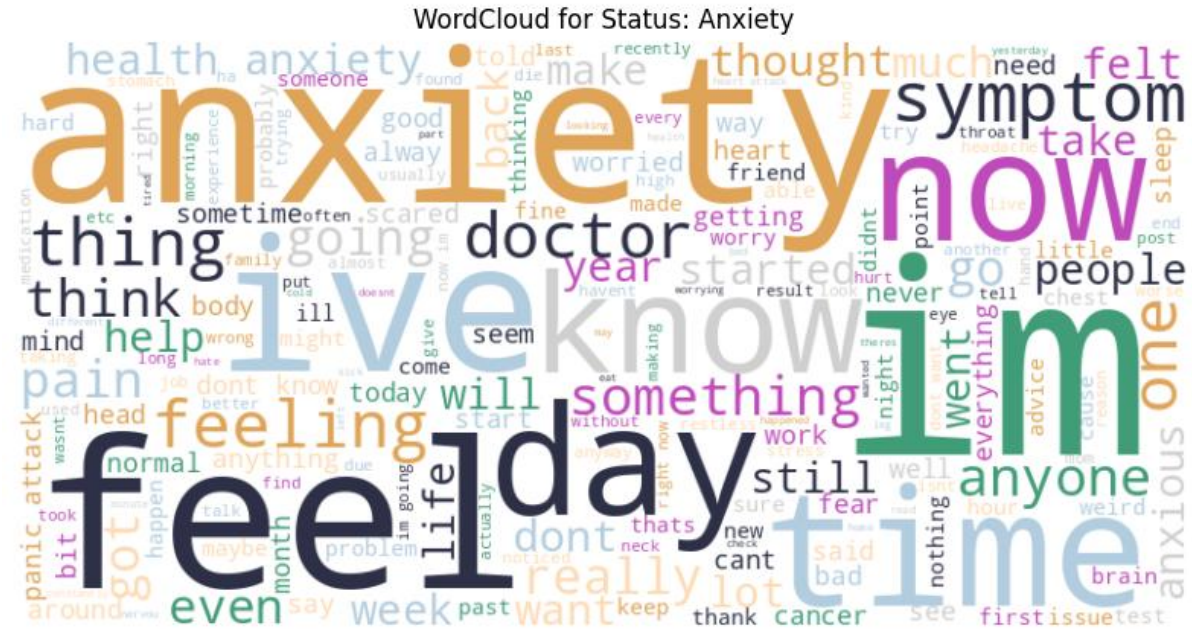

Figure 1.1 WordCloud for Status: Anxiety

- **Sampling:** To ensure rapid model prototyping and stay within computational limits (especially when fine-tuning a large Transformer model), the full dataset of 52,681 rows was **randomly sampled down to 6,000 rows**. This smaller, representative sample was used for all subsequent cleaning, training, and evaluation steps.

3. Data Cleaning

A clean_statement function was created to systematically clean every text post. The following steps were applied to the statement column:

1. **Drop Missing Values:** Any rows with missing statements (data.dropna()) were removed to prevent errors.
2. **Lowercase:** All text was converted to lowercase to ensure uniformity (e.g., 'Happy' and 'happy' are treated as the same word).
3. **Remove Punctuation and Numbers:** A regular expression (re.sub(r"[^a-zA-Z\s]", "", text)) was used to remove all characters that were not letters or spaces. This eliminates noise from punctuation, numbers, and special symbols.
4. **Remove Stopwords:** Common English "stopwords" (e.g., 'is', 'a', 'the', 'in') were removed using the NLTK library. These words are highly frequent but carry little predictive meaning.
5. **Remove Extra Spaces:** All extra whitespace and line breaks were stripped to create a clean, single string.

4. Exploratory Data Analysis (EDA)

After loading and sampling the data, an exploration analysis was performed. The most critical insight gained was that the dataset was **highly imbalanced**.

- **Key Insight:** The status categories were not equally represented. For example, in the 6,000-row sample, there were 1,894 posts for 'Normal' but only 131 for 'Personality disorder'.
- **Implication:** If we trained a model on this imbalanced data, it would become highly biased towards predicting the 'Normal' class and would perform very poorly on rare but critical classes like 'Suicidal' or 'Personality disorder'.
- **Visualization:** This imbalance was confirmed by creating a **pie chart** showing the percentage of posts for each status. This visualization was the primary justification for performing the "Data Transformation" step of oversampling.

