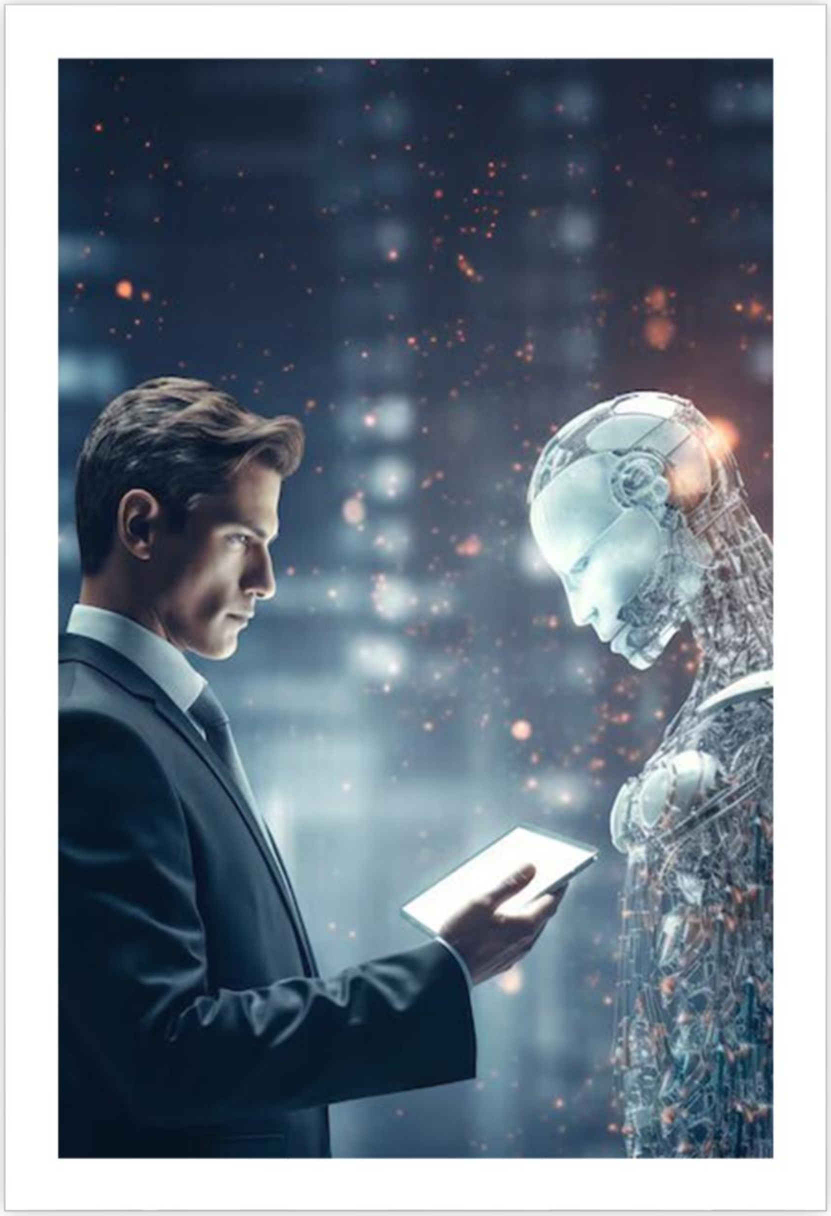


AI Consultant Application with Referral System

FINANCIAL AND BUSINESS MODELLING

Manish Walurkar | Feynn Labs | November 27, 2024

Abstract

Mental health challenges have become increasingly prevalent in modern society, with many individuals hesitant to seek professional help. This reluctance can lead to prolonged emotional distress or worsening conditions. The **AI Consultant Application** leverages advanced Natural Language Processing (NLP) and Artificial Intelligence (AI) technologies to engage users in meaningful conversations, assess their emotional well-being, and provide actionable guidance. By generating a **Mental Stability Score (MSS)**, the application determines the severity of the user's condition and either suggests personalized remedies or recommends professional therapists if necessary. This solution combines AI-driven empathy with a seamless referral system to address a critical gap in the mental health ecosystem.

Problem Statement

Mental health is a pressing concern worldwide. According to the **National Crime Records Bureau (NCRB)**, India's suicide rate is 12.4 per 100,000 people. Additionally, **Swachh India (NDTV)** reports that nearly 80% of individuals experiencing mental health issues avoid seeking professional help due to stigma, financial barriers, or lack of access.

