Chatbot for Mental Health

Domain: Health Care

Name: Leela Krishna Sai M.

Abstract

This study presents a Python-based mental health chatbot developed using existing datasets and deployed via Streamlit. Leveraging natural language processing and deep learning, the chatbot offers personalized support to users. Initial evaluations demonstrate the chatbot's capability in identifying mental health concerns and fostering user engagement. However, ongoing challenges include assessing long-term impact and addressing biases within the training data. Future research directions entail refining emotional intelligence, integrating professional mental health services, and prioritizing user data privacy. Through this approach, the chatbot represents a promising tool for accessible and effective mental health support.

Introduction

Introducing our Python-based mental health chatbot deployed on Streamlit, this innovative solution harnesses existing datasets and cutting-edge Natural Language Processing (NLP) techniques to provide personalized support. Seamlessly integrating empathetic dialogue and sophisticated linguistic analysis, our chatbot offers a secure space for users to express their emotions confidentially. By tailoring responses guided by NLP algorithms, it fosters a profound understanding of users' mental states, enabling timely interventions. Dedicated to enhancing mental health accessibility, our chatbot promotes proactive well-being practices, contributing to destigmatizing mental health conversations in a compassionate digital environment.

Need of study

- 1. Addressing mental health needs through technological innovation.
- 2. Utilizing existing datasets to enhance mental health support.
- 3. Deployment of a Python chatbot on Streamlit for accessibility.
- 4. Empowering users to articulate emotions in a secure digital space.
- 5. Contributing to destigmatizing mental health conversations through proactive intervention.

Problem Statement

1. Existing mental health support often lacks accessibility and personalization.

2. Limited availability of resources for proactive mental health management.

3. Challenges in articulating emotions and seeking timely intervention.

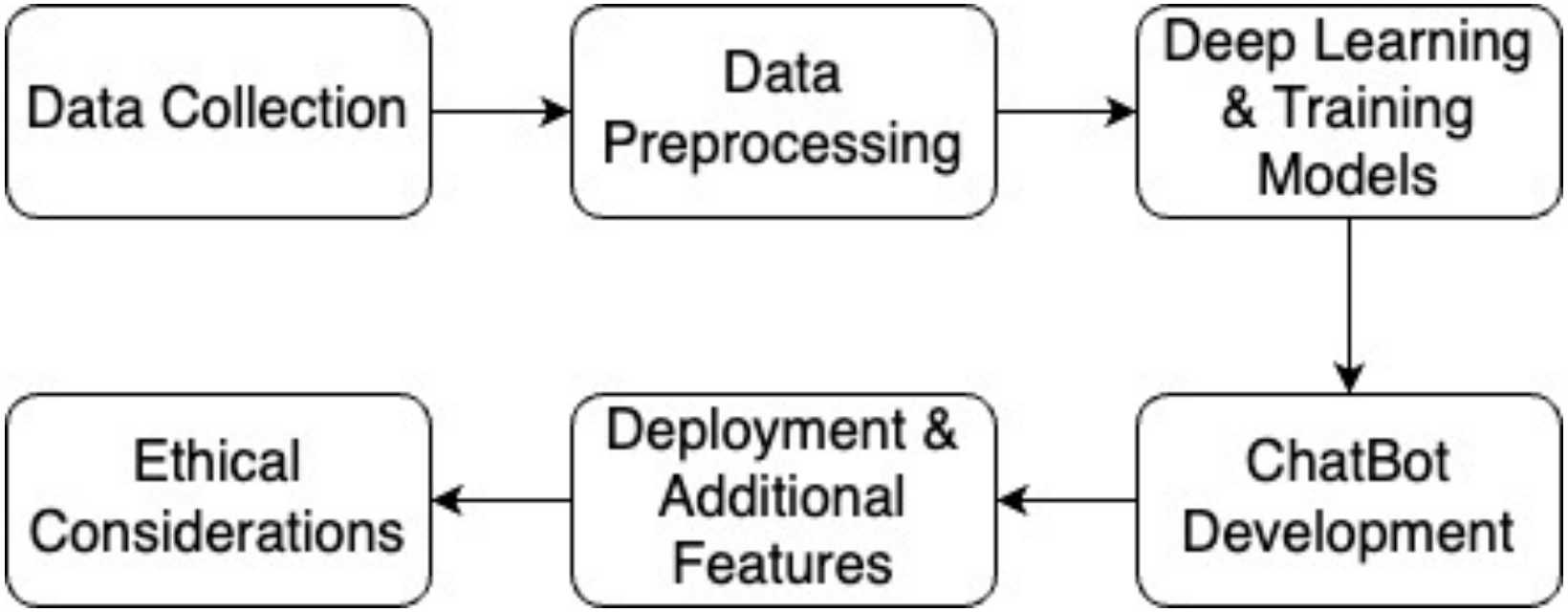
4. Inadequate utilization of existing datasets for tailored support solutions.

