UNIVERSITY OF FLORIDA

DEPARTMENT OF ENGINEERING EDUCATION

TECHNICAL REPORT

ON

AI POWERED MENTAL HEALTH CHATBOT

SUBMITTED BY

SAI SATWIK YARAPOTHINI (56927137)

In partial fulfillment of the requirement for the award of Master of Science in Applied Data Science

TABLE OF CONTENTS

Objective of the Project

The objective of this project is to develop and deploy an **Intelligent mental health support chatbot** that leverages Machine Learning and Natural Language Processing to provide empathetic, intent-aware responses to user messages. This AI-powered chatbot is designed to serve as a virtual assistant for individuals seeking emotional or informational support related to mental health.

- The chatbot is mainly capable of:
  - **Classifying the user's intent** (e.g., emotional support, general inquiry) using a trained ML model.
  - **Generating context-aware, human-like responses** using a lightweight Large language model (LLM) facebook/blenderbot-400M-distill.

- The chatbot is built upon a Unified Modeling dataset derived from three complementary sources:
  1. Counseling conversations for emotional support.
  2. Frequently asked mental health questions (FAQs).
  3. Sentiment-labeled mental health statements.

These datasets were preprocessed, integrated, and used to train a Random Forest-based Intent Classifier and drive a responsive dialogue using the Blenderbot model. The entire pipeline adheres to the CRISP-DM methodology which consists of steps like Data Collection & Understanding, Preprocessing & Feature Engineering, Model Training and Evaluation and Final Deployment as a Gradio-based Conversational Chatbot.

By the end of this milestone, the chatbot has been successfully deployed with a clean, interactive UI, supporting real-time mental health support simulations based on intelligent intent prediction and empathetic language generation.

Type of Tool

This project involves the design and deployment of a Conversational AI Chatbot focused on Mental health support, powered by Supervised Machine learning and Natural language processing (NLP) techniques. The end product is an intelligent assistant that predicts the **Intent** behind user messages and generates Empathetic, context-aware responses in real-time.

The tool developed in this project is a **Conversational Agent**, deployed using **Gradio**, and equipped to simulate human-like interactions while understanding the emotional or informational needs of the user. It combines the strength of classical ML models for classification with the fluency of a transformer-based LLM to create an engaging and helpful mental health support system.

The chatbot pipeline is comprised of two core components:

- **Intent Classification**: A **Random Forest classifier** trained to categorize incoming user messages into **Emotional support**, **General support**, or **FAQ-based informational queries**.
- **Response Generation**: A **Blenderbot-400M-distill** model used for generating human-like responses that match the predicted intent, leveraging the full conversational history for more contextually appropriate replies.

The focus of this final milestone was to integrate and deploy these components into a seamless **Multi-turn conversational chatbot UI**, where each user message is Interpreted for intent and then logged into the chat history and then chatbot responded using an LLM with contextual memory. Unlike traditional FAQ bots, this tool adapts to user tone and intent in real-time, offering a more supportive, safe, and human-like experience, especially vital in sensitive domains such as mental health.

Data to Be Used

This project utilizes three key datasets to train and improve the chatbot's ability to provide accurate and context-aware responses. The datasets cover mental health counseling conversations, structured FAQs, and sentiment-labeled emotional statements. Each dataset has been acquired from reliable sources and has been verified for accessibility and compliance with licensing restrictions.

❖ Dataset 1: Mental Health Counseling Conversations

- **Source**: Hugging Face
- **Dataset Name**: `Amod/Mental Health Counseling Conversations`
- **URL**: https://huggingface.co/datasets/Amod/mental_health_counseling_conversations
- **License**: Permitted for academic use, sourced from online therapy platforms.

Content:
This dataset contains question & answer pairs sourced from online counseling and therapy platforms. It features real mental health queries and psychologist responses, making it ideal for training a chatbot that provides context-aware therapy guidance.

Dataset Properties:

- Number of Entries: 3,512
- Columns: `Context` (User's question or concern), `Response` (Professional therapist's answer)

Use Case in Chatbot:
This dataset is used for training and testing the chatbot for `general_support` intent.

❖ Dataset 2: Mental Health FAQ for Chatbot

- **Source**: Kaggle
- **Dataset Name**: `Mental Health FAQ for Chatbot`
- **URL**: https://www.kaggle.com/datasets/narendrageek/mental-health-faq-for-chatbot
- **License**: Available for academic use, sourced from mental health organizations.

Content:
A small but focused dataset containing commonly asked mental health questions along with concise expert responses. It is structured to cover various FAQs related to symptoms, treatments, and general knowledge.

Dataset Properties:

- Number of Entries: 98
- Columns: `Questions`, `Answers`

Use Case in Chatbot:
It is used for `faq` intent, to answer factual and informational queries. It also assisted in engineering topic-based rule-driven features.

❖ Dataset 3: Sentiment Analysis for Mental Health

- **Source**: Kaggle
- **Dataset Name**: `Sentiment Analysis for Mental Health`
- **URL**: https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health
- **License**: Available for academic use, sourced from social media and mental health forums.

Content:
A curated collection of real-world user-generated posts labeled with various mental health statuses. This dataset enhances the chatbot's ability to understand emotionally charged language in a non-clinical setting.

Dataset Properties:

- Number of Entries: 53,043
- Columns: `Statement`, `Status` (Mental health condition label – Normal, Depression, Suicidal, Anxiety, Stress, Bipolar, Personality Disorder)

Use Case in Chatbot:
Used for `emotional_support` intent to identify and handle sensitive emotional inputs more effectively.

❖ Unified Dataset for Chatbot Modeling

Source: Custom-merged from the above three datasets.

Purpose:
To create a consistent modeling dataset combining emotional, counseling, and factual queries from multiple sources. It also integrates engineered features across datasets to support the ML pipeline for intent prediction.

Dataset Properties:

- Number of Entries: 49,228
- Columns:
  - `Input`: Cleaned user message or question
  - `Intent`: Task type (e.g., `emotional_support`, `faq`, `general_support`)
  - `Response`: Therapist reply, FAQ answer, or generic support text
  - `Statement_Word_Count`, `Has_Severe_Keyword`
  - `Question_Word_Count`, `Contains_Mental_Health_Topic`

- `Context_Word_Count`, `Response_Word_Count`, `Contains_Therapy_Method`

Use Case in Chatbot:
This unified dataset serves as the primary modeling dataset for training the **Intent Classifier**, which routes user input to the appropriate LLM-based response generator.By combining diverse sources into a consistent training pipeline, the chatbot ensures domain coverage, emotional sensitivity, and response contextuality.

Tech Stack

The development of this AI-powered mental health chatbot spanned the entire CRISP-DM lifecycle, involving extensive preprocessing, modeling, evaluation, and deployment. Below is the full tech stack used for each phase of the pipeline:

1. Programming Language

- **Python**: Used for all aspects of development including data processing, ML model building, and Interface deployment.

2. Data Preprocessing & Analysis

- **Pandas, NumPy**: For loading, cleaning, merging, and statistical summarization of datasets.
- **NLTK, SpaCy**: Used for NLP preprocessing tasks such as tokenization, stopword removal, and lemmatization.
- **re (Regular Expressions)**: For text normalization and pattern matching during feature engineering.

3. Data Visualization & EDA

- **Matplotlib, Seaborn**: For producing visual plots such as class distributions, boxplots, and correlation heatmaps.
- **WordCloud**: For text-based visualizations highlighting common words in mental health statements.

4. Feature Engineering & Selection

- **Scikit-learn**:
  - TF-IDF Vectorizer for text vectorization.
  - Feature importance and dimensionality checks.
- **Manual Feature Creation**:
  - Features like word counts, presence of severe keywords, and mental health keyword flags were created and normalized.

5. Machine Learning Models

- **Scikit-learn**:
  - Random Forest Classifier (for Intent Classification)
  - Model selection was based on validation performance and alignment with vectorizer dimensions.

6. Text Vectorization

- **TF-IDF (Scikit-learn)**: Used for converting text inputs into feature vectors suitable for ML models. Feature limits were set to match model requirements (e.g., 169 features for intent prediction).

7. LLM-based Chatbot Response Generation

- **Hugging Face Transformers**:
  - Model: `facebook/blenderbot-400M-distill`
  - Used for generating multi-turn, empathetic responses locally (without requiring an external API).
- **Tokenizer**: BlenderbotTokenizer was used to manage input formatting and truncation.

8. Chatbot UI & Deployment

- **Gradio**: Used to develop and deploy the final interactive conversational chatbot with multi-turn memory and real-time intent prediction. Gradio Blocks and gr.Chatbot components enabled a clean and responsive chat-like interface for end users.

9. Version Control & Code Management

- **Git & GitHub**: All code and milestone reports were managed in a private GitHub repository for version tracking and collaboration.

10. Reporting & Presentation

- **Microsoft Word**: For writing and formatting the PDF reports submitted for all milestones.
- **Microsoft PowerPoint**: Used to prepare and deliver the final project presentation and demo.

Exploratory Data Analysis (EDA) Report

- The purpose of Exploratory Data Analysis (EDA) is to gain insights from the final unified dataset that will be used to develop the AI-powered mental health chatbot.

- The primary goals of EDA were:

    - To validate the distribution and usefulness of engineered features.
    - To identify class imbalances in the target variables (`Intent`, `Label`).
    - To assess correlations and decide on suitable features for model training.

