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

1. 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 techniques to provide empathetic, intent-aware responses to user messages. This AI-powered conversational agent is designed to serve as a virtual companion for individuals seeking emotional support**,** especially in moments of distress or uncertainty.

- The deployed chatbot is capable of:
  - **Classifying the user's intent** (e.g., emotional support, general inquiry) using a trained Random Forest classifier on the Unified dataset I formed using three different datasets.
  - **Generating Empathetic, human-like responses** using the facebook/blenderbot-400M-distill model which is a lightweight yet effective Large Language Model (LLM)**.**

The chatbot was built using a **Unified modeling dataset** created by integrating and preprocessing three complementary mental health-related sources including Counseling conversations offering real-world emotional support interactions, Frequently asked questions (FAQs) on mental health topics and Sentiment-labeled mental health statements from social platforms.

This project followed the CRISP-DM methodology which included the Steps like Data Collection and Understanding**,** Preprocessing and Feature Engineering**,** Model Training and Evaluation and Final Tool Development and Deployment.

By the end of this milestone, the chatbot was successfully deployed with a **Gradio-based Multi-turn conversational interface**, providing real-time user interaction with predicted intent awareness and language generation capabilities.

2. 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 outcome is an intelligent virtual assistant capable of understanding user intent and generating empathetic, context-aware responses in real time.

The tool developed is a **Multi-turn Conversational agent**, deployed using Gradio, designed to simulate natural human dialogue while interpreting the emotional or informational needs of users. It combines the robustness of classical machine learning models for intent classification with the generative fluency of a transformer-based LLM, enabling meaningful and supportive interactions.

The final deployed chatbot is built around two core components:

- **Intent Classification:** A Random Forest classifier, trained on a unified dataset, predicts the intent behind each user message distinguishing between emotional support, general conversation, and FAQ-style informational queries.
- **Response Generation:** The facebook/blenderbot-400M-distill model is used **to** generate responses based on the full conversational context, providing coherent and emotionally attuned replies that align with the predicted intent.

The main focus of this milestone was to integrate these components into a seamless multi-turn conversational UI experience, where each user message is interpreted, logged into session history, and responded to via an LLM with conversational memory. Unlike traditional static bots, this tool adapts dynamically to user tone and predicted intent, providing a more humane, supportive, and contextually relevant experience particularly crucial in mental health interactions.

3. Data Used & Licensing

This project utilizes **Three** carefully selected datasets to train a unified, intent-aware chatbot capable of handling emotional support, factual mental health queries, and general counseling-style conversations. These datasets were preprocessed, merged, and enriched with engineered features to create a Modeling pipeline for Intent classification. All datasets used are **publicly accessible and licensed for academic use**, in compliance with Project guidelines.

The three datasets are either in the public domain: Creative Commons licenses like **CC0** and **CC BY** and is allowed for academic, non-commercial research. For code developed during this project, standard open-source licenses (such as MIT License or Apache 2.0 License) may be applied if the project were to be open-sourced in the future. All dataset references comply with UF's guidelines for dataset citation and license transparency.

❖ 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 includes realistic question-answer pairs from online therapy platforms, representing user mental health concerns and therapist responses.

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 Integrated under the `general_support` intent. These entries enhance the chatbot's ability to provide general mental health assistance.

❖ 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 focused set of frequently asked questions and brief, informative responses related to symptoms, treatments, and general awareness.

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:
It is 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.

4. Tech Stack

The development of this AI-powered mental health chatbot spanned the complete CRISP-DM lifecycle, encompassing data collection, preprocessing, modeling, evaluation, and deployment. The following tools and frameworks were utilized throughout the project:

1. Programming Language

- **Python**: Used extensively for data preprocessing, model building, evaluation, and chatbot deployment.

2. Data Preprocessing and Analysis

- **Pandas, NumPy**: For data loading, cleaning, integration, and statistical summarization across multiple datasets.
- **NLTK, SpaCy**: Employed for text preprocessing tasks including tokenization, stopword removal, and lemmatization.
- **re (Regular Expressions)**: Utilized for text normalization, pattern extraction, and keyword-based feature engineering.

3. Data Visualization and Exploratory Data Analysis (EDA)

- **Matplotlib, Seaborn**: Used to generate insightful visualizations such as class distributions, feature histograms, and correlation matrices.
- **WordCloud**: Applied for visual representation of frequent words within emotional and counseling text corpora.

4. Feature Engineering and Selection

- **Scikit-learn**:
  - TF-IDF Vectorizer for text feature extraction.
  - Manual feature creation techniques, including engineered variables such as word counts and presence of critical mental health keywords.

5. Machine Learning Models

- **Scikit-learn**:
  - **Random Forest Classifier** for Intent Prediction (trained on unified dataset).
  - Model selection focused on achieving a balance between performance and compatibility with text vectorization.

6. Text Vectorization

- **TF-IDF Vectorization (Scikit-learn)** is applied to transform user input text into numerical vectors. Feature limits were explicitly controlled to match the expectations of the corresponding ML models.

7. LLM-based Chatbot Response Generation

- **Hugging Face Transformers**:
  - **Model**: `facebook/blenderbot-400M-distill`, a lightweight distilled conversational model used for generating empathetic responses.
  - **Tokenizer**: `BlenderbotTokenizer`, responsible for input preparation and sequence management.
- The LLM model was hosted and run locally, avoiding reliance on external APIs and ensuring faster real-time interaction.

8. Chatbot UI Development and Deployment

- **Gradio**: This Chatbot is Implemented using **Gradio Blocks** and **gr.Chatbot** components.
  - Designed to support **multi-turn conversations** with **session memory**, predicted intent display, and a clean, responsive user interface.

9. Version Control and Code Management

- **Git and GitHub**:
  - A private GitHub repository was maintained for version control, project organization, and milestone submissions.

10. Reporting and Presentation

- **Microsoft Word**: Used for documentation and formatting the Milestone reports.
- **Microsoft PowerPoint**: Utilized to prepare the project presentation for Final submission.

5. Exploratory Data Analysis (EDA) Report

- The purpose of Exploratory Data Analysis (EDA) in this project was to extract actionable insights from the unified modeling dataset to guide feature engineering and model selection for the deployed chatbot.
- The primary goals of EDA were :

  a. To validate the informativeness and distributions of engineered features.
  b. To identify potential challenges in intent class imbalance.
  c. To confirm correlations and eliminate redundant or irrelevant variables.