5. Need for a technologically advanced and user-friendly mental health support system.

Objectives

- Develop a robust chatbot interface leveraging deep learning techniques to provide accessible mental health support.
- Enhance user engagement and interaction by implementing natural language processing algorithms for empathetic responses.
- Integrate sentiment analysis to gauge user emotions and tailor responses for personalized support.
- Evaluate the effectiveness of the chatbot through user feedback and assessment of mental health outcomes.

Proposed Methodology



Dataset

- Source of the Dataset : https://www.kaggle.com/datasets/elvis23/mental-health-conversational-data/data
- No. of Observations : 80 rows & 3 columns
- Column/Feature Details : The ‘tag’ identifies intent, ’patterns’ define triggers, and ‘responses’are pre-written replies for a chatbot conversation system.
- Details about the columns – categorical

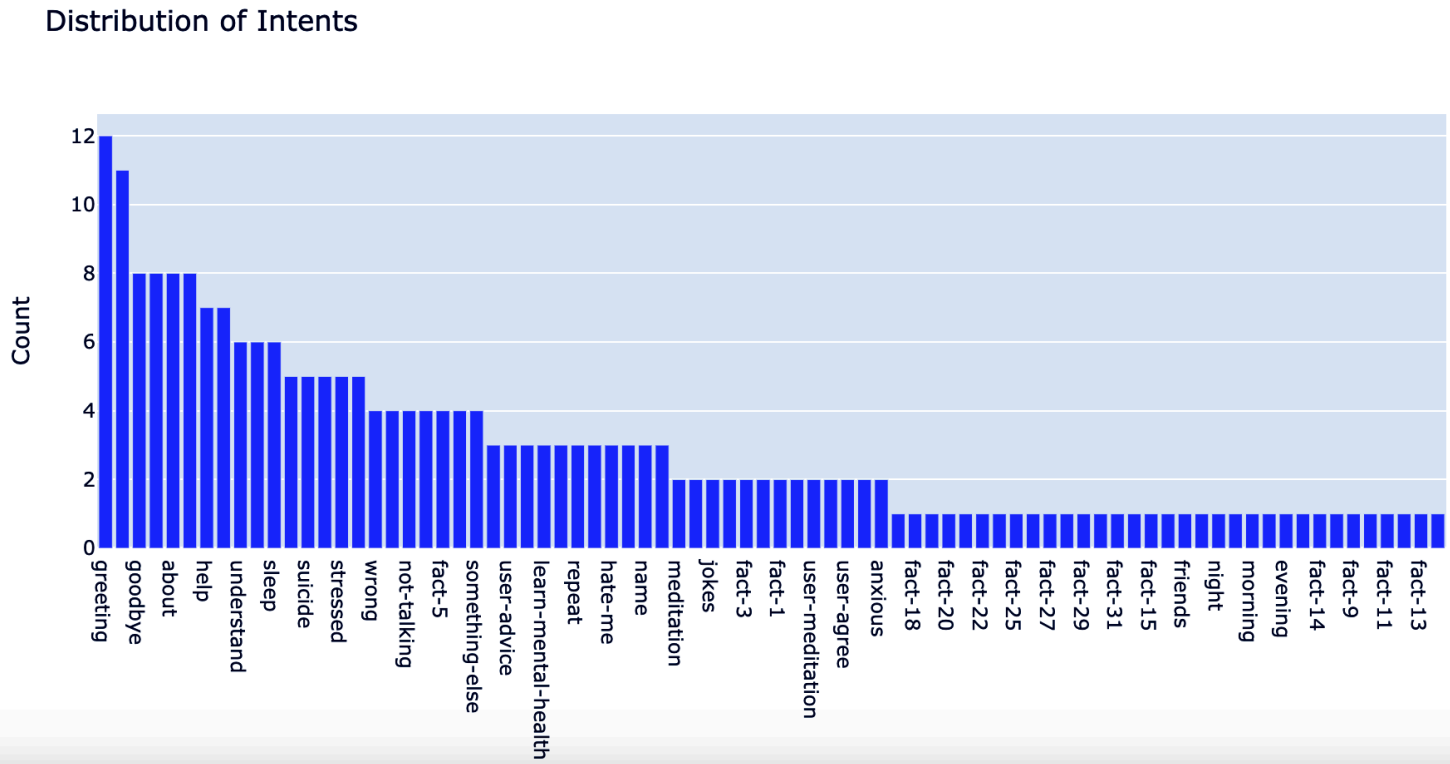
- Screenshot of the Dataset :