Key Plots & Visualizations:

❖ Distribution of Statement Word Count (Intent: emotional_support)

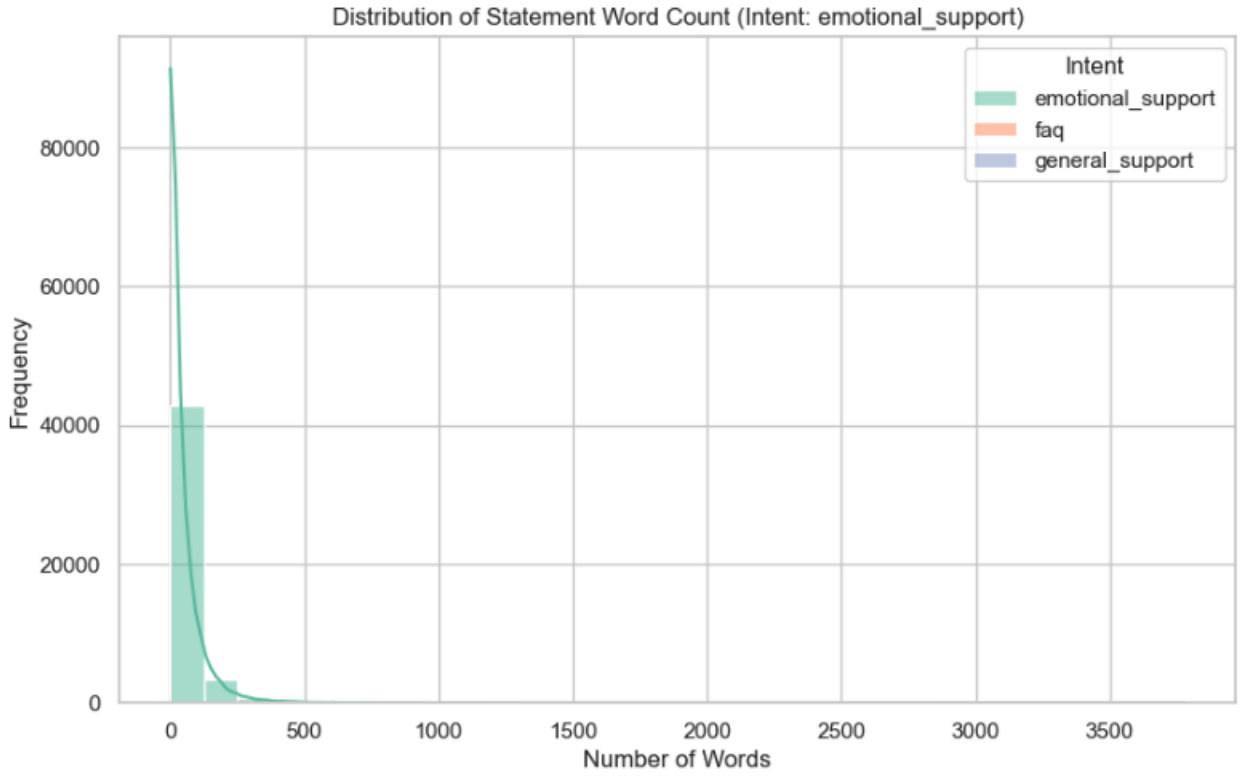


Fig 1: Distribution of Statement Word Count (Intent: emotional_support)

As seen in **Fig 1**, the `Statement_Word_Count` distribution is heavily Right-skewed, with most emotional support inputs falling below 50 words. However, there exists a noticeable tail of longer inputs, some extending beyond 1000 words. This indicates that users expressing emotional concerns often vary drastically in how much they share from brief statements to deeply detailed emotional outpourings.

Implication for Feature Engineering:
The presence of this long-tail distribution confirms that `Statement_Word_Count` is a highly informative feature. It captures a key behavioral trait the Extent of emotional elaboration in a user's message. This is especially relevant for **Detecting Emotional severity** and distinguishing subtle user intents within the emotional_support category.

Implication for Modeling:
This feature is used in both the **Intent Classifier** and **Label Classifier** for emotional_support inputs, as it helps the model discern between casual check-ins versus potentially critical or depressive statements.

❖ Distribution of Question Word Count (Intent: faq)

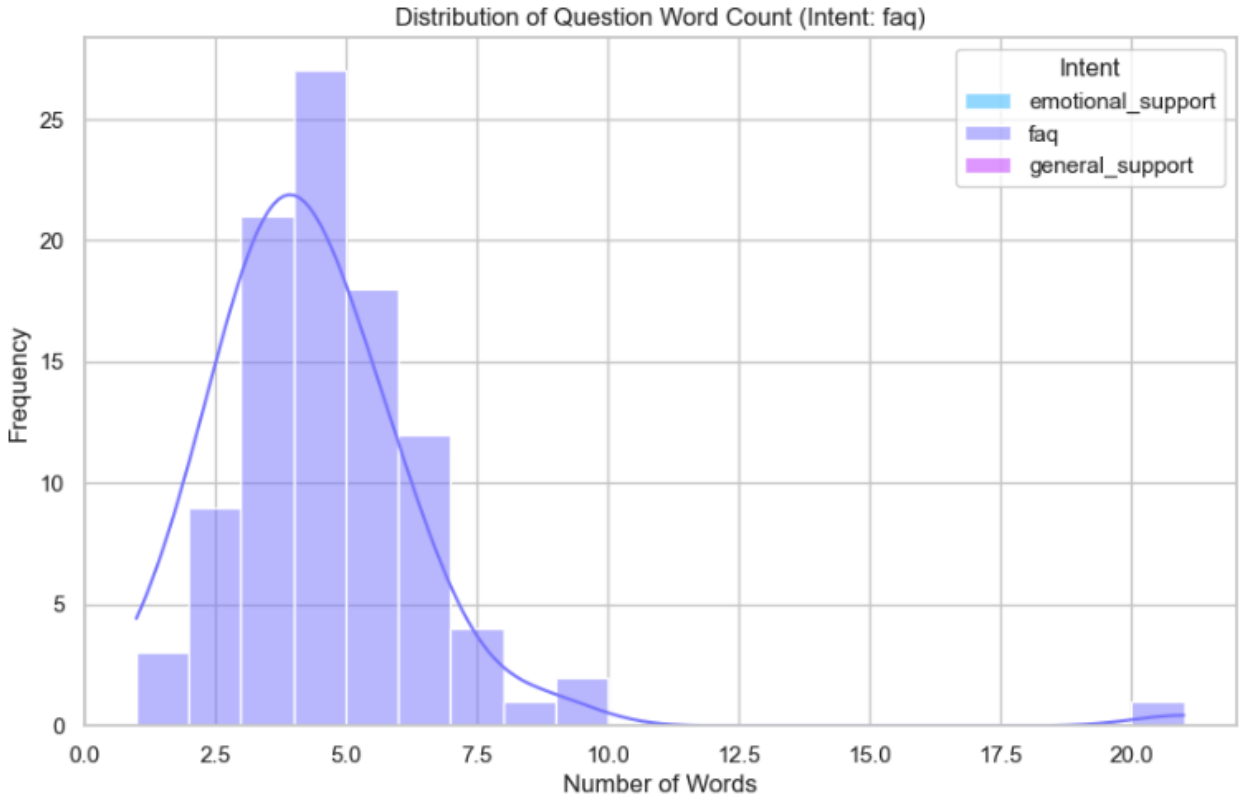


Fig 2: Distribution of Question Word Count (Intent: faq)

In **Fig 2**, we observe that FAQ-style questions tend to be very short and direct. Most fall within the 3–6 word range, forming a clear Left-skewed distribution. This pattern aligns with typical user behavior in informational queries, which are generally phrased as brief questions like "can anxiety be cured" or "what is bipolar disorder."

Implication for Feature Engineering:
The engineered feature `Question_Word_Count` provides clear separation for FAQ-style intent, which is important when distinguishing between question-based factual queries and conversational or therapeutic messages.

Implication for Modeling:
The feature becomes especially useful in the **Intent Classifier**, where distinguishing `faq` from `emotional_support` requires attention to input brevity. It also plays a role in the **Label Classifier** for FAQ entries, where question length may correlate with complexity of the topic (e.g., medication vs. recovery).

❖ Distribution of Response Word Count (Intent: general_support)

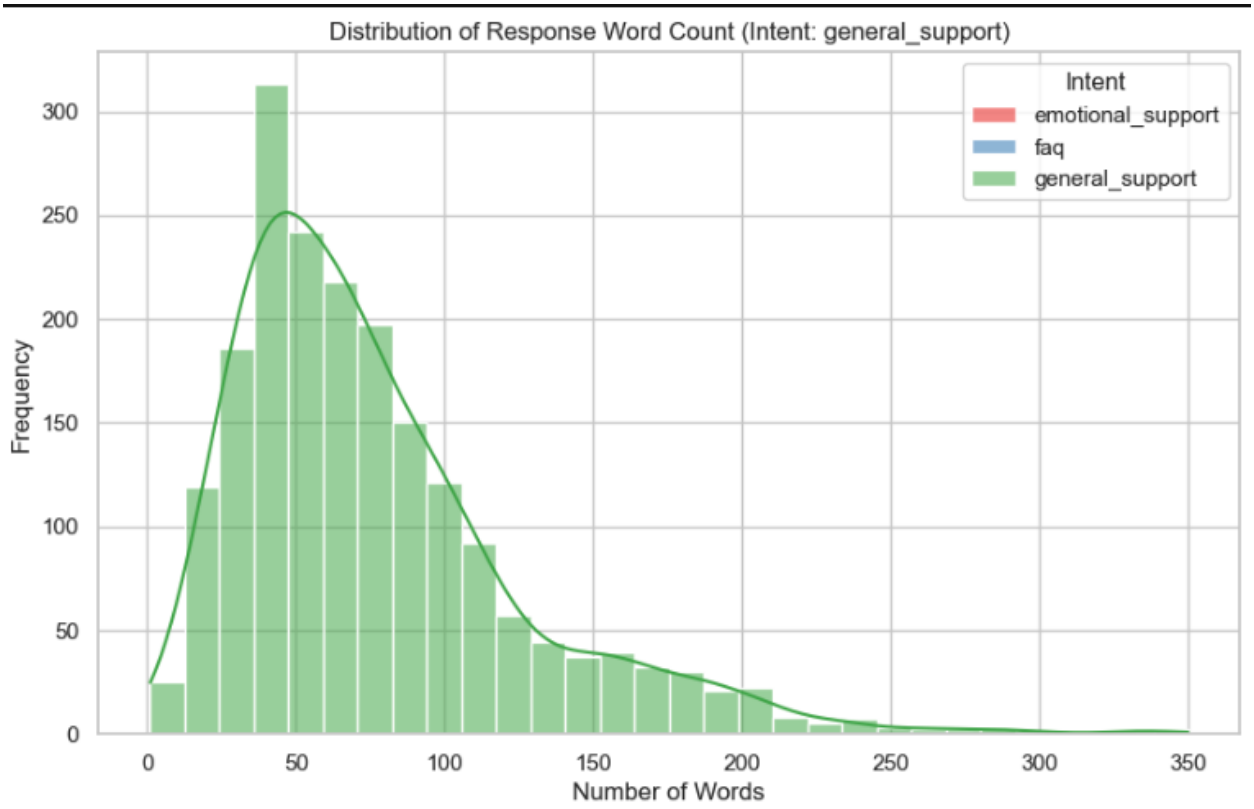


Fig 3: Distribution of Response Word Count (Intent: general_support)

This plot (Fig 3) presents the distribution of word count in therapist responses for the general_support intent. The peak of the distribution is around 40–50 words, with most responses falling below 150 words, but a long tail extending even further. This shows that therapists tend to offer Moderately detailed guidance**,** offering validation, suggestions, and structured advice without being overly verbose

Implication for Feature Engineering:

The `Response_Word_Count` feature reflects response depth and supportive content richness. This can be leveraged to help match the tone and structure of responses during chatbot response selection in future stages.