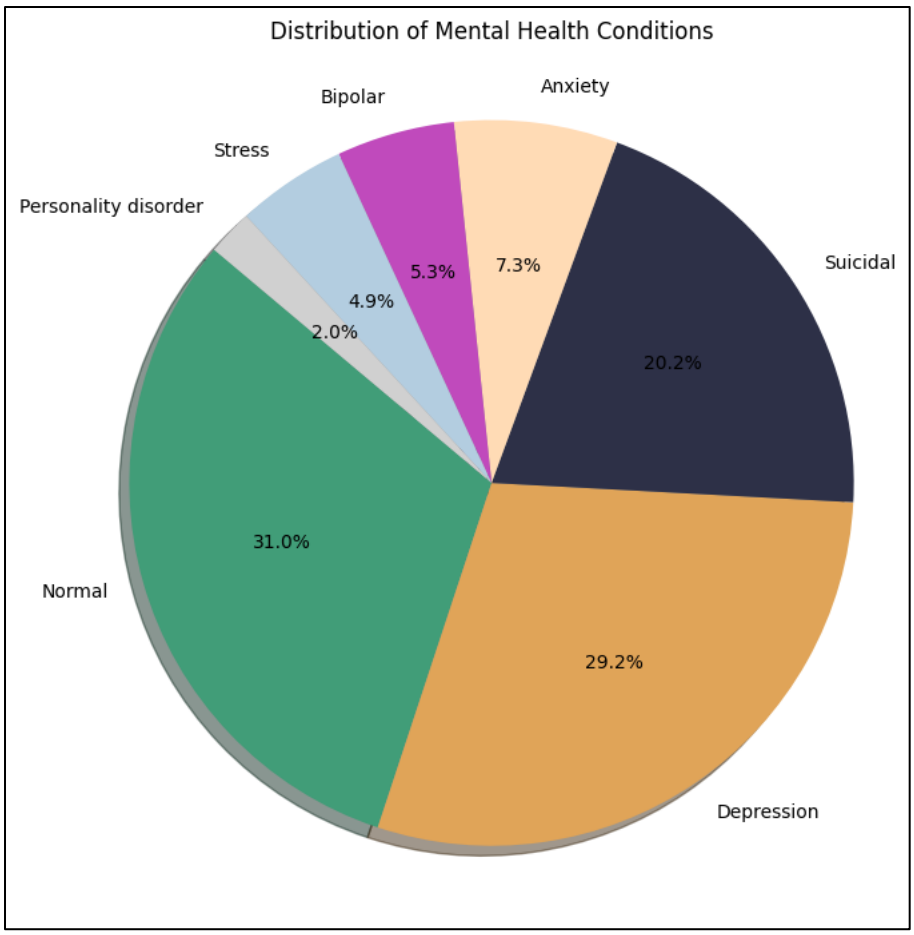

Figure 1.2 Distribution of status (Mental Health Conditions)

5. Feature Engineering

This project explored two different methods for feature engineering (turning text into numbers).

1. **Initial Method:** The project began by using a classic NLP technique called **TF-IDF (Term Frequency-Inverse Document Frequency)**. This method creates a large matrix where each column represents a word, and the value is a score based on how many times that word appears in a post, weighted by how rare it is in the entire dataset.
2. **Final Method:** We changed to a more advanced and powerful method called **WordPiece Tokenization**. This is the specific tokenizer used by the BERT model.

Rationale for Changing:

The TF-IDF method has a major weakness: it doesn't understand the meaning or context of words. It also requires "stemming" (e.g., chopping happiness down to happi), which causes a loss of information.

The **WordPiece** method is superior for this task. Instead of stemming, it breaks words into meaningful "sub-word" pieces (e.g., happiness becomes [happi] and [##ness]). This allows the model to learn the subtle relationship between happy and happiness on its own. Because this method is context-aware, it is the standard and "better" way to prepare data for a Transformer model like BERT.

6. Data Transformation

Three key transformations were performed on the cleaned data to prepare it for the model:

1. **Oversampling:** To solve the class imbalance discovered during EDA, **RandomOverSampler** was used. This technique randomly selected and duplicated posts from the minority classes (like 'Personality disorder') until every single class had the same number of samples (1,894), resulting in a perfectly balanced dataset of 13,258 rows.
2. **Label Encoding:** The model cannot read text labels. We used LabelEncoder from scikit-learn to convert the 7 unique 'status' strings (Anxiety, Bipolar, Depression, etc.) into numerical labels (0, 1, 2, etc.).
3. **Tokenization:** The BertTokenizer (using the WordPiece method) was applied to the statement column. This converted each sentence into two numerical vectors required by BERT:
    - input_ids: A list of numbers representing the sub-word tokens in the sentence.
    - attention_mask: A list of 1s and 0s that tells the model which tokens are real words, and which are just 'padding'.

Part 2: Model Exploration

1. Model Selection

- **Initial Baseline:** The project first explored classic machine learning models, including LogisticRegression, XGBoost and we use transformers to NLP BERT. These models were trained on the **TF-IDF** features.
    - Strengths: These models are extremely fast to train and easy to interpret. They provide a crucial **baseline performance score** to measure future improvements against.
    - Weaknesses: Their accuracy is limited because the TF-IDF features they use do not understand context. They can be easily confused by negation (e.g., "I am **not** happy") or sarcasm.