Key contributors to deteriorating mental health include academic pressure, financial instability, and strained relationships. While some stressors can be managed through self-help techniques or casual conversations, prolonged or severe mental distress often requires professional intervention.

The proposed application aims to provide:

1. **Immediate Support:** A conversational chatbot for addressing mild mental health concerns.
2. **Professional Guidance:** A referral system to connect users with licensed therapists when their MSS crosses a critical threshold.
3. **Accessibility:** A user-friendly, judgment-free platform available 24/7.

How AI Helps

AI technologies, especially NLP models, bring unique capabilities to mental health solutions:

1. **Real-Time Emotional Analysis:** AI analyzes user conversations to detect emotional cues, patterns, and stress indicators, ensuring the responses are contextually relevant and empathetic.
2. **Non-Judgmental Environment:** The chatbot provides a safe space for users to express themselves without fear of stigma or judgment.
3. **Personalized Remedies:** By considering historical interactions and user preferences, the chatbot offers tailored advice, such as mindfulness exercises or mood-boosting strategies.
4. **Scalable Support:** AI allows the application to handle multiple users simultaneously, making it a cost-effective solution for addressing widespread mental health needs.
5. **Referral Recommendations:** When the chatbot identifies severe cases, it seamlessly connects users to professional therapists, bridging the gap between self-help and expert intervention.

Bringing the change with the application?

Existing mental health apps like **Calm**, **Headspace**, and **Wysa** focus primarily on feature like guided meditations, motivational content, and rule-based chat bots. While effective for general wellness, these solutions often lack:

- **Personalized Interaction:** Most apps provide generic advice rather than tailoring responses to individual user needs.
- **Therapist Integration:** Few platforms offer direct connections to professional therapists.
- **Comprehensive Features:** Combining empathy-driven conversations, actionable advice, and professional referrals in one solution is rare.

The **AI Consultant Application** addresses these gaps by offering a personalized chatbot with real-time MSS evaluation, a referral system for therapists, and additional features like sleep tracking and meditative soundscapes.

Implementation Plan

Core Functionalities:

1. **Login System:**
   - User-friendly account creation with options for social media logins.
   - Secure session management to ensure data privacy.
2. **AI-Powered Chatbot:**
   - Employing **Transformer-based models** like GPT or BERT for natural, empathetic conversations.
   - Continuous learning from user interactions to improve response quality.
3. **Mental Stability Scoring (MSS):**
   - Generated using sentiment analysis, emotion detection, and surveys.
   - MSS serves as a critical metric to determine the user's mental state and recommend next steps.
4. **Referral System with Appointment Booking:**
   - Displays licensed therapists based on user location.
   - Facilitates appointment scheduling and provides reviews for informed decisions.
5. **Meditation and Sleep Monitoring:**
   - Offers calming music and tracks sleep patterns to enhance overall well-being.