Implication for Modeling:

While not used directly in classification tasks in this milestone, this feature will be important in later response ranking tasks and could support additional dialogue state modeling.

❖ Presence of Severe Keywords in Emotional Support Inputs:

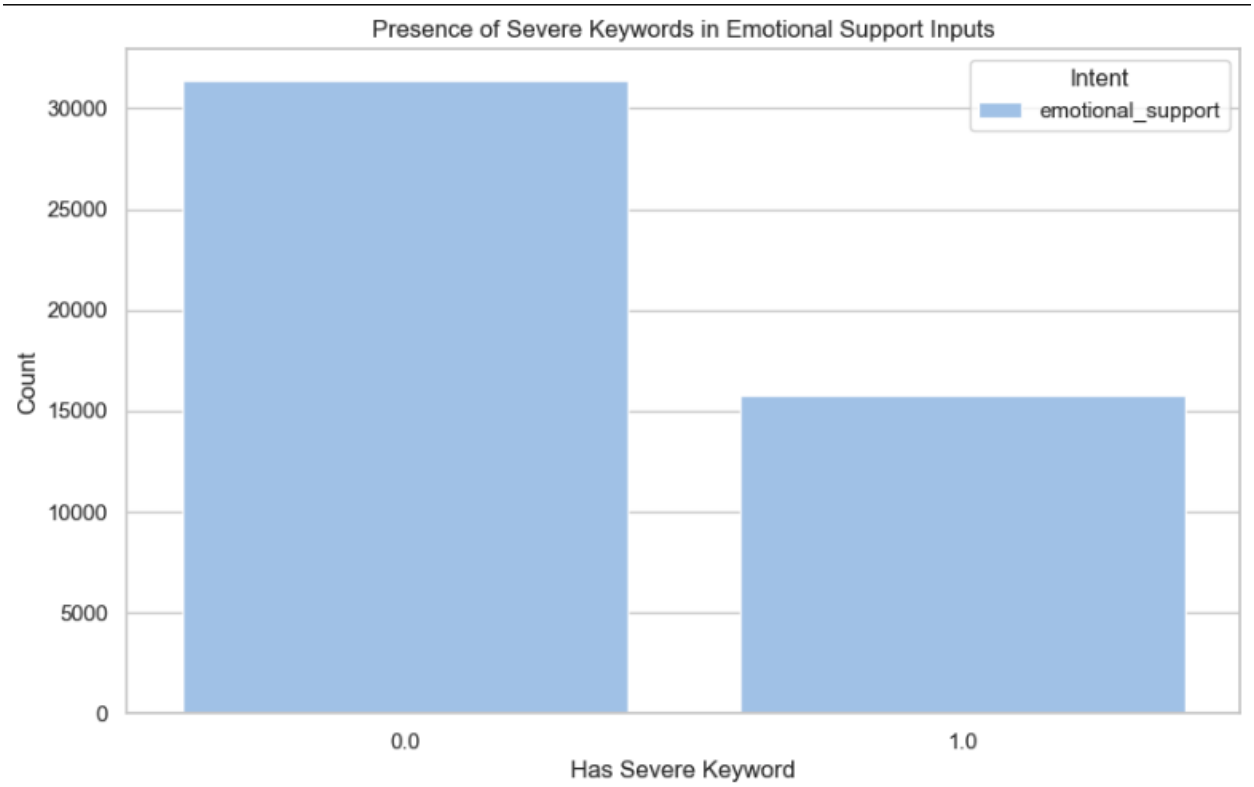


Fig 4: Presence of Severe Keywords in Emotional Support Inputs

The Fig 4 depicts the distribution of the binary feature `Has_Severe_Keyword` for emotional_support messages. While over 30,000 statements did not contain flagged severe words, nearly 16,000 entries were identified as containing critical phrases like “hopeless,” “suicidal,” or “worthless.” This confirms that high-risk user statements are significantly represented in the dataset.

Implication for Feature Engineering:

This engineered feature is critical for flagging inputs that may warrant special handling or priority.it provides a semantic shortcut for emotional intent severity, which might otherwise require deeper NLP modeling to extract.

Implication for Modeling:

The `Has_Severe_Keyword` feature is used in both the **Intent Classifier** and **Label Classifier** for emotional inputs. It helps models focus on emotionally intense statements and improves classification accuracy for severe sentiment labels (e.g., depression, suicidal thoughts).

❖ Top 10 Label Categories:

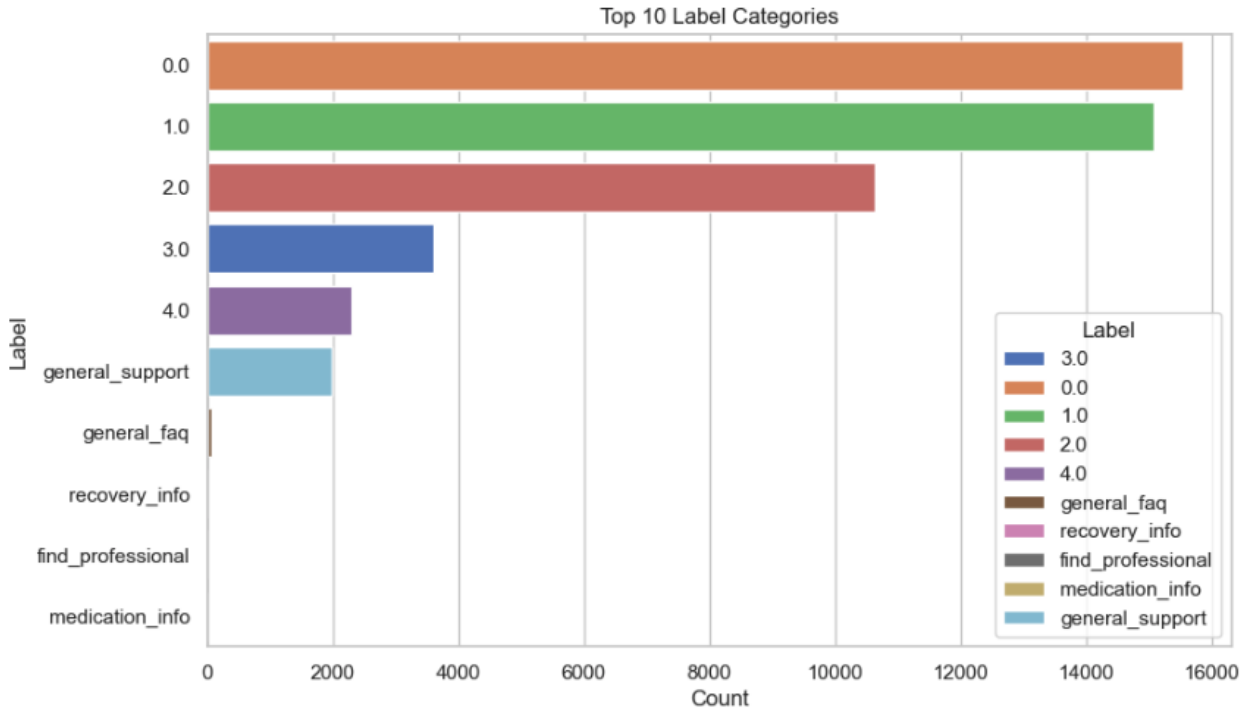


Fig 5 : Top 10 Label Categories

Fig 5 shows the most common label values across the entire dataset. The majority of records are numerically labeled (e.g., 0.0 to 4.0), corresponding to sentiment classes from the sentiment analysis dataset. Non-numeric labels such as `find_professional` and `general_support` represent FAQ and counseling-related categories.

Implication for Feature Engineering:
This visualization highlights the multi-source nature of the `Label` variable. Labels differ in meaning depending on the intent, and thus per-intent label modeling is necessary.

Implication for Modeling:
The **Label Classifier** must be trained independently for each intent type. The features that help predict sentiment-based labels (e.g., `Statement_Word_Count`) are irrelevant to FAQ categories and vice versa.