Test Classification Report:

	precision	recall	f1-score	support
Anxiety	0.84	0.86	0.85	768
Bipolar	0.87	0.82	0.84	556
Depression	0.78	0.74	0.76	3081
Normal	0.94	0.95	0.94	3269
Personality disorder	0.75	0.69	0.72	215
Stress	0.68	0.70	0.69	517
Suicidal	0.69	0.74	0.71	2131
accuracy			0.81	10537
macro avg	0.79	0.78	0.79	10537
weighted avg	0.81	0.81	0.81	10537

Figure 2.1 Test Classification Report of the Initial baseline model

- **Final Model:** We changed to a NLP **DistilBERT (distilbert-base-uncased)** model.

  Rationale: DistilBERT is a state-of-the-art **Transformer** model. It was chosen because it is pre-trained in a massive corpus of text and is specifically designed to understand context. By reading the whole sentence, it can learn the difference in meaning between "happy" and "not happy." This contextual understanding was the main reason for switching, as it was hypothesized to (and did) provide a massive boost in performance and accuracy for this nuanced task.

Test Classification Report:

	precision	recall	f1-score	support
Anxiety	0.97	0.98	0.97	370
Bipolar	0.99	0.99	0.99	402
Depression	0.81	0.71	0.75	381
Normal	0.90	0.91	0.91	353
Personality disorder	0.99	1.00	1.00	383
Stress	0.96	0.98	0.97	398
Suicidal	0.75	0.81	0.78	365
accuracy			0.91	2652
macro avg	0.91	0.91	0.91	2652
weighted avg	0.91	0.91	0.91	2652

Figure 2.2 Test Classification Report of the final model

2. Model Training

The model was fine-tuned using the Trainer class from the Hugging Face transformers library, which automates the training and evaluation process.

- **Hyperparameters:** The key settings used in the TrainingArguments were:
    - num_train_epochs = 2: The model passed over the entire training dataset two times.
    - per_device_train_batch_size = 16: The model was trained on 16 samples at a time.
    - save_strategy = "epoch": The model was saved after each epoch.
- **Validation:** We set eval_strategy="epoch" and load_best_model_at_end=True. This is a form of validation where the Trainer evaluates the model's performance on the test set after each of the two epochs. It then automatically identifies and loads the model from the epoch that achieved the best accuracy (in this case, Epoch 2), ensuring we are using the best possible version of the model.

3. Model Evaluation

The fine-tuned DistilBERT model's performance was significantly better than the baseline models.

- **Evaluation Metrics:** The final model achieved an overall **accuracy of 91.17%** on the unseen test set. The **weighted F1-score was 0.91**, indicating a strong balance between precision and recall across all classes.
- **Classification Report:** The detailed report showed very high performance for most classes, such as **Anxiety (0.97 F1)** and **Bipolar (0.99 F1)**. It also revealed the model's main challenge: it sometimes confused 'Depression' (0.75 F1) and 'Suicidal' (0.78 F1). This is a logical and understandable confusion, as the language used in these two states is often very similar.
- **Visualizations:**