Market Analysis and Customer Segmentation

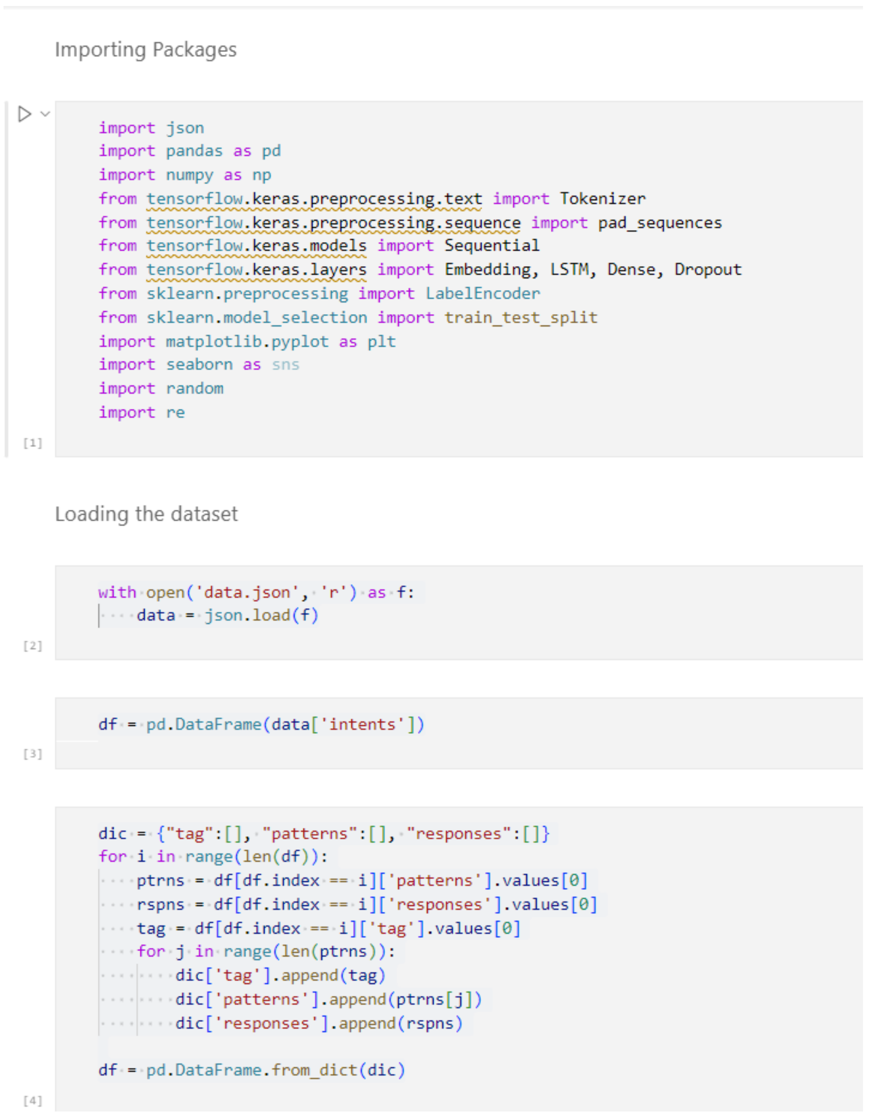
MARKET GROWTH:

- The global mental health app market was valued at **$4.2 billion in 2021** and is projected to grow at a **CAGR of 15–20% through 2028**.

CUSTOMER SEGMENTS:

1. **Teens (13–17):** Addressing school-related stress and peer pressure.
2. **Young Adults (18–24):** Coping with relationships, academics, and early career challenges.
3. **Working Adults (25–54):** Managing work-life balance and family responsibilities.
4. **Seniors (55+):** Dealing with retirement, health issues, and grief.

Implementation of code



Importing Packages

```python
import json
import pandas as pd
import numpy as np
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import seaborn as sns
import random
import re
```