	tag	patterns	responses
0	greeting	[Hi, Hey, Is anyone there?, Hi there, Hello, H...	[Hello there. Tell me how are you feeling toda...
1	morning	[Good morning]	[Good morning. I hope you had a good night's s...
2	afternoon	[Good afternoon]	[Good afternoon. How is your day going?]
3	evening	[Good evening]	[Good evening. How has your day been?]
4	night	[Good night]	[Good night. Get some proper sleep, Good night...
...	...	...	...
75	fact-28	[What do I do if I'm worried about my mental h...	[The most important thing is to talk to someon...
76	fact-29	[How do I know if I'm unwell?]	[If your beliefs , thoughts , feelings or beha...
77	fact-30	[How can I maintain social connections? What i...	[A lot of people are alone right now, but we d...
78	fact-31	[What's the difference between anxiety and str...	[Stress and anxiety are often used interchange...
79	fact-32	[What's the difference between sadness and dep...	[Sadness is a normal reaction to a loss, disap...

80 rows × 3 columns

Exploratory Data Analysis

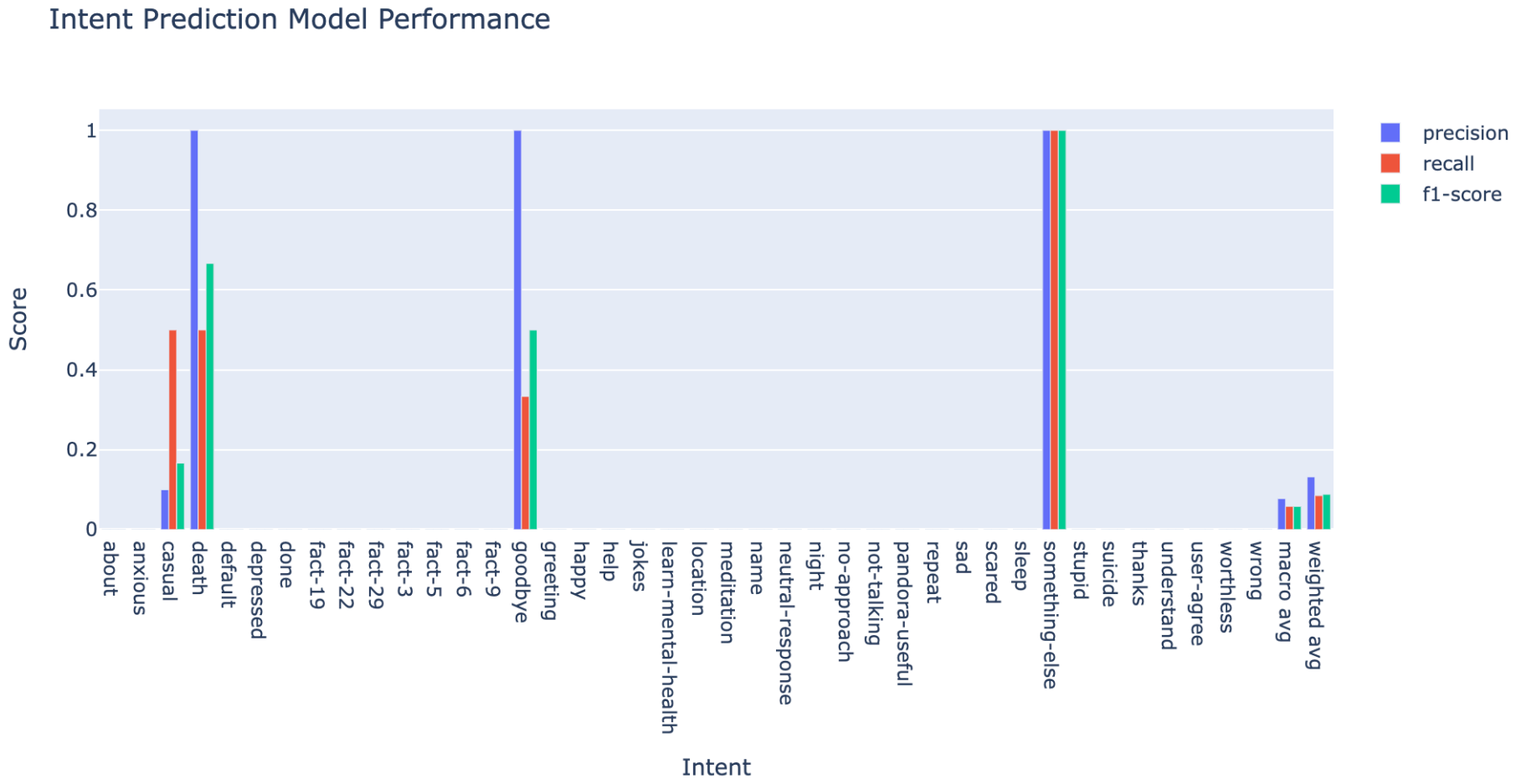
In [12]:
```
import plotly.graph_objects as go

intent_counts = df['tag'].value_counts()
fig = go.Figure(data=[go.Bar(x=intent_counts.index, y=intent_counts.values)])
fig.update_layout(title='Distribution of Intents', xaxis_title='Intents', yaxis_title='Count')
fig.show()
```