❖ Correlation Heatmap of Numerical Features

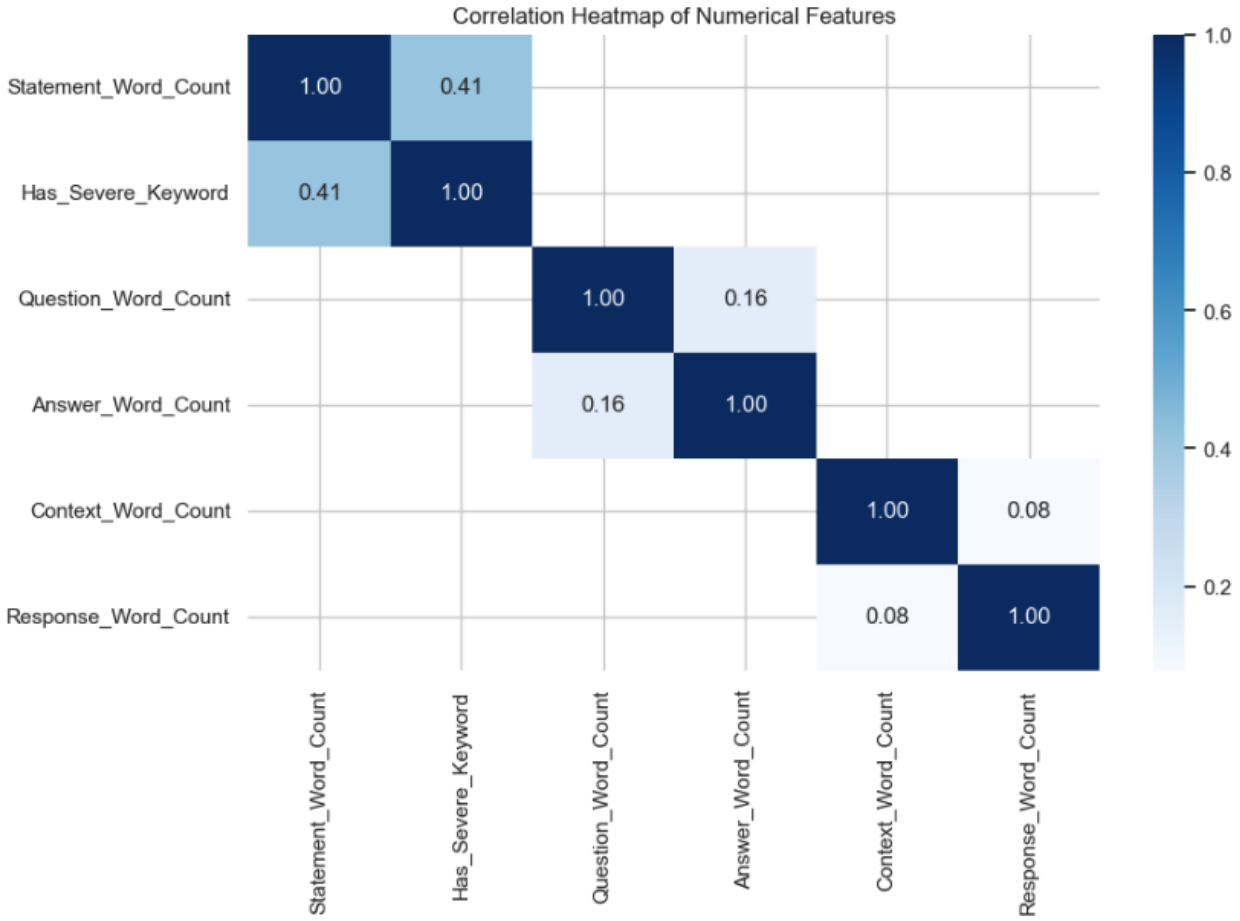


Fig 6: Correlation Heatmap of Numerical Features

Fig 6 presents the Pearson correlation matrix for all engineered numerical features. The strongest observed correlation is between `Statement_Word_Count` and `Has_Severe_Keyword` (~0.41), suggesting that longer statements are somewhat more likely to contain severe terms. Other correlations, such as between `Question_Word_Count` and `Answer_Word_Count`, are mild (~0.16), and most other features appear to be largely independent.

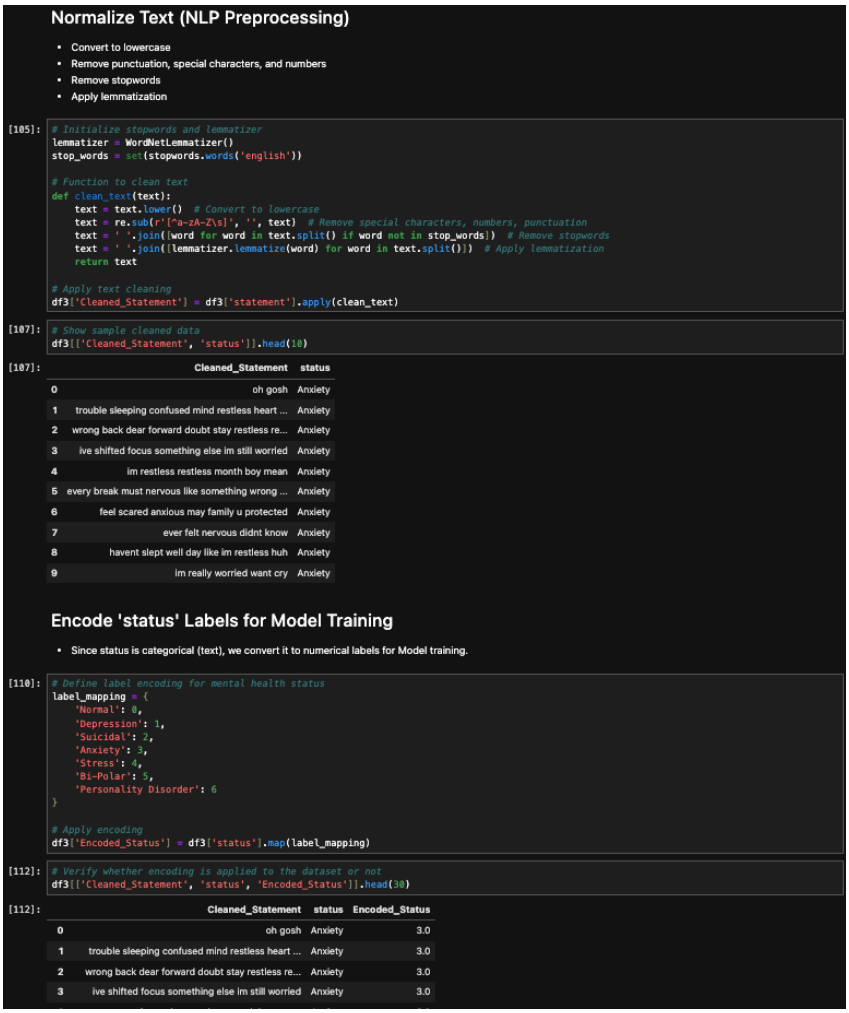
Implication for Feature Engineering:
This confirms that the engineered features are **Non-redundant** and can be safely used together in multivariate models.

Implication for Modeling:
We decided to retain all numerical features and scale them appropriately before model training. Since no multicollinearity is observed, dimensionality reduction (e.g., PCA) is not necessary for this task.

Feature Engineering

Feature engineering played a central role in transforming the unified mental health dataset into a format suitable for training intelligent classification models. The final dataset was a merged and cleaned combination of three distinct sources counseling conversations, mental health FAQs, and sentiment-labeled user statements. Since these sources varied significantly in structure and purpose, we created engineered features that helped unify and enhance the dataset while retaining its contextual richness.

Among the engineered features, `Statement_Word_Count` and `Has_Severe_Keyword` were derived from the sentiment analysis dataset as shown in Fig 7 and were applicable only to rows where the intent was emotional_support. `Statement_Word_Count` captured the length of a user's emotional message, which, as illustrated in Fig 1, followed a long-tailed distribution. This indicated wide variation in how much users tend to share, ranging from brief updates to emotionally complex narratives. `Has_Severe_Keyword`, on the other hand, was a binary indicator signaling the presence of high-risk phrases like “worthless” or “suicidal.” As seen in Fig 4, this feature had strong class separation potential for modeling mental health severity, making it highly relevant for emotional classification tasks.



Normalize Text (NLP Preprocessing)

- Convert to lowercase
- Remove punctuation, special characters, and numbers
- Remove stopwords
- Apply lemmatization

[105]:
```
# Initialize stopwords and lemmatizer
lemmatizer = WordNetLemmatizer()
stop_words = set(stopwords.words('english'))

# Function to clean text
def clean_text(text):
    text = text.lower()  # Convert to lowercase
    text = re.sub(r'[^a-zA-Z\s]', '', text)  # Remove special characters, numbers, punctuation
    text = ' '.join([word for word in text.split() if word not in stop_words])  # Remove stopwords
    text = ' '.join([lemmatizer.lemmatize(word) for word in text.split()])  # Apply lemmatization
    return text

# Apply text cleaning
df3['Cleaned_Statement'] = df3['statement'].apply(clean_text)
```

[107]:
```
# Show sample cleaned data
df3[['Cleaned_Statement', 'status']].head(10)
```

[107]:

	Cleaned_Statement	status
0	oh gosh	Anxiety
1	trouble sleeping confused mind restless heart ...	Anxiety
2	wrong back dear forward doubt stay restless re...	Anxiety
3	ive shifted focus something else im still worried	Anxiety
4	im restless restless month boy mean	Anxiety
5	every break must nervous like something wrong ...	Anxiety
6	feel scared anxious may family u protected	Anxiety
7	ever felt nervous didnt know	Anxiety
8	havent slept well day like im restless huh	Anxiety
9	im really worried want cry	Anxiety

Encode 'status' Labels for Model Training

- Since status is categorical (text), we convert it to numerical labels for Model training.

[110]:
```
# Define label encoding for mental health status
label_mapping = {
    'Normal': 0,
    'Depression': 1,
    'Suicidal': 2,
    'Anxiety': 3,
    'Stress': 4,
    'Bi-Polar': 5,
    'Personality Disorder': 6
}

# Apply encoding
df3['Encoded_Status'] = df3['status'].map(label_mapping)
```

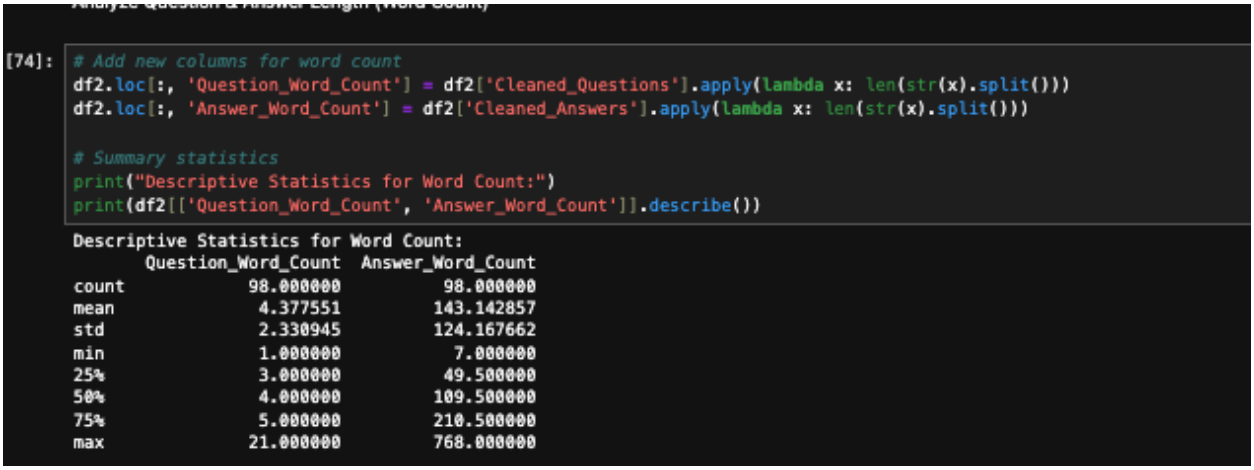
[112]:
```
# Verify whether encoding is applied to the dataset or not
df3[['Cleaned_Statement', 'status', 'Encoded_Status']].head(30)
```

[112]:

	Cleaned_Statement	status	Encoded_Status
0	oh gosh	Anxiety	3.0
1	trouble sleeping confused mind restless heart ...	Anxiety	3.0
2	wrong back dear forward doubt stay restless re...	Anxiety	3.0
3	ive shifted focus something else im still worried	Anxiety	3.0

Fig 7: Feature Engineering on Sentiment Analysis Dataset

From the FAQ dataset, `Question_Word_Count` and `Answer_Word_Count` were engineered to quantify the length of questions and their corresponding responses a shown in Fig 8. Figure 2 shows that FAQ questions are typically concise, with most entries containing fewer than 6 words, helping models identify question-style queries. `Answer_Word_Count`, though not directly used in classification models, has been preserved for future tasks involving chatbot response selection.