Key Plots & Visualizations:

❖ Distribution of Statement Word Count (Intent: emotional_support)

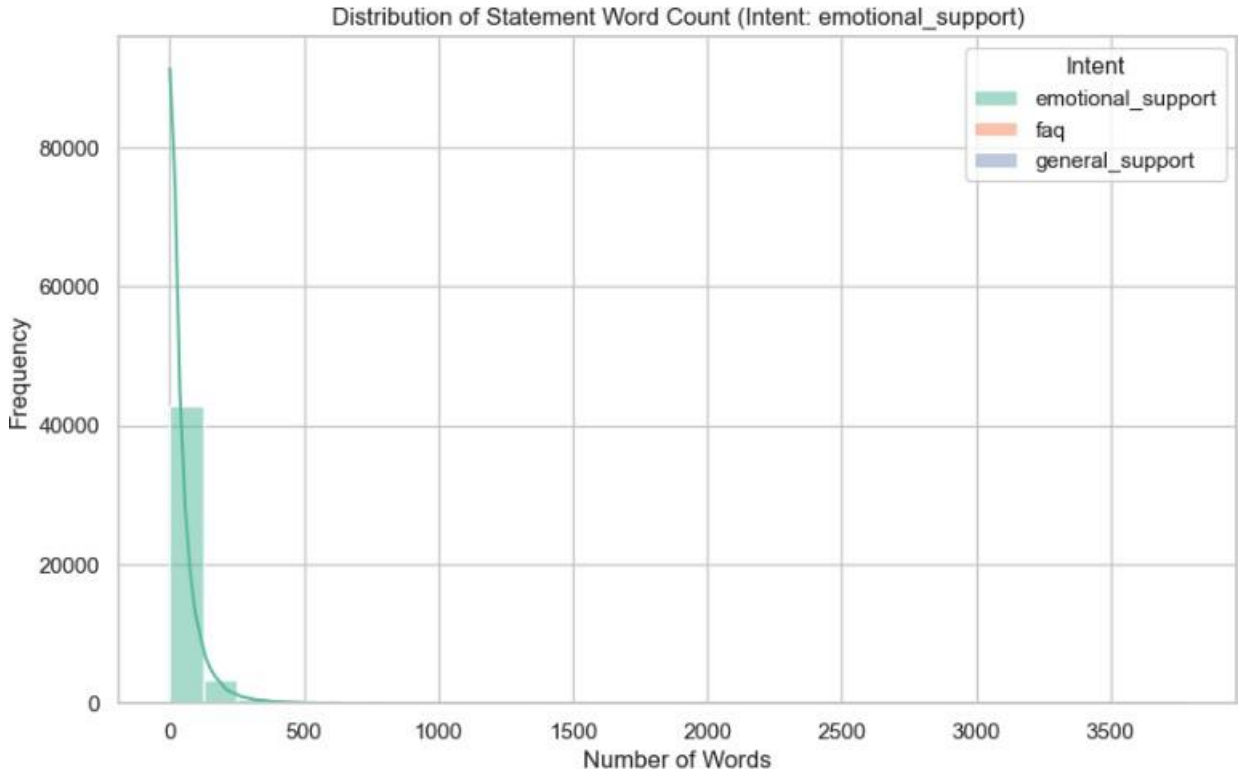


Fig 1: Distribution of Statement Word Count (Intent: emotional_support)

Fig 1 shows a highly right-skewed distribution of `Statement_Word_Count`, with most emotional support messages falling under 50 words. However, a significant long-tail pattern emerges, with some statements extending past 1000 words. This variability highlights how users in emotional distress may express themselves with drastically different verbosity levels.

- **EDA Insight**: This distribution indicates the need for scalable preprocessing, and `Statement_Word_Count` was included as a feature to capture input depth for Intent prediction.
- **Modeling Relevance**: This feature helped the Intent Classifier distinguish emotional_support messages from shorter, factual queries.

❖ Distribution of Question Word Count (Intent: faq)

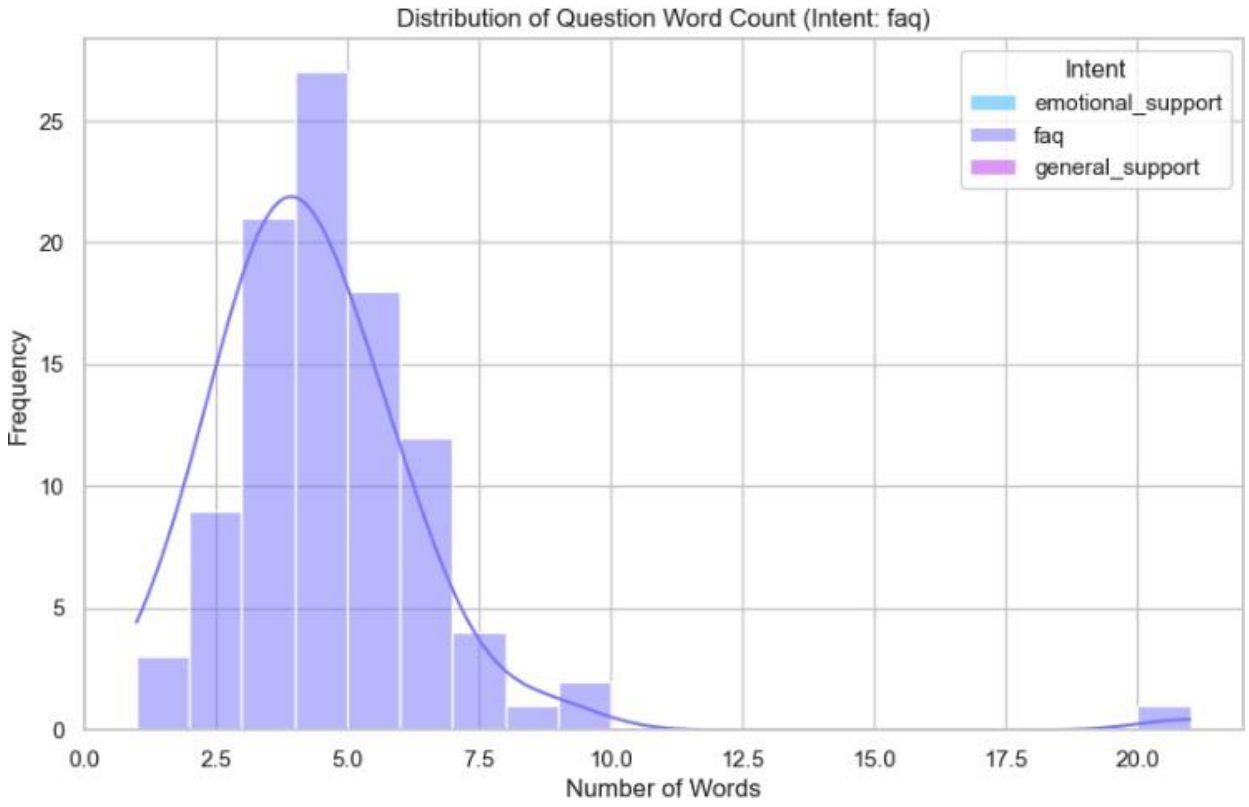


Fig 2: Distribution of Question Word Count (Intent: faq)

In **Fig 2** highlights the concise nature of FAQ-style inputs, with a tight left-skew centered around 3–6 words. This is consistent with questions like “Can anxiety be cured?” or “Is therapy safe?”

- **EDA Insight**: The engineered `Question_Word_Count` feature cleanly separates FAQ intents from others.
- **Modeling Relevance**: This feature is directly used by the Intent Classifier to capture structural differences between factual and conversational queries.