Performance Analysis

```
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.svm import SVC
from sklearn.metrics import classification_report
import plotly.graph_objects as go
```

```
# Split the dataset into training and testing sets
X = df['patterns']
y = df['tag']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Vectorize the text data using TF-IDF
vectorizer = TfidfVectorizer()
X_train_vec = vectorizer.fit_transform(X_train)
X_test_vec = vectorizer.transform(X_test)

# Train a Support Vector Machine (SVM) classifier
model = SVC()
model.fit(X_train_vec, y_train)

# Predict intents for the testing set
y_pred = model.predict(X_test_vec)

# Evaluate the model's performance
report = classification_report(y_test, y_pred, output_dict=True, zero_division=0)

# Convert float values in the report to dictionaries
report = {label: {metric: report[label][metric] for metric in report[label]} for label in report if isinstance(repor

# Extract evaluation metrics
labels = list(report.keys())
evaluation_metrics = ['precision', 'recall', 'f1-score']
metric_scores = {metric: [report[label][metric] for label in labels if label in report] for metric in evaluation_met

# Visualize the model's performance using a Plotly bar plot
fig = go.Figure()
for metric in evaluation_metrics:
    fig.add_trace(go.Bar(name=metric, x=labels, y=metric_scores[metric]))

fig.update_layout(title='Intent Prediction Model Performance',
                  xaxis_title='Intent',
                  yaxis_title='Score',
                  barmode='group')

fig.show()
```


Intent Prediction Model Performance
Score
1
0.8
0.6
0.4
0.2
0
about
anxious
casual
death
default
depressed
done
fact-19
fact-22
fact-29
fact-3
fact-5
fact-6
fact-9
goodbye
greeting
happy
help
jokes
learn-mental-health
location
meditation
name
neutral-response
night
no-approach
not-talking
pandora-useful
repeat
sad
scared
sleep
something-else
stupid
suicide
thanks
understand
user-agree
worthless
wrong
macro avg
weighted avg
Intent
precision
recall
f1-score

Results and Disussion

- Gathering the Data - We have gathered the data from kaggle.
- Cleaning the Data –

  Number of missing values in Before Cleaning: 0

  Number of missing values after cleaning: 0

  Because The entire Dataset is in Variables
- We have created the following machine learning classification models SVM.