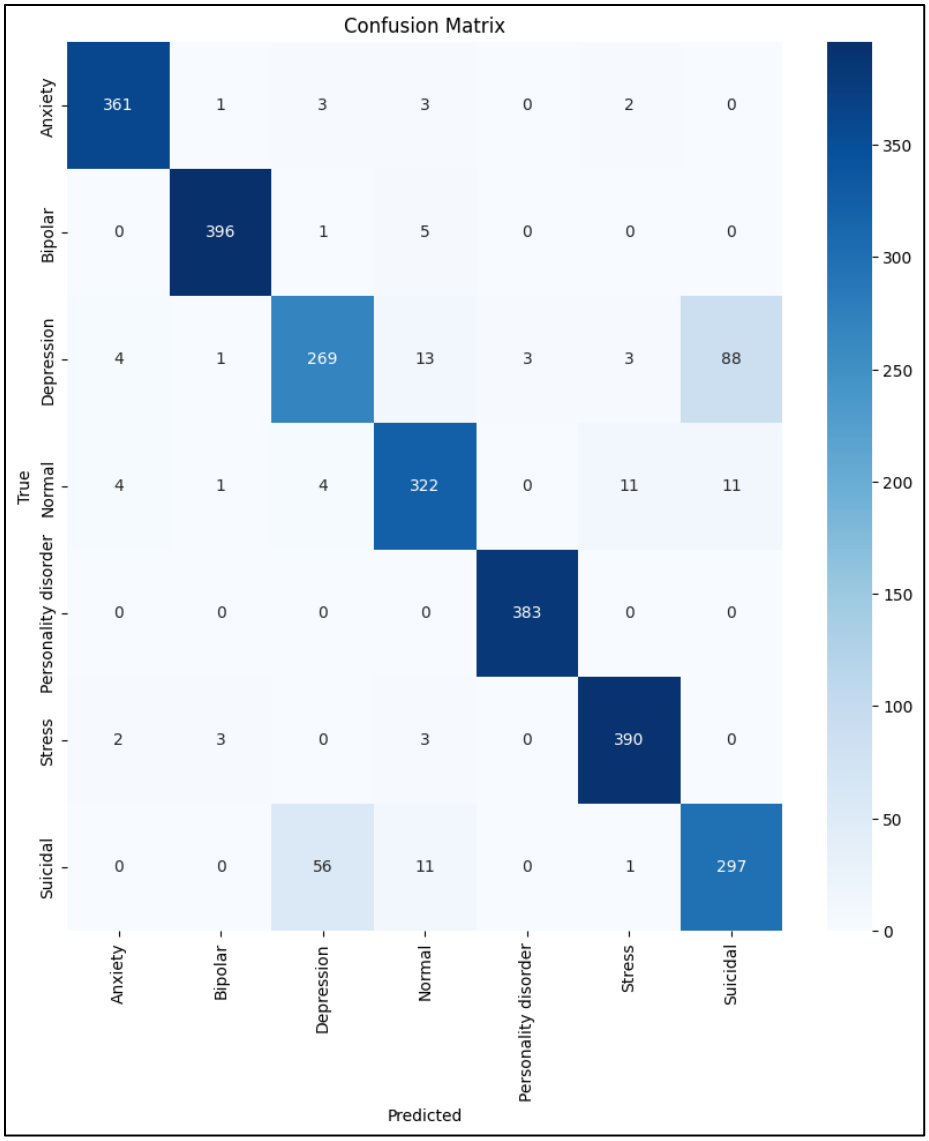

Fig.3.1 confusion matrix

4. Code Implementation

1. Data Cleaning Function: This function was used to preprocess all raw text.

Python

```
# --- Data Cleaning ---
import re
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))
```

```
def clean_statement(text):
  text = text.lower() # 1. Lowercase
  text = re.sub(r"[^a-zA-Z\s]", "", text) # 2. Remove non-letters
  text = ' '.join([word for word in text.split() if word not in stop_words]) # 3. Remove stopwords
  text = re.sub(r"\s+", " ", text).strip() # 4. Remove extra spaces
  return text

# Apply the cleaning
data['statement'] = data['statement'].apply(clean_statement)
```

2. Balancing and Encoding: These steps transformed the data shape and labels.

Python

```
# Data Transformation: Balancing
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(sampling_strategy='auto', random_state=42)
X_resampled, y_resampled = ros.fit_resample(data[['statement']], data['status'])
data_balanced = pd.DataFrame(X_resampled, columns=['statement'])
data_balanced['status'] = y_resampled

# --- 6. Data Transformation: Label Encoding ---
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
data_balanced['label'] = label_encoder.fit_transform(data_balanced['status'])
```

3. Tokenization and Model Training: This is the core of the BERT pipeline.

Python

```
# --- 5. Feature Engineering: Tokenization ---
from transformers import BertTokenizer
tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

# Tokenize the training and test sets
train_encoding = tokenizer(list(train_texts), padding=True, truncation=True, max_length=200)
test_encoding = tokenizer(list(test_texts), padding=True, truncation=True, max_length=200)

# --- 2. Model Training ---
from transformers import Trainer, TrainingArguments, AutoModelForSequenceClassification

model = AutoModelForSequenceClassification.from_pretrained(
    "distilbert-base-uncased",  # Load the DistilBERT model
    num_labels=len(label_encoder.classes_) # Tell it we have 7 classes
)

training_args = TrainingArguments(
    output_dir="./results",
    eval_strategy="epoch",
    num_train_epochs=2,
    load_best_model_at_end=True, # Use validation to pick best model
    metric_for_best_model="accuracy",
)
```

```
trainer = Trainer(
    model=model,
    args=training_args,
    train_dataset=train_dataset, # The tokenized training data
    eval_dataset=test_dataset,   # The tokenized test data
    compute_metrics=compute_metrics, # Function to calculate accuracy/F1
)

# Start the training
trainer.train()
```

Remark: currently our model can only work with English language, if any other language input it will shows as normal. For the solution we considered to use the translation algorithms for multiple language or fine tuning the model with Myanmar language data.

Ref: 1. Sarkar, S. (n.d.) Sentiment analysis for mental health [dataset]. Kaggle. Available at: https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health/data