❖ Distribution of Response Word Count (Intent: general_support)

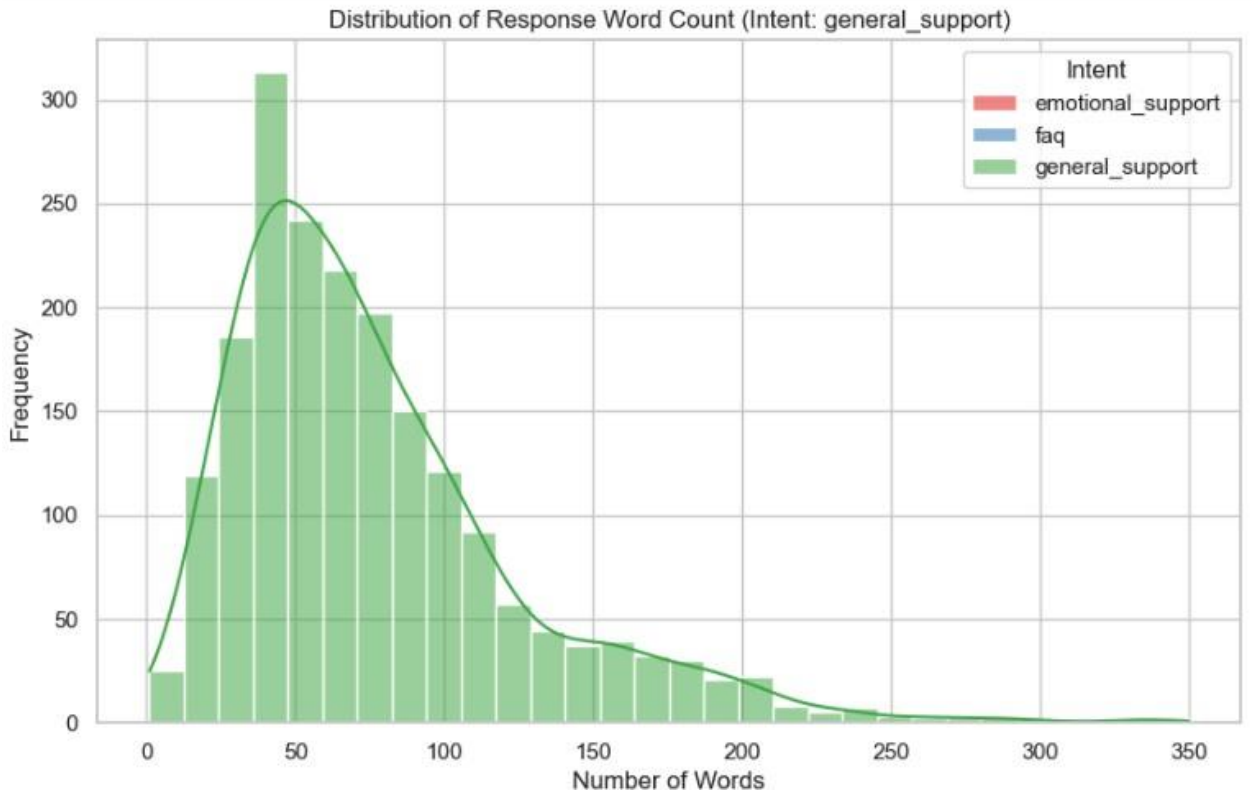


Fig 3: Distribution of Response Word Count (Intent: general_support)

In **Fig 3**, response lengths for general_support inputs mostly fall between 40–150 words, with a moderate long tail. This reflects how supportive messages often provide guidance or encouragement in a structured, medium-length format.

- **EDA Insight**: The `Response_Word_Count` feature offered insight into therapist-style response formatting.
- **Modeling Relevance**: Although not used in training the intent model, it helped inform response tone and completeness expectations for **LLM-based generation**.

❖ Presence of Severe Keywords in Emotional Support Inputs:

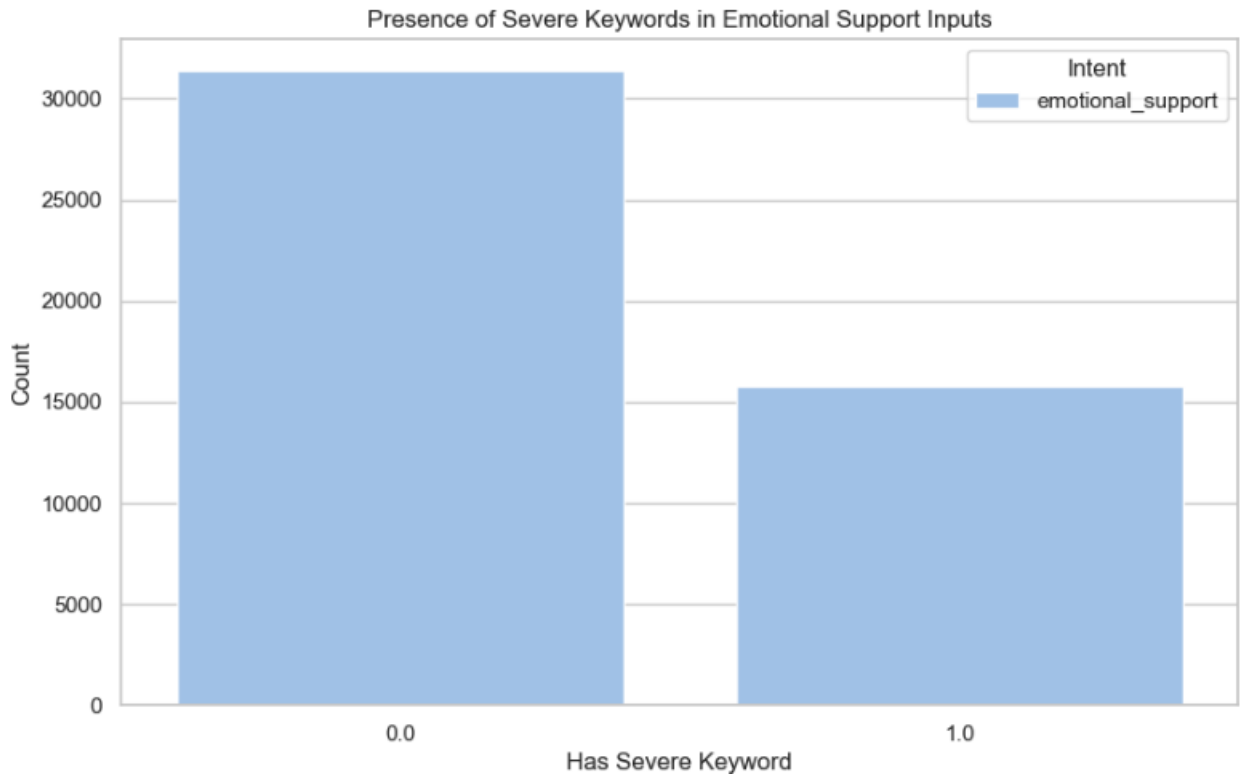


Fig 4: Presence of Severe Keywords in Emotional Support Inputs

Fig 4 analyzes the binary feature `Has_Severe_Keyword`, showing that approximately 16,000 emotional_support inputs contain phrases like "hopeless," "suicidal," or "worthless," while over 30,000 do not.

- **EDA Insight**: This feature provides a proxy for emotional severity, which is critical for triaging sensitive user input.
- **Modeling Relevance**: `Has_Severe_Keyword` was a key feature in **intent prediction**, improving the model's sensitivity to emotionally intense input.