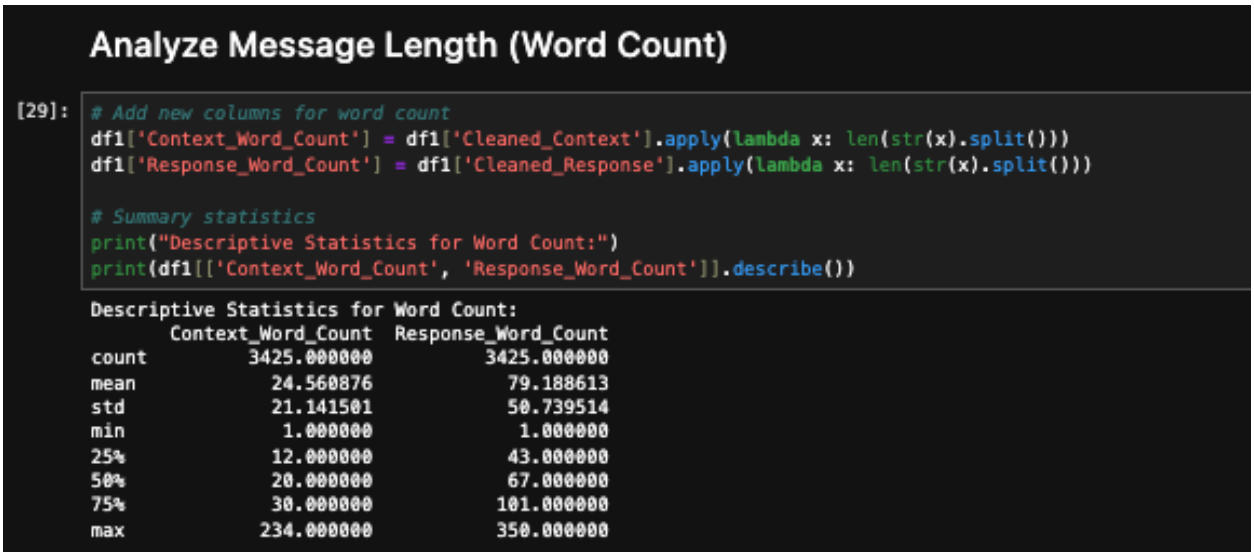
```
[74]: # Add new columns for word count
df2.loc[:, 'Question_Word_Count'] = df2['Cleaned_Questions'].apply(lambda x: len(str(x).split()))
df2.loc[:, 'Answer_Word_Count'] = df2['Cleaned_Answers'].apply(lambda x: len(str(x).split()))

# Summary statistics
print("Descriptive Statistics for Word Count:")
print(df2[['Question_Word_Count', 'Answer_Word_Count']].describe())
```

```
Descriptive Statistics for Word Count:
       Question_Word_Count  Answer_Word_Count
count            98.000000          98.000000
mean              4.377551         143.142857
std               2.330945         124.167662
min               1.000000           7.000000
25%               3.000000          49.500000
50%               4.000000         109.500000
75%               5.000000         210.500000
max              21.000000         768.000000
```

Fig 8: Feature Engineering on FAQ Dataset

Similarly, for the counseling dataset, features like `Context_Word_Count` and `Response_Word_Count` were extracted to model the depth of therapy conversations using the code as shown in Fig 9. As evident from Fig 3, therapist responses generally ranged between 40 to 150 words, supporting the use of this feature in modeling response complexity and possibly aiding in tailoring chatbot-generated replies in the deployment phase.



Analyze Message Length (Word Count)

```
[29]: # Add new columns for word count
df1['Context_Word_Count'] = df1['Cleaned_Context'].apply(lambda x: len(str(x).split()))
df1['Response_Word_Count'] = df1['Cleaned_Response'].apply(lambda x: len(str(x).split()))

# Summary statistics
print("Descriptive Statistics for Word Count:")
print(df1[['Context_Word_Count', 'Response_Word_Count']].describe())
```

```
Descriptive Statistics for Word Count:
       Context_Word_Count  Response_Word_Count
count         3425.000000          3425.000000
mean            24.560876            79.188613
std             21.141501            50.739514
min              1.000000             1.000000
25%             12.000000            43.000000
50%             20.000000            67.000000
75%             30.000000           101.000000
max            234.000000           350.000000
```

Fig 9: Feature Engineering on Counselling Dataset

To ensure modeling integrity, only relevant features were used per classifier. For example, emotional support label classification used only features engineered from emotional statements,

while FAQ-related label prediction used question-based features. This per-intent filtering ensured no data leakage or invalid imputations occurred.

- Final merged Unified dataset after relevant feature engineering looked like this:

Final enriched dataset shape: (49228, 12)

	Input	Intent	Label	Response	Statement_Word_Count	Has_Severe_Keyword	Question_Word_Count	Answer_Word_Count
19867	diagnosed early life always dealt gotten bit o...	emotional_support	1.0		51.0	1.0	NaN	NaN
41081	sleep face kinda swollen let allergic thing ge...	emotional_support	0.0		23.0	0.0	NaN	NaN
35900	effyobie stop speaking sophisticated way pleas...	emotional_support	0.0		3.0	0.0	NaN	NaN
45523	article really helpful anxiety disorder contro...	emotional_support	4.0		71.0	1.0	NaN	NaN
5832	twtan sinking ekw	emotional_support	0.0		3.0	0.0	NaN	NaN
21716	really regret iti chance finally end allnow ca...	emotional_support	2.0		16.0	1.0	NaN	NaN
39623	husband reading cpt depression tweet life newb...	emotional_support	1.0		63.0	1.0	NaN	NaN
2295	certain people government increasingly stupid ...	emotional_support	0.0		3.0	0.0	NaN	NaN
27904	going chat therapist next session need find so...	emotional_support	0.0		45.0	0.0	NaN	NaN
21254	wish grew healthier relationship hope least de...	emotional_support	1.0		255.0	1.0	NaN	NaN
28602	today really bad day energy cried thought suic...	emotional_support	4.0		27.0	0.0	NaN	NaN
48916	im guy dont like girl like guy mean im gay	general_support	general_support	doesnt sound like finding attracted anyone cou...	NaN	NaN	NaN	NaN

Fig 10: Final Unified Dataset after Feature Engineering

Categorical Variable Encoding

The unified dataset consisted of three types of labels — `Intent`, `Label`, and `Response`, each serving a specific modeling purpose. For classification, encoding categorical variables was necessary, but it had to be done selectively and contextually to preserve semantic integrity across intent types as shown in Unified dataset (Fig. 10)

- The `Intent` column in the unified dataset contained three unique values: `emotional_support`, `faq`, and `general_support`. For model training, we applied **Label Encoding** to convert these intent strings into numerical format, but only during internal processing (e.g., during stratified train-test split). In all outputs and evaluations, class names were retained to maintain interpretability.
- The `Label` column was more nuanced. For the `emotional_support` intent, the label was already present as a numerical column named `Encoded_Status`, representing categories such as Normal, Depression, and Anxiety. For the `faq` intent, however, labels were manually derived using a **Rule-based keyword mapping** of FAQ questions. Categories like `find_professional`, `medication_info`, and `recovery_info` were created by applying a simple keyword-based classifier function. This allowed us to convert unlabeled FAQ entries into structured, trainable categories. Finally, for the `general_support` intent, all rows were assigned the fixed label `general_support`, since no subcategories were available or needed.
- Encoding was applied **per intent** using `LabelEncoder()` inside loops to prevent accidental mixing of semantically different labels. This ensured that numeric values assigned to sentiment categories (e.g., 0 for Normal) didn't overlap with unrelated categories like `recovery_info` from FAQs. This structured approach allowed intent-specific models to be trained with contextually correct encodings as shown in following code snippet Fig 11

```
#Importing Necessary Libraries
import pandas as pd

# Load all the 3 datasets we have
df1 = pd.read_csv("/Users/satwik/Documents/GitHub/cap5771sp25-project/Data/Cleaned Data/Cleaned_Counseling_Conversations.csv")
df2 = pd.read_csv("/Users/satwik/Documents/GitHub/cap5771sp25-project/Data/Cleaned Data/Cleaned_Mental_Health_FAQ.csv")
df3 = pd.read_csv("/Users/satwik/Documents/GitHub/cap5771sp25-project/Data/Cleaned Data/Cleaned_Sentiment_Analysis.csv")

# Generating Label for Dataset 2 : Rule-based Labeling for FAQ
def map_faq_label(question):
    question = str(question).lower()
    if "medication" in question:
        return "medication_info"
    elif "treatment" in question:
        return "treatment_options"
    elif "professional" in question or "doctor" in question:
        return "find_professional"
    elif "recover" in question or "heal" in question:
        return "recovery_info"
    else:
        return "general_faq"

df2["Label"] = df2["Cleaned_Questions"].apply(map_faq_label)
```

```
# Creating Modeling Views of each dataset

# Dataset 1 → Counseling Conversations
df1_model = df1[["Cleaned_Context", "Cleaned_Response"]].copy()
df1_model.rename(columns={"Cleaned_Context": "Input", "Cleaned_Response": "Response"}, inplace=True)
df1_model["Intent"] = "general_support"
df1_model["Label"] = "general_support"
```

```
# Dataset 2 → Mental Health FAQ
df2_model = df2[["Cleaned_Questions", "Cleaned_Answers", "Label"]].copy()
df2_model.rename(columns={"Cleaned_Questions": "Input", "Cleaned_Answers": "Response"}, inplace=True)
df2_model["Intent"] = "faq"
```

```
# Dataset 3 → Sentiment Analysis
df3_model = df3[["Cleaned_Statement", "Encoded_Status"]].copy()
df3_model.rename(columns={"Cleaned_Statement": "Input", "Encoded_Status": "Label"}, inplace=True)
df3_model["Intent"] = "emotional_support"
df3_model["Response"] = ""  # No therapist response available in source dataset
```

```
# Ensure consistent column order across all datasets
df1_model = df1_model[["Input", "Intent", "Label", "Response"]]
df2_model = df2_model[["Input", "Intent", "Label", "Response"]]
df3_model = df3_model[["Input", "Intent", "Label", "Response"]]

# Combining all datasets
df_unified = pd.concat([df1_model, df2_model, df3_model], ignore_index=True)
# Export unified dataset
df_unified.to_csv("/Users/satwik/Documents/GitHub/cap5771sp25-project/Data/Modeling_Dataset.csv", index=False)
print("Unified Dataset Created ")
print("Shape:", df_unified.shape)
```

```
Unified Dataset Created
Shape: (52216, 4)
```