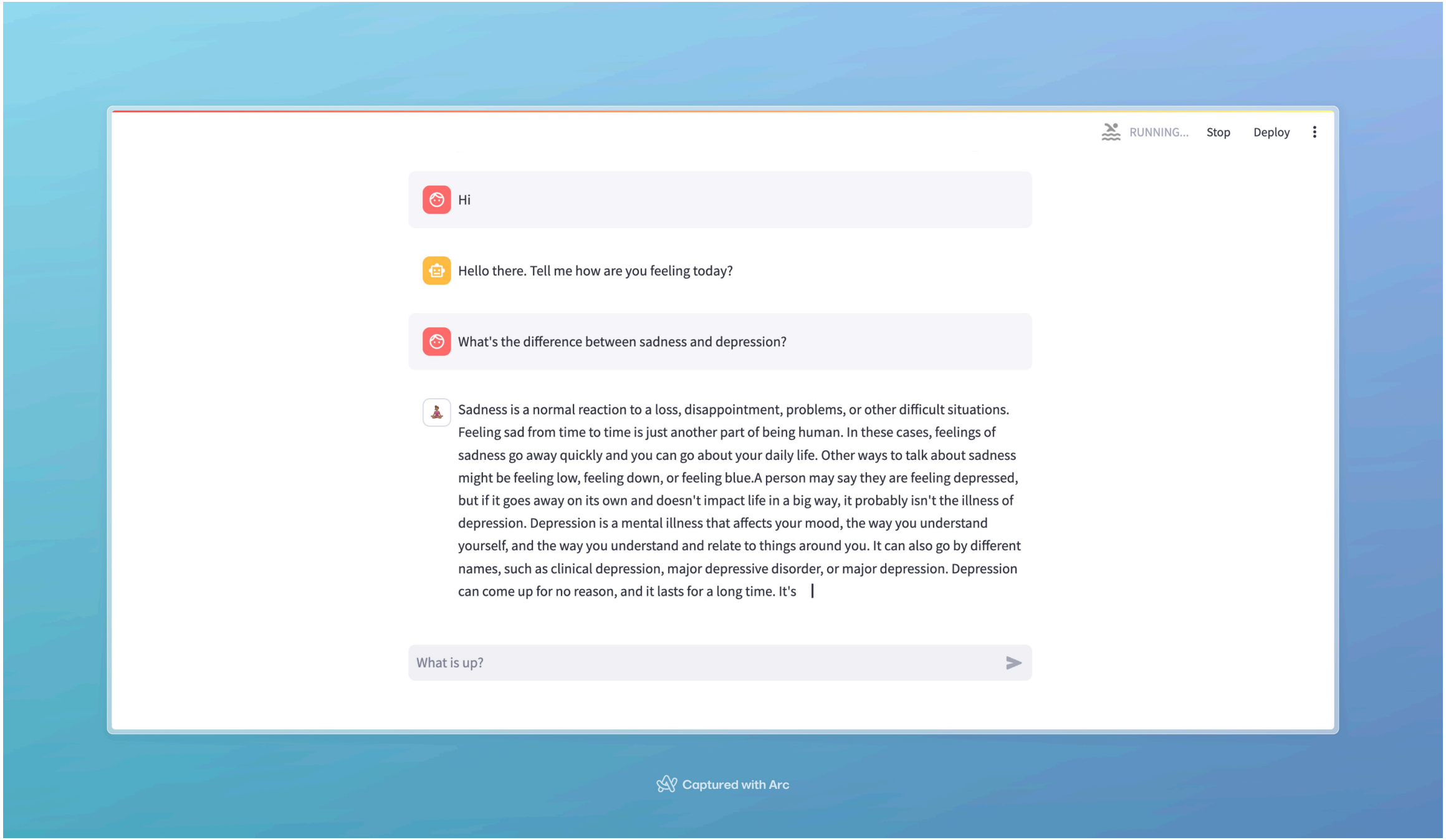
```python
# Prediction Model Deployment

# A trained SVM model named 'model' and a vectorizer named 'vectorizer'

# Function to predict intents based on user input
def predict_intent(user_input):
    # Vectorize the user input
    user_input_vec = vectorizer.transform([user_input])

    # Predict the intent
    intent = model.predict(user_input_vec)[0]

    return intent


# Function to generate responses based on predicted intents
def generate_response(intent):
    # Implement your logic here to generate appropriate responses based on the predicted intents
    if intent == 'greeting':
        response = "Hello! How can I assist you today?"
    elif intent == 'farewell':
        response = "Goodbye! Take care."
    elif intent == 'question':
        response = "I'm sorry, I don't have the information you're looking for."
    else:
        response = "I'm here to help. Please let me know how I can assist you."

    return response


# Example usage
while True:
    # Get user input
    user_input = input("User: ")

    # Predict intent
    intent = predict_intent(user_input)

    # Generate response
    response = generate_response(intent)

    print("Chatbot:", response)
```

```
User: hi
Chatbot: Hello! How can I assist you today?
User: I am feeling unwell
Chatbot: I'm here to help. Please let me know how I can assist you.
User: farewell
Chatbot: I'm here to help. Please let me know how I can assist you.
User: How do I know if I'm unwell?
Chatbot: Hello! How can I assist you today?

User:
```

```python
# Preprocess data and create a dictionary of patterns and responses
patterns_dict = {}
for intent in data['intents']:
    for pattern, response in zip(intent['patterns'], intent['responses']):
        patterns_dict[pattern.lower()] = response

# Function to predict intent
def predict_intent(user_input):
    # Convert user input to lowercase for case-insensitive matching
    user_input_lower = user_input.lower()

    # Search for matching pattern in the dictionary
    if user_input_lower in patterns_dict:
        return patterns_dict[user_input_lower]
    else:
        return "I'm here to help. Please let me know how I can assist you."

# Streamed response emulator
def response_generator():
    response = predict_intent(prompt)  # Use the function to get response based on user input
    for word in response.split():
        yield word + " "
        time.sleep(0.05)


st.markdown("<h1 style='text-align: center;'>Enigma Chatbot</h1>", unsafe_allow_html=True)
st.markdown("<h1 style='text-align: center;'>🧘 A chatbot for MentalHealth 🧘</h1>", unsafe_allow_html=True)

# Initialize chat history
if "messages" not in st.session_state:
    st.session_state.messages = []
```

```python
# Display chat messages from history on app rerun
for message in st.session_state.messages:
    with st.chat_message(message["role"]):
        st.markdown(message["content"])

# Accept user input
if prompt := st.chat_input("What is up?"):
    # Add user message to chat history
    st.session_state.messages.append({"role": "user", "content": prompt})
    # Display user message in chat message container
    with st.chat_message("user"):
        st.markdown(prompt)

    # Display assistant response in chat message container
    with st.chat_message("🧙"):
        response = st.write_stream(response_generator())
    # Add assistant response to chat history
    st.session_state.messages.append({"role": "assistant", "content": response})
```