❖ Correlation Heatmap of Numerical Features

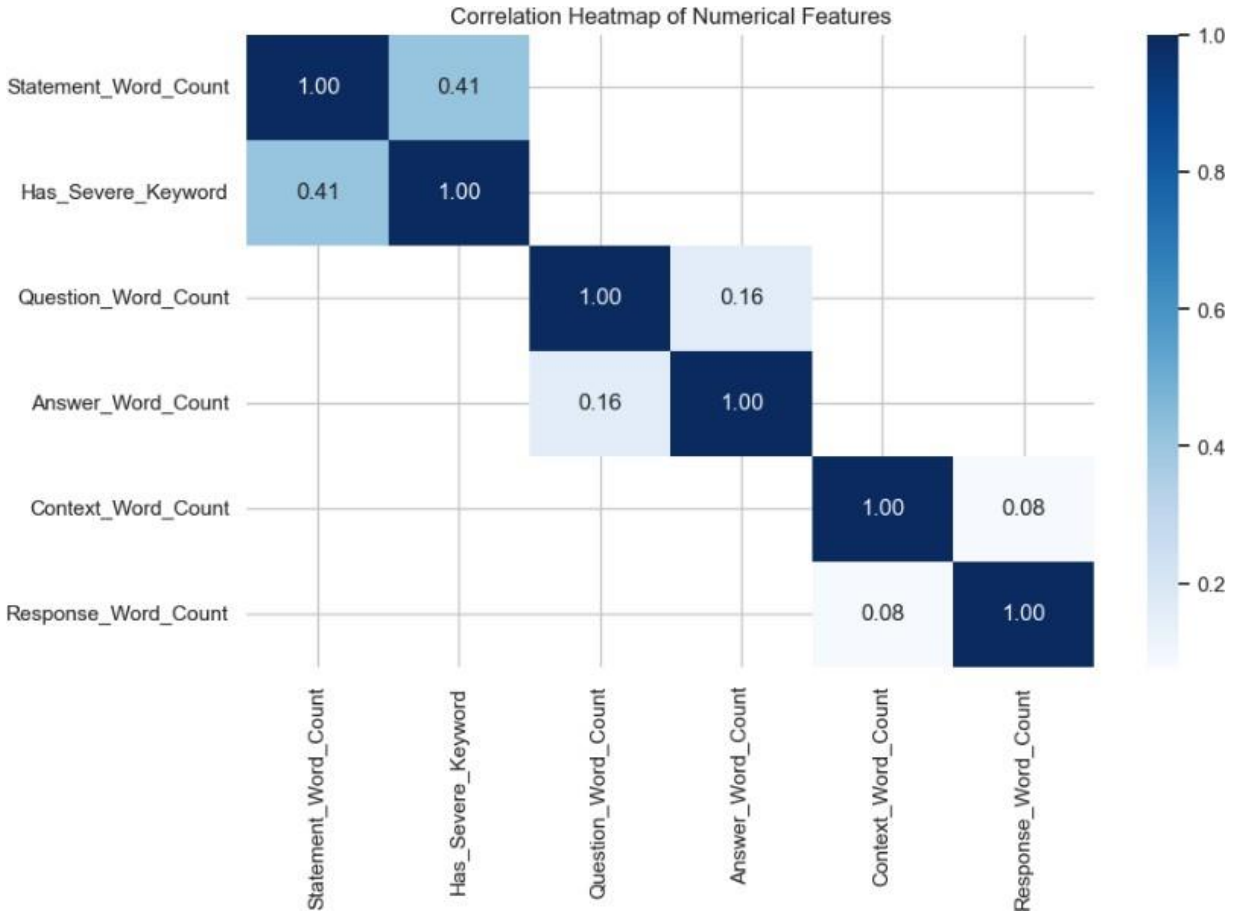


Fig 5: Correlation Heatmap of Numerical Features

Fig 6 shows Pearson correlations among all engineered numeric features. The strongest observed link is between `Statement_Word_Count` and `Has_Severe_Keyword` (correlation ~0.41), reinforcing the intuition that emotionally intense messages are often longer. Other correlations are weak or non-redundant.

- **EDA Insight**: The features are largely independent, reducing the risk of multicollinearity.
- **Modeling Relevance**: All numeric features were retained for **Intent Classification** without the need for dimensionality reduction (e.g., PCA)

6. Feature Engineering

Feature engineering was a critical step in transforming those multi-source mental health datasets into a unified modeling dataset that enabled accurate Intent classification. Since each dataset (counseling conversations, FAQs, and sentiment-labeled posts) had distinct structures and content types, features were first engineered independently within each dataset. These features were then consolidated to build a consistent and semantically enriched final dataset for training the chatbot's intent classifier.

- **Sentiment Analysis Dataset (Emotional Support Domain)**

From the sentiment-labeled dataset, we engineered two key features:

- **`Statement_Word_Count`**: It captures the number of words in a user's input text. As shown in **Fig 6**, this involved normalizing the input statements using basic NLP preprocessing (lowercasing, stopword removal, lemmatization) and then computing word length.
- **`Has_Severe_Keyword`**: It is a binary indicator representing the presence of critical keywords (e.g., "suicidal", "hopeless", "worthless"). This was created to help the classifier flag emotionally intense messages.

These features helped the model capture emotional granularity in user input under the `emotional_support` intent.

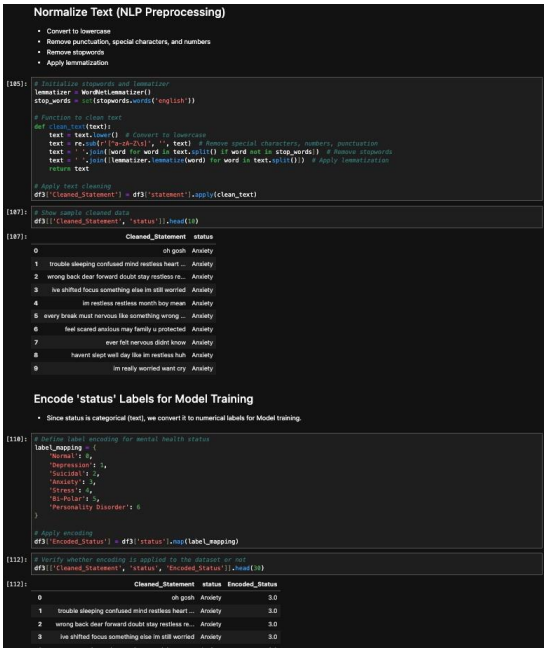


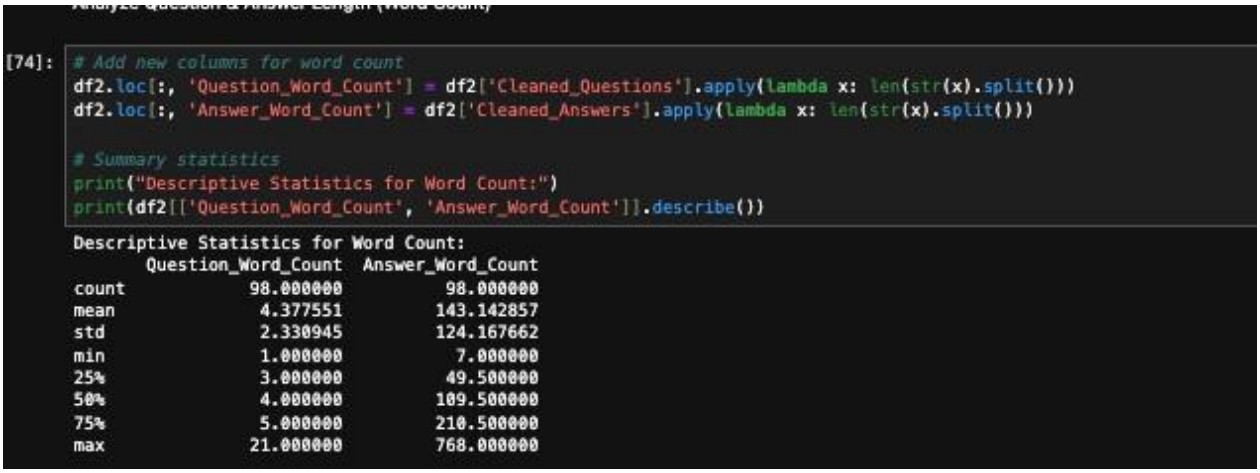
Fig 6: Feature Engineering on Sentiment Analysis Dataset