Fig 11: Applying Categorical Encoding on Unified Dataset

Feature Selection

a. Feature Importance Evaluation

- To determine the most relevant features for each classification task, we evaluated feature importance using a combination of exploratory data analysis (EDA) insights, correlation analysis, and model-driven evaluation techniques. In particular, the engineered features were assessed both visually and statistically to verify their individual relevance and non-redundancy.
- The correlation heatmap presented in Fig 6 confirmed that most numerical features in the unified dataset were weakly correlated with one another (correlation coefficients < 0.4), indicating that each feature captured distinct information. The only moderate correlation observed was between Statement_Word_Count and Has_Severe_Keyword (~0.41), which makes intuitive sense that longer emotional messages are more likely to contain high-severity terms.
- Further, we used model-based feature importance estimation inside the Random Forest classifiers trained during Phase 1 and Phase 2. In the **Intent Classification** task, text-based TF-IDF features were the dominant signal, but auxiliary features like Statement_Word_Count and Question_Word_Count helped boost performance for edge cases. In the **Label Classification** task (especially for emotional_support), Random Forest consistently ranked Has_Severe_Keyword and Statement_Word_Count as top predictors, validating their engineered value.
- In summary, the selected features were found to be both statistically and behaviorally meaningful. This evaluation step justified their inclusion in model training pipelines for all tasks.

b. Feature Inclusion/Exclusion and Dimensionality Reduction

Feature selection in this project followed a **task-specific approach**, where the set of features used varied based on the classification objective — specifically, whether the model was predicting the user's **Intent** or the finer-grained **Label**. This selective feature inclusion strategy was driven by the heterogeneity of our unified dataset, which originated from three separate mental health-related datasets (counseling conversations, FAQs, and sentiment analysis). Each of these sources contributed different types of features that were only applicable in certain contexts.

Intent Classification:

For the **Intent Classifier**, the primary goal was to predict the intent category of a given user input (i.e., `emotional_support`, `faq`, or `general_support`). Since the model's core input was textual, we relied on a **TF-IDF vectorized representation** of the `Input` column as the dominant feature. However, we included two auxiliary features to enhance model performance, particularly for edge cases:

- `Statement_Word_Count` – helped differentiate verbose emotional statements from short factual queries.
- `Question_Word_Count` – helped flag FAQ-style queries due to their conciseness.

Including these features allowed the model to leverage both semantic content (via TF-IDF) and structural input length cues, improving classification accuracy. Features that were only applicable post-response (e.g., `Response_Word_Count`) were excluded from this phase since they are not available at prediction time.

Label Classification:

For the **Label Classifier**, we adopted a more granular feature selection strategy, filtering the dataset by **Intent type** and choosing features specifically relevant to that intent's content format. The goal was to avoid including irrelevant or null-heavy columns and to ensure models were trained only on semantically meaningful inputs.

Feature Selection Breakdown by Intent:

- **emotional_support**
  - **Features Used**: `Input`, `Statement_Word_Count`, `Has_Severe_Keyword`
  - Justification: The presence and length of emotional text, along with severity indicators, provide rich signals for classifying mental health states such as depression or anxiety.

- **faq**
  - **Features Used**: `Input`, `Question_Word_Count`, `Contains_Mental_Health_Topic`
  - Justification: These features are tailored to factual queries. Word count captures question brevity, and keyword presence supports topic-specific classification (e.g., medication vs. recovery).
- **general_support**
  - **Features Used**: `Input` only
  - Justification: This intent had only one label (`general_support`) across the dataset, rendering label classification unnecessary. Models were not trained for this category due to the lack of label diversity.

This selective inclusion ensured that each model was trained using **only features that made sense within the context of the message type**. It also improved model interpretability, reduced noise, and minimized unnecessary computations. These feature selections will be showed in code snippets later in Data Modeling section at each models.

Data Modeling

Task 1: Intent Classification

The goal of this task was to classify user inputs into one of three intents: `emotional_support`, `faq`, or `general_support`. Since this is a multiclass classification problem, three different model architectures were trained and evaluated.

Model 1: Logistic Regression

Data Splitting Strategy: A stratified 80/20 split was used to ensure that the class imbalance (especially the underrepresented `faq` intent) was proportionally represented in both training and testing sets.

Training & Feature Selection:

- The input text was vectorized using **TF-IDF**.
- Two numerical features were appended: `Statement_Word_Count` and `Question_Word_Count`.
- These features were scaled using `StandardScaler` prior to training.

Evaluation and Results:

- The model achieved **95% accuracy**, with strong performance on the dominant `emotional_support` class.
- However, macro F1-score (~0.63) reflected reduced performance on minority classes like `faq`.
- Overall, the model performed well on the majority class but struggled with recall for `faq`, suggesting further balancing or upsampling may be required.

Model 2: Multinomial Naive Bayes

Data Splitting Strategy: The same stratified split used in Model 1 was maintained.

Training & Feature Selection:

- The model used **only the TF-IDF representation** of the input text (no numerical features included).
- This approach favored high-dimensional sparse input, which aligns with NB's assumptions.

Evaluation and Results:

- The model achieved **96% accuracy**, but macro F1-score dropped to ~0.39.
- The classifier completely failed to identify `faq` intent (0.0 F1-score), showing a **strong bias toward the dominant class**.
- While efficient and fast to train, NB was not suitable due to poor generalization on minority classes.

Model 3: Random Forest Classifier

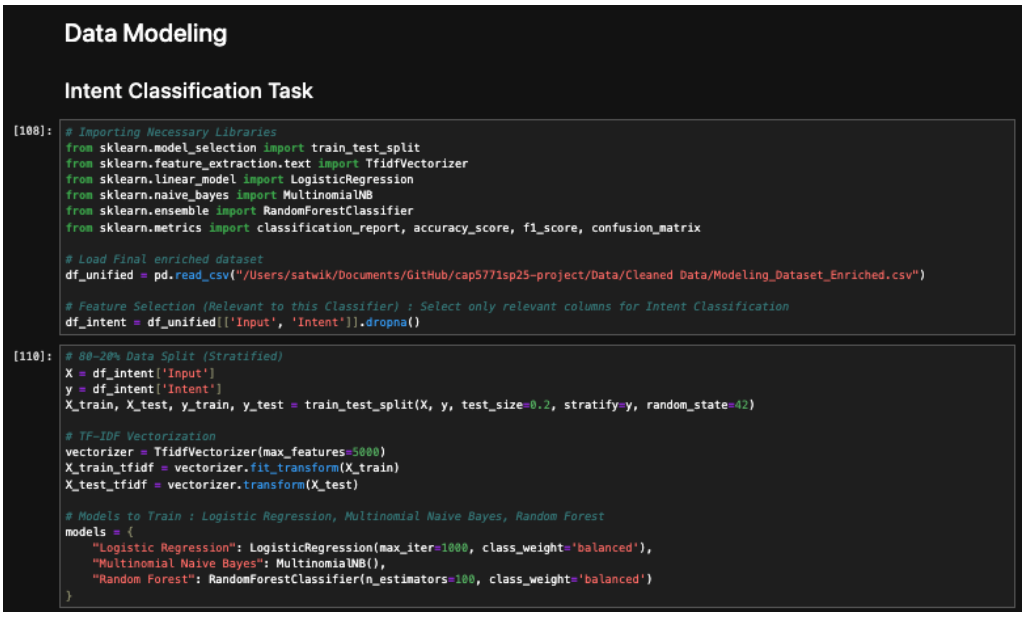
Data Splitting Strategy: Same stratified 80/20 split was used.

Training & Feature Selection:

- Combined **TF-IDF vectors** and selected numerical features (`Statement_Word_Count`, `Question_Word_Count`).
- Since Random Forests are insensitive to feature scaling, no additional transformation was needed.