Deploy
Enigma Chatbot
🧘🏾‍♀️ A chatbot for MentalHealth 🧘🏾‍♀️
What is up?
Captured with Arc


RUNNING...
Stop
Deploy
Hi
Hello there. Tell me how are you feeling today?
What's the difference between sadness and depression?
Sadness is a normal reaction to a loss, disappointment, problems, or other difficult situations. Feeling sad from time to time is just another part of being human. In these cases, feelings of sadness go away quickly and you can go about your daily life. Other ways to talk about sadness might be feeling low, feeling down, or feeling blue.A person may say they are feeling depressed, but if it goes away on its own and doesn't impact life in a big way, it probably isn't the illness of depression. Depression is a mental illness that affects your mood, the way you understand yourself, and the way you understand and relate to things around you. It can also go by different names, such as clinical depression, major depressive disorder, or major depression. Depression can come up for no reason, and it lasts for a long time. It's
What is up?
Captured with Arc

Deploy

concentrate or make decisions. Most people lose interest in things that they used to enjoy and may isolate themselves from others. There are also physical signs of depression, such as problems with sleep, appetite and energy and unexplainable aches or pains. Some may experience difficult thoughts about death or ending their life (suicide). Depression lasts longer than two weeks, doesn't usually go away on its own, and impacts your life. It's a real illness, and it is very treatable. It's important to seek help if you're concerned about depression.

How do I know if I'm unwell?

If your beliefs , thoughts , feelings or behaviours have a significant impact on your ability to function in what might be considered a normal or ordinary way, it would be important to seek help.

Bye

See you later.

What is up?

Conclusion

This project leveraged Streamlit to create a visually appealing mental health support chatbot, integrating NLP and deep learning techniques. The chatbot demonstrated efficacy in identifying mental health concerns and fostering user connection through personalized interactions. However, challenges remain in assessing its long-term impact and mitigating biases inherent in training data. Future endeavors should prioritize evaluating the chatbot's long-term effectiveness, refining its emotional intelligence, and integrating it with professional mental health services. Additionally, efforts should focus on fortifying user data privacy to ensure ethical and sustainable support. By addressing these aspects, chatbots can become valuable tools in providing accessible and effective mental health support.

Future Enhancement

1. **Long-term impact:** Conduct studies to assess the chatbot's effectiveness in improving mental health outcomes over extended periods.
2. **Bias mitigation:** Analyze and address potential biases in the training data to ensure equitable support for diverse users.
3. **Emotional intelligence:** Refine the chatbot's ability to understand and respond to complex emotions for more nuanced interactions.
4. **Integration with professionals:** Explore ways to connect users with professional help when needed, ensuring seamless transitions.
5. **Privacy and security:** Implement robust measures to ensure user data privacy and enhance trust in the chatbot's ethical use.