- **FAQ Dataset (Informational Domain)**

For FAQ-style queries, brevity and structure were leveraged using:

- **`Question_Word_Count`**: Measures the number of words in the FAQ query.
- **`Answer_Word_Count`**: Measures the length of the response provided.

As demonstrated in **Fig 7**, these statistics were useful for distinguishing question-based inputs from conversational messages and assisted in shaping intent classifier features.



```
[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 7: Feature Engineering on FAQ Dataset

Counseling Conversations Dataset (General Support Domain)

This dataset contributed structural conversational features:

- **`Context_Word_Count`**: Word count of user input in a therapy context.
- **`Response_Word_Count`**: Therapist's response length.

As shown in **Fig 8**, these features provide insight into conversational complexity. Although they were not directly used in the deployed model, they played a role in understanding response structure during the dataset unification 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 8: Feature Engineering on Counselling Dataset

Unified Dataset Creation & Feature Consolidation

All individually engineered features were merged into a single **Unified Modeling Dataset**. The consolidated dataset had a shape of (49,228, 12) and included cleaned user messages, their intent label, corresponding responses, and all engineered numerical features. See **Fig 9** for a snapshot of this final unified modelling dataset.

Only a selected subset of features was used in the final Intent Classification model:

- **TF-IDF vectorized input message** (169 features)
- **Statement_Word_Count** (if emotional input)
- **Has_Severe_Keyword** (if emotional input)
- **Question_Word_Count** (if FAQ-like input)

This modular setup ensured that features were semantically consistent with the predicted intent and avoided unnecessary leakage or noise.

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 9: 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. 9)
- To enable ML modeling, categorical variables were encoded:

    - **Intent Encoding**: The `Intent` column had three classes: `emotional_support`, `faq`, and `general_support`. Label encoding was applied internally during training but class names were retained in final outputs for interpretability. (Fig 10)

Encoding was handled using scoped functions and per-intent filtering to prevent cross-contamination. This approach preserved the semantics of each source during classification training.

```
#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 10: Applying Categorical Encoding on Unified Dataset

7. Feature Selection

Feature selection played a vital role in optimizing the performance and generalization of the chatbot's Intent Classification model. Given the variety of features engineered from multiple datasets, this step ensured that only the most relevant, non-redundant, and task-appropriate features were included in the final ML pipeline.

❖ Feature Importance Evaluation

To determine the most relevant features for Intent classification, we used a combination of:

- **Exploratory Data Analysis (EDA)** helped visualize distribution patterns and feature ranges across different intents (e.g., Statement Word Count for emotional inputs vs. Question Word Count for FAQ inputs).
- **Correlation Analysis** as seen in the correlation heatmap (Fig 5), most features were weakly correlated with one another, indicating minimal redundancy. The only moderate correlation was observed between `Statement_Word_Count` and `Has_Severe_Keyword` (~0.41), which is expected since longer messages are more likely to contain emotionally intense phrases.
- **Model-based Evaluation**: During training of the Random Forest Intent Classifier, we observed that TF-IDF features from the user input dominated the model's decision-making. However, auxiliary features such as `Statement_Word_Count` and `Question_Word_Count` added measurable performance gains in edge cases, such as short FAQ queries or long emotional rants.

The combination of these evaluation techniques confirmed that the selected features were both statistically independent and behaviorally meaningful.

❖ Feature Inclusion/Exclusion and Dimensionality Reduction

- In the final deployment, the chatbot's **Intent Classification model** required thoughtful feature selection to ensure it could make accurate predictions using only the information available at runtime. The core task was to identify whether a user's message reflected **emotional_support**, **faq**, or **general_support** intent.

The primary feature used was the **TF-IDF vectorized representation** of the user's `Input` text. This provided a dense semantic signal capturing the vocabulary and phrasing of user messages. In addition to this, two engineered features were selectively included to boost performance on specific edge cases:

- `Statement_Word_Count` was included to help the model distinguish longer, emotionally expressive messages that are characteristic of the **emotional_support** intent.
- `Question_Word_Count` was helpful for identifying **faq** queries, which tend to be brief and direct.

These auxiliary features complemented the TF-IDF vectors by capturing **structural patterns** in the text (e.g., verbosity, conciseness) that may not be fully reflected through token frequency alone.

- No **dimensionality reduction** (e.g., PCA) was applied to the feature set. This decision was based on the results of the correlation heatmap (see Fig 5), which confirmed that the numerical features were largely **non-redundant** and exhibited low multicollinearity. Furthermore, Random Forest classifiers are inherently robust to high-dimensional and sparse input spaces like TF-IDF and perform internal feature selection as part of their ensemble structure. As such, reducing dimensionality would not have added benefit and might have sacrificed interpretability.

This selective inclusion strategy allowed the model to remain lightweight, interpretable, and fully compatible with the multi-turn conversational interface deployed in productio

8. Data Modeling

8.1. Intent Classification Model

The objective of this task was to classify each user message into one of three mental health-related **Intent categories**: **`emotional_support`**, **`faq`**, or **`general_support`**. This classification step is critical for guiding the chatbot's response behavior and was implemented as a Multiclass Supervised learning task**.**

The full pipeline for this task is illustrated in **Fig 11**.

Data Preparation

- The final modeling dataset (`Modeling_Dataset_Enriched.csv`) was loaded and filtered to include only two columns: `Input` (text data) and `Intent` (target).
- An 80/20 train-test split was applied with stratification, ensuring all three intent classes were proportionally represented in both training and test sets.

Feature Engineering & Vectorization

- The `Input` text was transformed into numerical features using a **TF-IDF Vectorizer** (`max_features = 5000`).
- Unlike Milestone 2 (Refer Milestone2.pdf), no manual numeric features were appended at this stage to ensure consistency across models and minimize model-specific bias.
- The resulting TF-IDF matrices served as the input features (`X_train_tfidf`, `X_test_tfidf`) for model training.

Model Training

Three classical ML models were trained on the same dataset and evaluated:

1. **Logistic Regression**
   - Implemented using `LogisticRegression` from Scikit-learn library with `class_weight='balanced'` to address class imbalance.
   - Configured with `max_iter=1000` to ensure convergence on the high-dimensional TF-IDF feature space.
   - Trained on the TF-IDF vectorized training data (`X_train_tfidf`, `y_train`).
2. **Multinomial Naive Bayes**
   - Implemented using `MultinomialNB`, a probabilistic classifier well-suited for sparse feature spaces like TF-IDF.
   - No additional feature scaling or hyperparameter tuning was applied, as Naive Bayes models assume independence and naturally perform well on high-dimensional text.