Loading the dataset

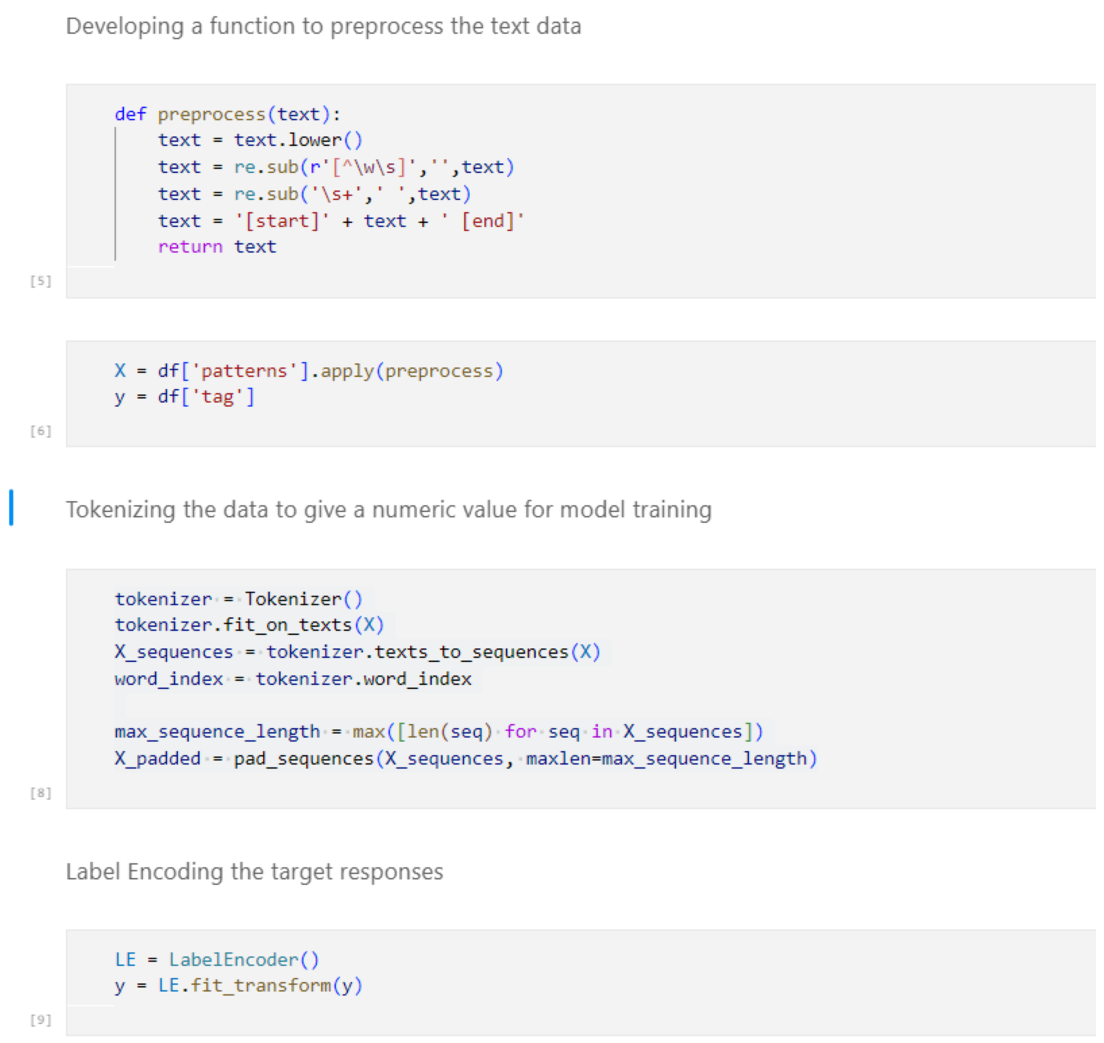
```python
with open('data.json', 'r') as f:
    data = json.load(f)
```

```python
df = pd.DataFrame(data['intents'])
```

```python
dic = {"tag":[], "patterns":[], "responses":[]}
for i in range(len(df)):
    ptrns = df[df.index == i]['patterns'].values[0]
    rspns = df[df.index == i]['responses'].values[0]
    tag = df[df.index == i]['tag'].values[0]
    for j in range(len(ptrns)):
        dic['tag'].append(tag)
        dic['patterns'].append(ptrns[j])
        dic['responses'].append(rspns)

df = pd.DataFrame.from_dict(dic)
```

Description:

This code reads training data from a JSON file (data.json) and organizes it into a structured format using a pandas Data Frame. Each entry includes a tag (representing a category or intent), patterns (possible user inputs), and responses (corresponding bot replies). The structured data is prepared for further processing.



Developing a function to preprocess the text data

```
def preprocess(text):
    text = text.lower()
    text = re.sub(r'[^\w\s]','',text)
    text = re.sub('\s+',' ',text)
    text = '[start]' + text + ' [end]'
    return text
```

[5]

```
X = df['patterns'].apply(preprocess)
y = df['tag']
```

[6]

Tokenizing the data to give a numeric value for model training

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts(X)
X_sequences = tokenizer.texts_to_sequences(X)
word_index = tokenizer.word_index