Evaluation and Results:

- Achieved the **highest performance** with an accuracy of **98.5%** and macro F1-score of **0.68**.
- Generalized well across all three classes, correctly identifying most of the minority class samples.
- Demonstrated balanced performance and interpretability, making it the **best-performing model** for Intent Classification.



```
[116]: # Evaluation
results = []
for name, model in models.items():
    model.fit(X_train_tfidf, y_train)
    y_pred = model.predict(X_test_tfidf)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred, average='macro', zero_division=0)  # <- fix here
    print(f"{name} Classification Report:\n")
    print(classification_report(y_test, y_pred, zero_division=0))   # <- fix here
    results.append((name, acc, f1))

# Showing Results
df_results = pd.DataFrame(results, columns=["Model", "Accuracy", "Macro F1 Score"])
print("\nModel Performance Summary:")
print(df_results)
```

```
Logistic Regression Classification Report:

                    precision    recall  f1-score   support

emotional_support       0.99      0.95      0.97      9430
              faq       0.21      0.50      0.30        20
  general_support       0.47      0.90      0.62       396

         accuracy                           0.95      9846
        macro avg       0.56      0.79      0.63      9846
     weighted avg       0.97      0.95      0.96      9846

Multinomial Naive Bayes Classification Report:

                    precision    recall  f1-score   support

emotional_support       0.96      1.00      0.98      9430
              faq       0.00      0.00      0.00        20
  general_support       1.00      0.10      0.19       396

         accuracy                           0.96      9846
        macro avg       0.65      0.37      0.39      9846
     weighted avg       0.96      0.96      0.95      9846

Random Forest Classification Report:

                    precision    recall  f1-score   support

emotional_support       0.99      1.00      0.99      9430
              faq       0.75      0.15      0.25        20
  general_support       1.00      0.68      0.81       396

         accuracy                           0.99      9846
        macro avg       0.91      0.61      0.68      9846
     weighted avg       0.99      0.99      0.98      9846


Model Performance Summary:
                   Model  Accuracy  Macro F1 Score
0    Logistic Regression  0.950030        0.629205
1  Multinomial Naive Bayes  0.961913        0.389382
2            Random Forest  0.985375        0.684411
```

Fig 12. Intent Classification Task

Based on the evaluation results e got in Fig 12, to identify the most suitable model for predicting user intent, we trained and evaluated three classifiers: Logistic Regression, Multinomial Naive Bayes, and Random Forest. While both Logistic Regression and Naive Bayes achieved high overall accuracy, they struggled with minority classes like `faq`, yielding low macro F1-scores and recall. **Random Forest emerged as the most robust model** by achieving the highest macro F1-score (~0.68) and maintaining balanced performance across all three intent categories. Its ability to handle class imbalance, learn nonlinear relationships, and incorporate both text and numerical features made it the best choice for intent detection in this task.

Task 2: Label Classification

Label classification was conducted **independently for each intent**, due to the heterogeneous nature of labels. Only two intents (`emotional_support` and `faq`) had label variation. We trained Logistic Regression and Random Forest for each.

Model 1: Logistic Regression (Label Classifier per Intent)

Data Splitting Strategy: Separate **stratified 80/20 splits** were created for each intent category, ensuring per-label distribution was preserved.

Training & Feature Selection:

- For `emotional_support`: Used `Input` (TF-IDF), `Statement_Word_Count`, and `Has_Severe_Keyword`.
- For `faq`: Used `Input` (TF-IDF), `Question_Word_Count`, and `Contains_Mental_Health_Topic`.
- Features were scaled using StandardScaler.

Evaluation and Results:

- For `emotional_support`, macro F1-score was **~0.73** with notable improvement in precision and recall for more common labels like `0.0` and `1.0`.
- For `faq`, performance was **moderate (80% accuracy)** but less reliable due to the **small test set** (~20 records total). Minority labels had zero precision and recall.
- Still, Logistic Regression served as a good **baseline** for both contexts.

Model 2: Random Forest (Label Classifier per Intent)

Data Splitting Strategy: Same per-intent stratified split was used.

Training & Feature Selection:

- Features were kept the same as Model 4.
- Random Forest required no scaling.

Evaluation and Results:

- For `emotional_support`, performance was slightly lower than Logistic Regression in terms of F1-score (**~0.65**), particularly on rare labels like `4.0`.
- For `faq`, achieved **90% accuracy**, with better handling of underrepresented labels than LR.
- Overall, Random Forest showed more robustness in handling skewed label distributions and **performed best** for FAQ label prediction.

Label Classification Task

[137]:

```
from sklearn.utils.multiclass import unique_labels

# Loop through each intent safely
for intent, config in intent_configs.items():
    print(f"\n--- Training models for Intent: {intent} ---")
    subset = df[df['Intent'] == intent].dropna(subset=config['features'] + ['Label'])
    X_text = subset['Input']
    y = subset['Label'].astype(str)
    numeric_features = [col for col in config['features'] if col != 'Input']
    X_numeric = subset[numeric_features]

    if len(y.unique()) < 2:
        print(f"Skipping {intent} - only 1 class present.")
        continue

    # Try stratified split, fallback if error
    try:
        X_text_train, X_text_test, X_num_train, X_num_test, y_train, y_test = train_test_split(
            X_text, X_numeric, y, test_size=0.2, random_state=42, stratify=y
        )
    except ValueError:
        print("Stratified split failed. Using regular split.")
        X_text_train, X_text_test, X_num_train, X_num_test, y_train, y_test = train_test_split(
            X_text, X_numeric, y, test_size=0.2, random_state=42
        )

    tfidf = TfidfVectorizer(max_features=3000)
    X_text_train_vec = tfidf.fit_transform(X_text_train)
    X_text_test_vec = tfidf.transform(X_text_test)

    scaler = StandardScaler()
    X_num_train_scaled = scaler.fit_transform(X_num_train)
    X_num_test_scaled = scaler.transform(X_num_test)

    X_train_final = scipy.sparse.hstack([X_text_train_vec, X_num_train_scaled])
    X_test_final = scipy.sparse.hstack([X_text_test_vec, X_num_test_scaled])

    for model_name, model in models.items():
        print(f"\n Training {model_name} on {intent}")

        try:
            model.fit(X_train_final, y_train)
            y_pred = model.predict(X_test_final)

            acc = accuracy_score(y_test, y_pred)
            f1 = f1_score(y_test, y_pred, average='macro')

            print(f"Classification Report for {model_name} ({intent}):")
            print(classification_report(y_test, y_pred))

            results_label_classification.append({
                "Intent": intent,
                "Model": model_name,
                "Accuracy": acc,
                "Macro_F1": f1
            })

        except ValueError as e:
            print(f"Skipping {model_name} on {intent} due to error: {e}")
```

```
--- Training models for Intent: emotional_support ---

 Training Logistic Regression on emotional_support
Classification Report for Logistic Regression (emotional_support):
              precision    recall  f1-score   support

         0.0       0.88      0.95      0.91      3108
         1.0       0.74      0.73      0.73      3017
         2.0       0.69      0.67      0.68      2127
         3.0       0.80      0.80      0.80       720
         4.0       0.68      0.44      0.53       458

    accuracy                           0.78      9430
   macro avg       0.76      0.72      0.73      9430
weighted avg       0.77      0.78      0.78      9430


 Training Random Forest on emotional_support
Classification Report for Random Forest (emotional_support):
              precision    recall  f1-score   support

         0.0       0.87      0.94      0.90      3108
         1.0       0.64      0.80      0.71      3017
         2.0       0.69      0.51      0.59      2127
         3.0       0.84      0.69      0.76       720
         4.0       0.86      0.17      0.29       458

    accuracy                           0.74      9430
   macro avg       0.78      0.62      0.65      9430
weighted avg       0.75      0.74      0.73      9430


--- Training models for Intent: faq ---

 Training Logistic Regression on faq
Classification Report for Logistic Regression (faq):
                   precision    recall  f1-score   support

find_professional       0.00      0.00      0.00         1
      general_faq       0.88      1.00      0.93        14
  medication_info       0.00      0.00      0.00         1
    recovery_info       0.50      0.67      0.57         3
treatment_options       0.00      0.00      0.00         1

         accuracy                           0.80        20
        macro avg       0.28      0.33      0.30        20
     weighted avg       0.69      0.80      0.74        20


 Training Random Forest on faq
Classification Report for Random Forest (faq):
                   precision    recall  f1-score   support

find_professional       1.00      1.00      1.00         1
      general_faq       0.93      1.00      0.97        14
  medication_info       0.00      0.00      0.00         1
    recovery_info       0.75      1.00      0.86         3
treatment_options       0.00      0.00      0.00         1

         accuracy                           0.90        20
        macro avg       0.54      0.60      0.56        20
     weighted avg       0.82      0.90      0.85        20
```

```
=== Model Performance Summary for Label Classification ===
              Intent                Model  Accuracy  Macro_F1
0  emotional_support  Logistic Regression  0.779533  0.731776
1  emotional_support        Random Forest  0.743584  0.649719
2                faq  Logistic Regression  0.800000  0.300952
3                faq        Random Forest  0.900000  0.564532
4  emotional_support  Logistic Regression  0.779533  0.731776
5  emotional_support        Random Forest  0.743584  0.649719
6                faq  Logistic Regression  0.800000  0.300952
7                faq        Random Forest  0.900000  0.564532
```


Fig 13. Label Classification Task

Based on the code results in Fig 13, Since label meanings varied across intents, we trained separate classifiers for `emotional_support` and `faq` messages using Logistic Regression and Random Forest. For `emotional_support`, Logistic Regression slightly outperformed Random Forest in terms of macro F1-score (~0.73 vs. ~0.65), demonstrating better recall for mid-frequency sentiment classes. However, for the `faq` intent, Random Forest handled the small and imbalanced dataset more effectively, producing a higher weighted F1-score and recall for rare labels. Given these findings, the best model per intent was selected based on performance within each subgroup, ensuring tailored, high-quality predictions for each type of user input.

Chatbot Development and Deployment

The final goal of this project was to transform a series of machine learning models and NLP pipelines into a fully functional, **Interactive Mental Health Support Chatbot**. This conversational agent was deployed using the **Gradio framework**, supporting multi-turn interactions, real-time intent classification, and empathetic response generation using a distilled transformer model from Hugging Face.