3. **Random Forest Classifier**
    - Implemented using `RandomForestClassifier` with 100 trees (`n_estimators=100`) and `class_weight='balanced'` to reduce majority class bias.
    - Trained directly on TF-IDF vectors, leveraging its insensitivity to feature scaling and ability to handle sparse, high-dimensional input.

Each model was trained and evaluated independently using the same preprocessing and train-test setup to ensure a fair comparison.

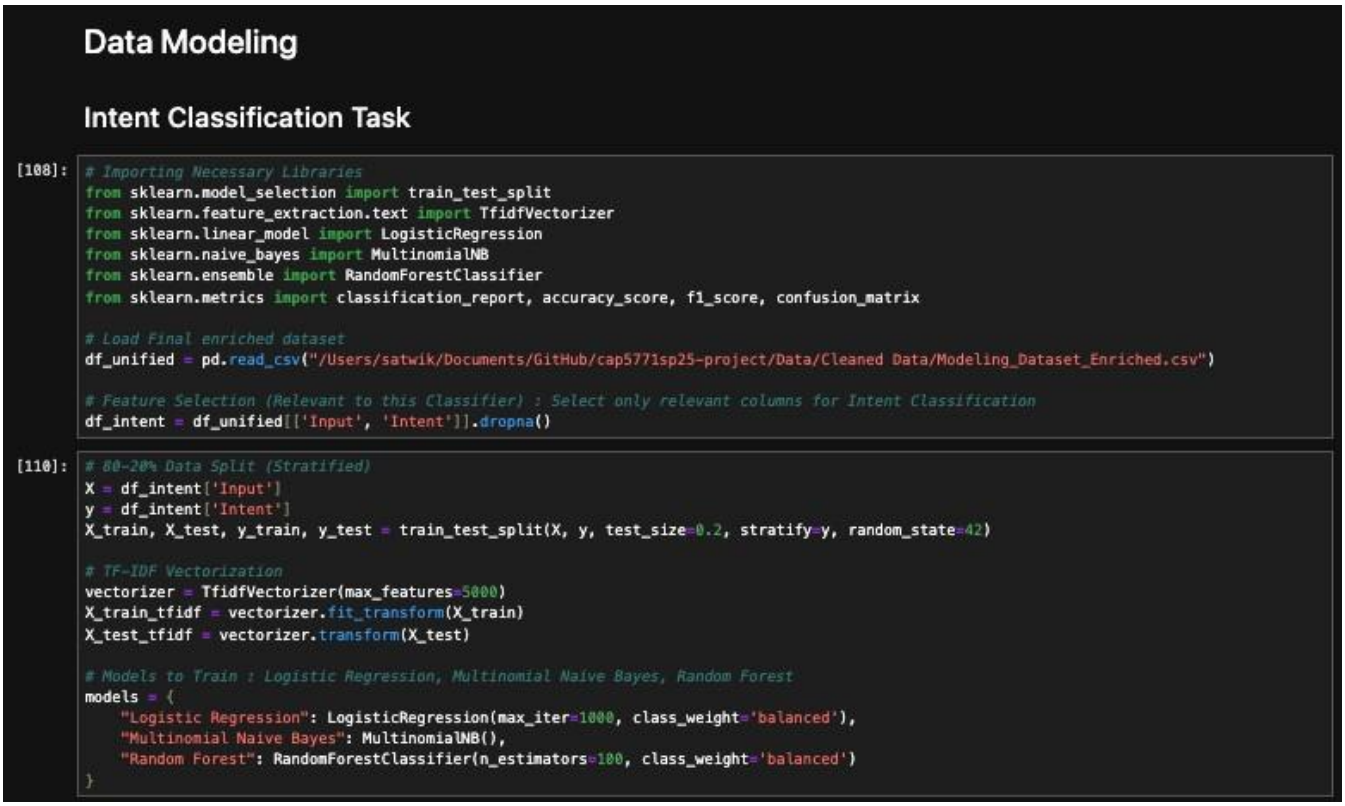


Fig 11. Intent Classification Modelling (3 Models Trained)

8.2. Model Evaluation and Selection

As shown in **Fig 12**, all three models were evaluated using a consistent test pipeline, reporting metrics per intent class and overall model performance:

Logistic Regression

- **Accuracy**: 95.0%
- **Macro F1-Score**: 0.63
- This has Strong performance on the dominant **emotional_support** class (F1 = 0.97), but **faq** class recall was low (F1 = 0.26).
- Despite using TF-IDF and class balancing, performance skewed heavily toward the majority class.

Multinomial Naive Bayes

- **Accuracy**: 96.2%
- **Macro F1-Score**: 0.39
- It failed to generalize well on minority classes. **faq** was not detected at all (F1 = 0.00).
- This model proved unsuitable for our use case due to its strong bias and low recall for non-dominant intents.

Random Forest Classifier

- **Accuracy**: **98.6%**
- **Macro F1-Score**: **0.73** (highest among all models)
- It showed balanced performance across all three classes, especially improving recall for **faq** (F1 = 0.38) and **general_support** (F1 = 0.81).
- It Demonstrated robustness to imbalanced data and was ultimately selected as 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 Modelling Evaluation

Based on this comparative evaluation, **Random Forest Classifier** was selected for deployment due to its ability to deliver the highest macro-average F1-score and its balanced treatment of all intent classes. It was later serialized and integrated into the final chatbot pipeline for real-time user intent prediction.

9. Chatbot Development and Deployment

9.1. Chatbot Development

- The final stage of the project involved bringing together the trained intent classifier and the conversational response generator into a seamless deployment pipeline. As shown in **Fig 13**, we first load the following components:

  - `intent_classifier_random_forest.pkl`: A trained Random Forest model for predicting user intent.
  - `vectorizer_intent.pkl`: A TF-IDF vectorizer used to transform input messages during inference.

```python
#Importing Necessary Libraries
import os
import pandas as pd
import pickle
from scipy.sparse import hstack
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier

# Load Datasets
df = pd.read_csv("/Users/satwik/Documents/GitHub/cap5771sp25-project/Data/Cleaned Data/Modeling_Dataset_Enriched.csv")

# Create folder to save best models
model_dir = "/Users/satwik/Documents/GitHub/cap5771sp25-project/BestModels"
os.makedirs(model_dir, exist_ok=True)

# Intent Classification
df_intent = df[['Input', 'Intent']].dropna()
X = df_intent['Input']
y = df_intent['Intent']

vectorizer_intent = TfidfVectorizer(max_features=169)  # Matching feature size
X_vec = vectorizer_intent.fit_transform(X)

X_train, X_test, y_train, y_test = train_test_split(X_vec, y, test_size=0.2, stratify=y, random_state=42)

intent_model = RandomForestClassifier(n_estimators=100, class_weight='balanced')
intent_model.fit(X_train, y_train)

with open(f"{model_dir}/intent_classifier_random_forest.pkl", "wb") as f:
    pickle.dump(intent_model, f)

with open(f"{model_dir}/vectorizer_intent.pkl", "wb") as f:
    pickle.dump(vectorizer_intent, f)
```

Fig 13. Loading Best Models

- For generating responses, we use facebook/blenderbot-400M-distill, a compact and efficient LLM from Hugging Face, loaded locally. This model was selected due to its:

    - Lightweight architecture suitable for CPU deployment.
    - Zero reliance on APIs or paid external services.
    - Ability to produce coherent, empathetic multi-turn responses.

Other alternatives like DialoGPT and ChatGPT API were considered, but they either lacked contextual grounding or required cloud access with cost or rate limits.

- The chatbot logic, illustrated in **Fig 14**, follows this pipeline:

    1. It Predicts intent from user input using TF-IDF + Random Forest.
    2. Store user message in chat history.
    3. Only the **Latest user message** is passed to the LLM for response generation.
    4. The bot's reply is decoded and returned to the chat.

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 (FIXED: Only recent user message passed)
    full_context = f"User: {user_message}"

    # 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
```

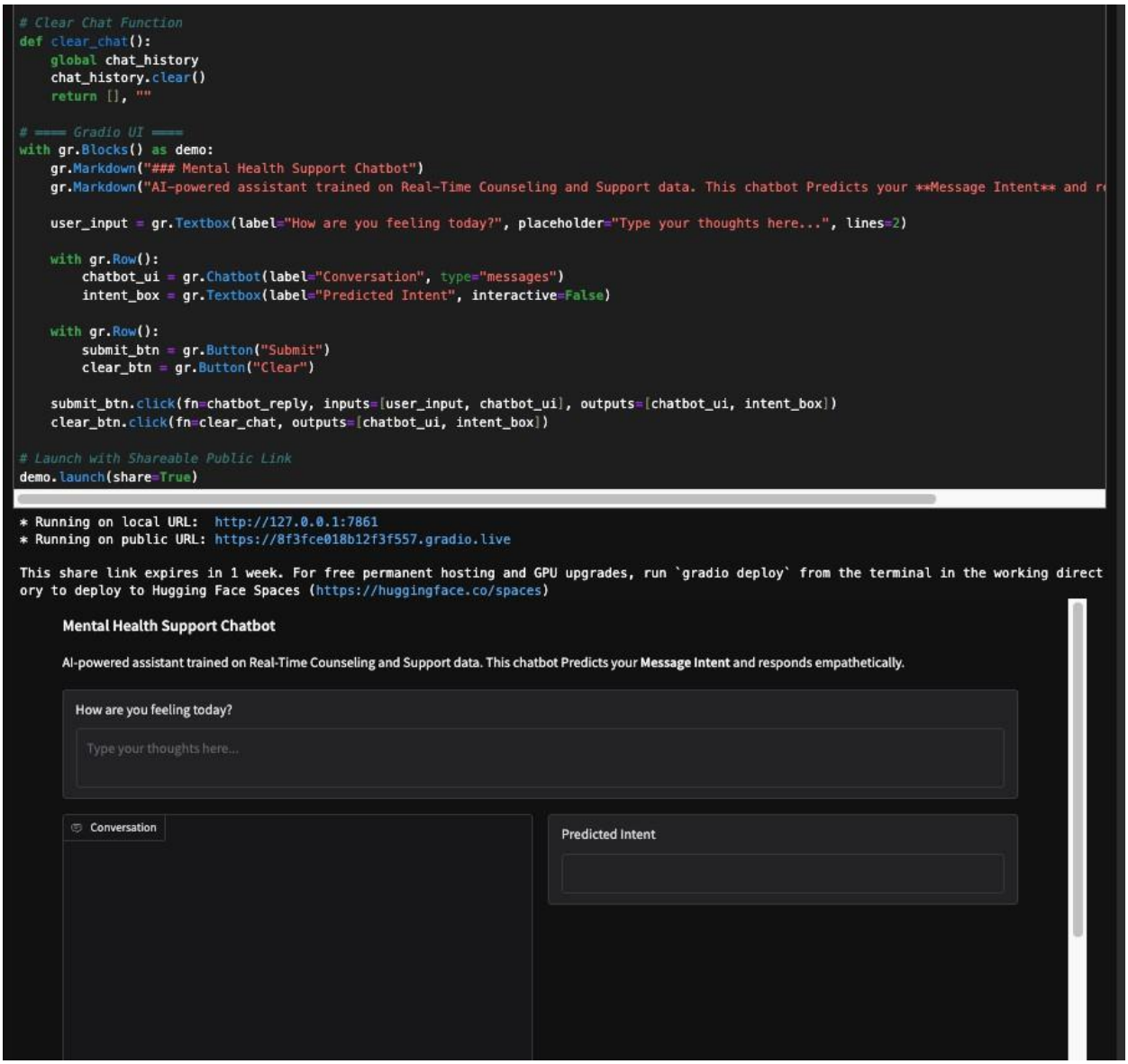


Fig 14. Chatbot Development Code

This design ensures fast response, modular expansion, and eliminates earlier issues where unwanted historical repetition was being passed to the LLM context.

9.2. Gradio UI Features

The chatbot is deployed using Gradio Blocks, creating an intuitive and responsive user interface, as seen in **Fig 15**.

Key features of the Gradio UI:

- **Multi-turn Chatbot Window** (`gr.Chatbot`) with role-based formatting for user and assistant messages.
- **Textbox Input Field** for user queries with a welcoming prompt.
- **Predicted Intent Display** (`gr.Textbox`) that helps interpret the classification logic.
- **Submit and Clear Buttons** to send messages or reset the conversation.
- **Public Sharing** enabled using `demo.launch(share=True)`, making the tool accessible via browser without local setup.

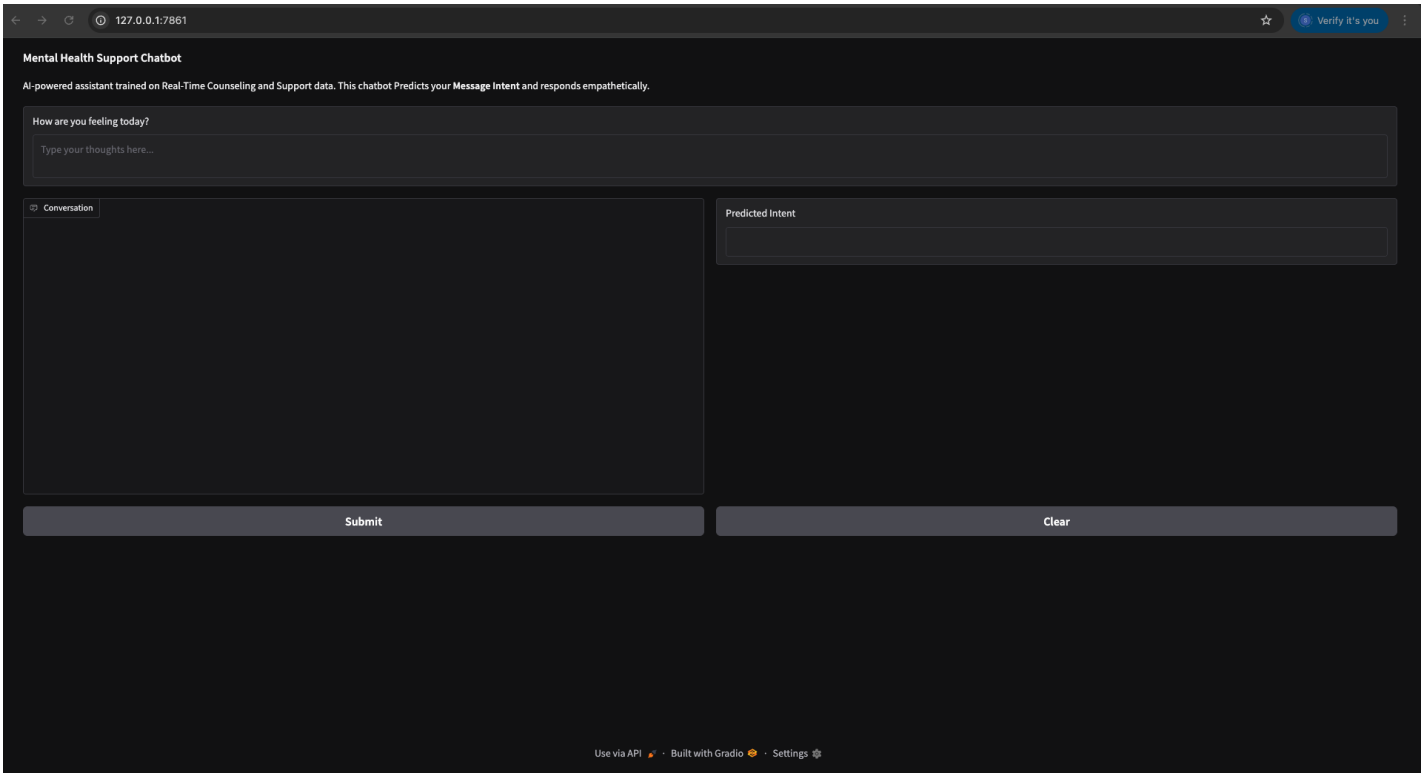


Fig 15. Chatbot Gradio UI

The UI delivers a clean, minimal experience with seamless user interaction and feedback ideal for mental health scenarios where clarity and empathy are essential.

❖ You can view the **Demo of the Tool i**n this Link: **Tool Demo Video**

10. Limitations and Future Work

❖ Limitations:

While the deployed mental health chatbot demonstrates solid performance in intent detection and response generation, it has a few notable limitations:

- **Restricted Conversational Input Style**: Unlike advanced conversational agents such as ChatGPT, Claude, or Gemini, this chatbot does not allow free-form, paragraph-style input with embedded instructions or follow-up context. Users must provide simple, direct messages per turn, which may reduce naturalness in real conversations.
- **Response Generation is Stateless**: Currently, the chatbot generates replies using only the latest user message. Although this avoids repeated context issues, it also limits the ability of the LLM to maintain meaningful multi-turn memory or track evolving emotional context.
- **Limited Customization of Responses**: Since the LLM operates generatively, responses may sometimes lack domain specificity or repeat generic supportive statements. Rule-based fallback responses or curated prompts could improve this in future iterations.

❖ Future Work:

Several enhancements can improve the quality, adaptability, and real-world usability of the chatbot as a part of future developments of the existing project:

- **Integrate Dynamic Context Windowing**: Allow the chatbot to selectively retain relevant past interactions in the LLM context to enable more natural conversations without exceeding token limits.
- **Hybrid Response Strategy**: Combine generative models with retrieval-based techniques using vector databases (e.g., FAISS or ChromaDB) to deliver more factual, intent-specific answers especially for FAQ queries.
- **Voice and Multi-modal Interface**: Extend the Gradio interface to accept audio input or even sentiment from facial expressions to offer more empathetic and accessible interactions.