Chatbot Pipeline Overview

The chatbot operates in three major stages for each user input:

1. **Intent Prediction (ML-Based):**
   - Uses a trained **Random Forest classifier** to categorize the user's message into one of the supported intent types:
     - `emotional_support`
     - `faq`
     - `general_support`
   - Input messages are transformed using a **TF-IDF vectorizer (169 features)** to ensure dimensional compatibility with the trained model.
2. **Response Generation (LLM-Based):**
   - For every user message, the chatbot leverages the **facebook/blenderbot-400M-distill** model (loaded via Hugging Face Transformers) to generate an appropriate, human-like response.
   - A **Conversation history** is maintained across multiple turns, and the response is generated using the full context of previous user-bot exchanges.
   - The model runs locally, without any need for cloud API calls, ensuring low latency and full control over inference.
3. **User Interface (Gradio UI):**
   - The chatbot is deployed using **Gradio's Blocks and Chatbot components**, creating a chat-style interface.
   - The UI includes:
     - A user message input box labeled "How are you feeling today?"
     - A **Predicted Intent** box that displays real-time results from the intent classifier.
     - A scrolling multi-turn chat window that logs the full conversation.
     - "Submit" and "Clear" buttons to facilitate continuous interaction and resetting of chat history.

Chat Memory & Personalization

To simulate a realistic conversational experience, the chatbot maintains **session-based memory**, storing all prior exchanges in the form of:

```
User: [Message]
Bot: [Response]
```

This chat history is passed as context to the transformer model at each turn, allowing the chatbot to respond based on prior context rather than isolated inputs.

Deployment Setup

- All models and vectorizers are stored in a dedicated `BestModels/` directory.
- The deployed chatbot runs entirely in a local Python environment using `gradio.launch(share=True)` for public sharing.
- No cloud deployment (e.g., Hugging Face Spaces or Streamlit Cloud) was necessary, although such extensions can be made for future scalability.

Gradio Chatbot App

```python
import gradio as gr
import pickle
import os
from transformers import BlenderbotTokenizer, BlenderbotForConditionalGeneration
from sklearn.feature_extraction.text import TfidfVectorizer

# Load Models
model_dir = "/Users/satwik/Documents/GitHub/cap5771sp25-project/BestModels"

with open(os.path.join(model_dir, "intent_classifier_random_forest.pkl"), "rb") as f:
    intent_model = pickle.load(f)

with open(os.path.join(model_dir, "vectorizer_intent.pkl"), "rb") as f:
    intent_vectorizer = pickle.load(f)

#  Load BlenderBot
blenderbot_name = "facebook/blenderbot-400M-distill"
tokenizer = BlenderbotTokenizer.from_pretrained(blenderbot_name)
blenderbot = BlenderbotForConditionalGeneration.from_pretrained(blenderbot_name)

# Chat History
chat_history = []

# Response Logic
def chatbot_reply(user_message, history):
    global chat_history

    # Step 1: Intent Prediction
    X_vec = intent_vectorizer.transform([user_message])
    predicted_intent = intent_model.predict(X_vec)[0]

    # Step 2: Add User Message
    chat_history.append(("User", user_message))

    # Step 3: Build LLM Context
    full_context = ""
    for speaker, message in chat_history:
        full_context += f"{speaker}: {message}\n"

    # Step 4: Generate LLM Response
    inputs = tokenizer(full_context, return_tensors="pt", truncation=True, max_length=128)
    reply_ids = blenderbot.generate(**inputs)
    response = tokenizer.decode(reply_ids[0], skip_special_tokens=True)

    # Step 5: Add Bot Response
    chat_history.append(("Bot", response))

    # Format for gr.Chatbot
    formatted_chat = []
    for speaker, message in chat_history:
        role = "user" if speaker == "User" else "assistant"
        formatted_chat.append({"role": role, "content": message})
    return formatted_chat, predicted_intent
```

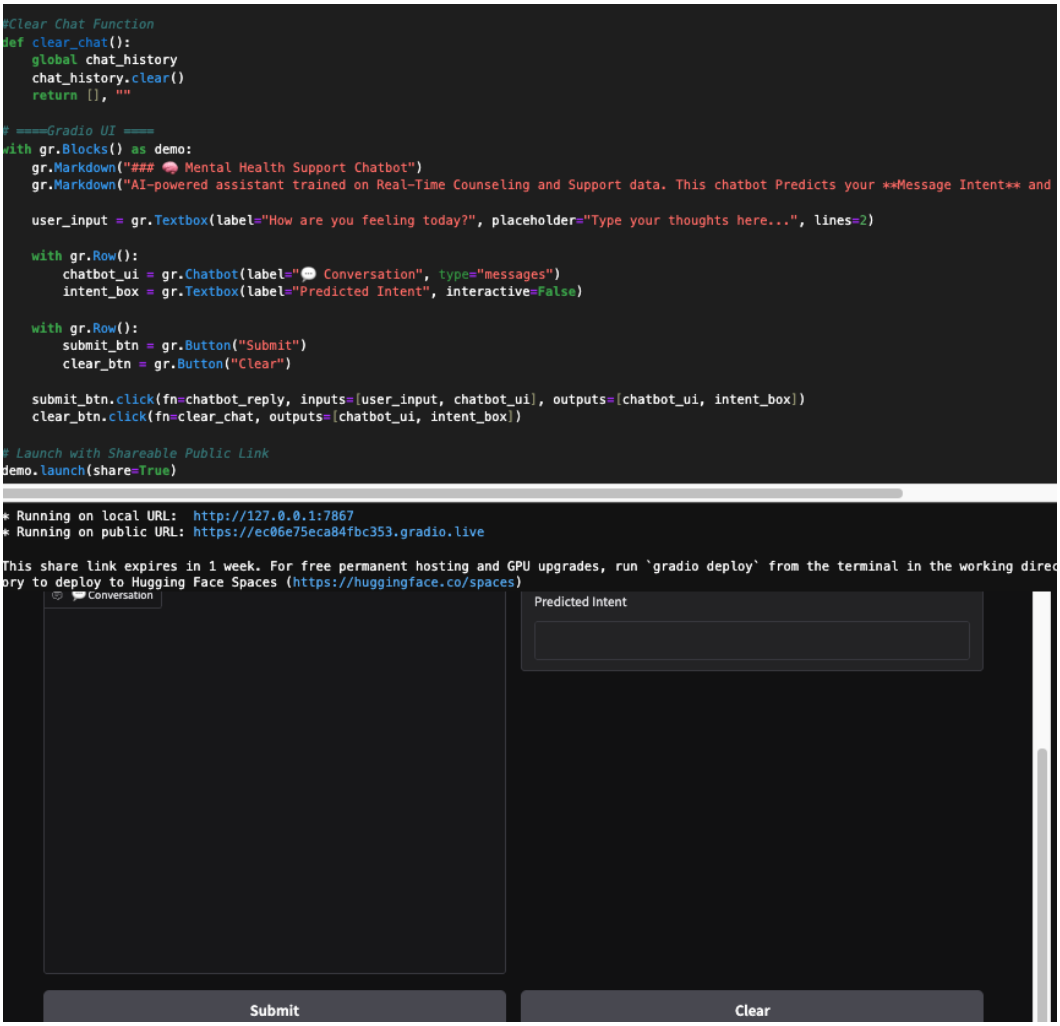


Fig 14. Chatbot Building Code

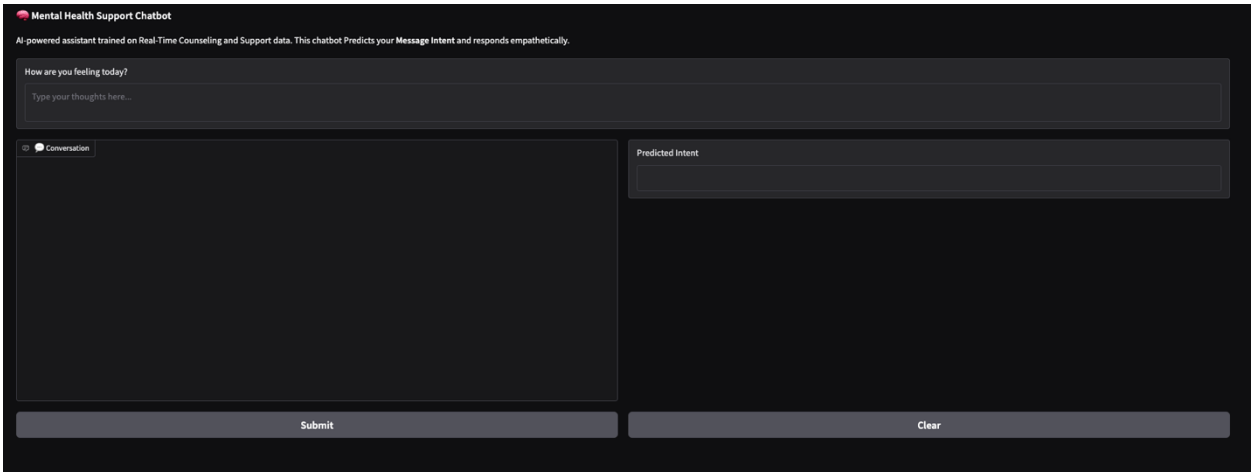


Fig 15. Final Gradio Chatbot UI

Conclusion

This project successfully demonstrates the design and development of an AI-powered **Mental Health Support Chatbot** using a combination of **machine learning** and **transformer-based NLP models**. By unifying three diverse and complementary datasets related to emotional counseling, mental health FAQs, and sentiment-labeled statements, we built a robust backend system capable of understanding user intent and generating empathetic responses.

Throughout the project, we followed the **CRISP-DM methodology**, progressing from data collection and preprocessing to exploratory analysis, feature engineering, model training, and final deployment. The most impactful innovation in this chatbot lies in its hybrid architecture—where **intent classification is handled by supervised ML**, and **response generation is powered by a lightweight, locally deployed LLM** (facebook/blenderbot-400M-distill). This architecture ensures a scalable, real-time, and human-like conversational experience.

While the label classification module was explored during earlier phases, it was excluded from the final deployment due to compatibility challenges. This simplification allowed us to focus on building a stable and responsive tool.

The final chatbot interface was developed using **Gradio**, delivering a user-friendly, multi-turn chat experience with real-time **intent display**, maintaining conversational memory across user interactions. This makes the tool highly usable for end-users seeking emotional support, basic mental health information, or general therapeutic conversation.

In future iterations, the system can be extended with:

- Larger-scale transformer models (e.g., Blenderbot-3B or GPT-based APIs)
- Fine-tuning on domain-specific therapy dialogues
- Integration with mental health crisis detection and escalation systems

Overall, the deployed chatbot demonstrates the power of combining **ML interpretability** with **NLP fluency** in the service of mental health—a domain where compassionate, context-aware interaction can have a meaningful impact