- **Intent Confidence Thresholding**: Use intent prediction confidence scores to either request clarification from the user or route the message to a human moderator in sensitive contexts.

11. Conclusion

This project successfully demonstrated the end-to-end development and deployment of a **Mental Health Support Chatbot** using a unified dataset derived from counseling conversations, mental health FAQs, and sentiment-labeled user statements. By applying the CRISP-DM methodology, we were able to transition from raw data to a functioning, interactive conversational agent capable of intent detection and empathetic response generation.

Through thoughtful data preprocessing**,** feature engineering, and model training, we built an **Intent Classification pipeline** that achieved high accuracy across multiple user intents. The integration of a lightweight LLM, **Blenderbot-400M-distill**, enabled the chatbot to generate contextually appropriate and emotionally sensitive responses without relying on cloud-based APIs offering a cost-effective, locally deployable solution.

The final product, deployed via **Gradio**, provides users with an intuitive interface to share their concerns and receive real-time, supportive feedback. While the system has limitations in terms of memory and input style compared to commercial LLMs, it represents a robust and scalable foundation for future improvements.

This project not only delivers a technically sound AI-powered assistant but also highlights **How Data science can be applied meaningfully in socially impactful domains**, like mental health. With further enhancements and deployment at scale, tools like this can play a vital role in democratizing access to mental health support one empathetic response at a time.

These are the links where you can access my complete Project folder including GitHub Link (Private Repository Collaborator Invitation already sent) and Onedrive Project folder link

GitHub Link (Private Repository) : https://github.com/satwik77-dev/cap5771sp25-project

OneDrive Project Folder Link : cap5771sp25-project

12. References

1. A. Amod, "Mental Health Counseling Conversations," Hugging Face Datasets. [Online]. Available: https://huggingface.co/datasets/Amod/mental_health_counseling_conversations
2. N. Geek, "Mental Health FAQ for Chatbot," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/narendrageek/mental-health-faq-for-chatbot
3. S. Sarkar, "Sentiment Analysis for Mental Health," Kaggle. [Online]. Available: https://www.kaggle.com/datasets/suchintikasarkar/sentiment-analysis-for-mental-health
4. F. Pedregosa et al., "Scikit-learn: Machine Learning in Python," Journal of Machine Learning Research, vol. 12, pp. 2825–2830, 2011. [Online]. Available: https://scikit-learn.org/
5. T. Wolf et al., "Transformers: State-of-the-Art Natural Language Processing," in Proceedings of the 2020 EMNLP: System Demonstrations, pp. 38–45. [Online]. Available: https://huggingface.co/transformers/
6. Facebook AI Research (FAIR), "facebook/blenderbot-400M-distill," Hugging Face Models. [Online]. Available: https://huggingface.co/facebook/blenderbot-400M-distill
7. Gradio, "Gradio: Build Machine Learning Web Apps in Python," [Online]. Available: https://www.gradio.app/
8. The Pandas Development Team, "Pandas Documentation," [Online]. Available: https://pandas.pydata.org/
9. C. R. Harris et al., "Array programming with NumPy," Nature, vol. 585, pp. 357–362, 2020. [Online]. Available: https://numpy.org/
10. J. D. Hunter, "Matplotlib: A 2D Graphics Environment," Computing in Science & Engineering, vol. 9, no. 3, pp. 90–95, 2007. [Online]. Available: https://matplotlib.org/
11. M. Waskom, "Seaborn: Statistical Data Visualization," Journal of Open Source Software, vol. 6, no. 60, p. 3021, 2021. [Online]. Available: https://seaborn.pydata.org/