max_sequence_length = max([len(seq) for seq in X_sequences])
X_padded = pad_sequences(X_sequences, maxlen=max_sequence_length)
```

[8]

Label Encoding the target responses

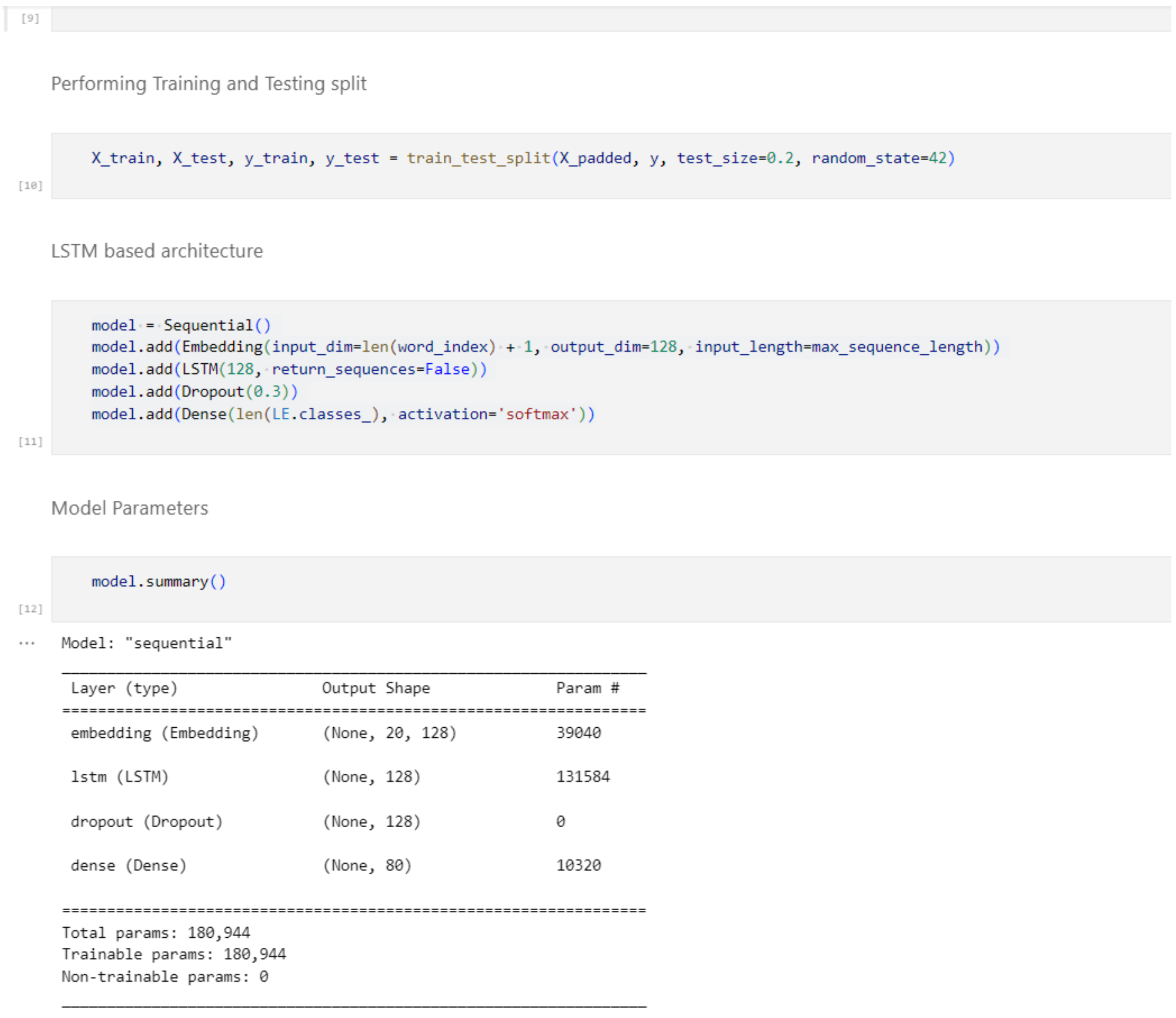
```
LE = LabelEncoder()
y = LE.fit_transform(y)
```

[9]

Description:

This function preprocesses text by:

1. Converting it to lowercase.
2. Removing punctuation and excess whitespace.
3. Adding special markers ([start] and [end]) to indicate the beginning and end of a sentence.
   This ensures the input data is clean and ready for tokenization.
4. This section tokenizes the input text using a Tokenizer, converting words into numeric sequences for model training. The sequences are padded to ensure uniform length using pad_sequences. The dataset is then split into training and testing sets using an 80-20 ratio.



[9]

Performing Training and Testing split

```
X_train, X_test, y_train, y_test = train_test_split(X_padded, y, test_size=0.2, random_state=42)
```

[10]

LSTM based architecture

```
model = Sequential()
model.add(Embedding(input_dim=len(word_index) + 1, output_dim=128, input_length=max_sequence_length))
model.add(LSTM(128, return_sequences=False))
model.add(Dropout(0.3))
model.add(Dense(len(LE.classes_), activation='softmax'))
```

[11]

Model Parameters

```
model.summary()
```

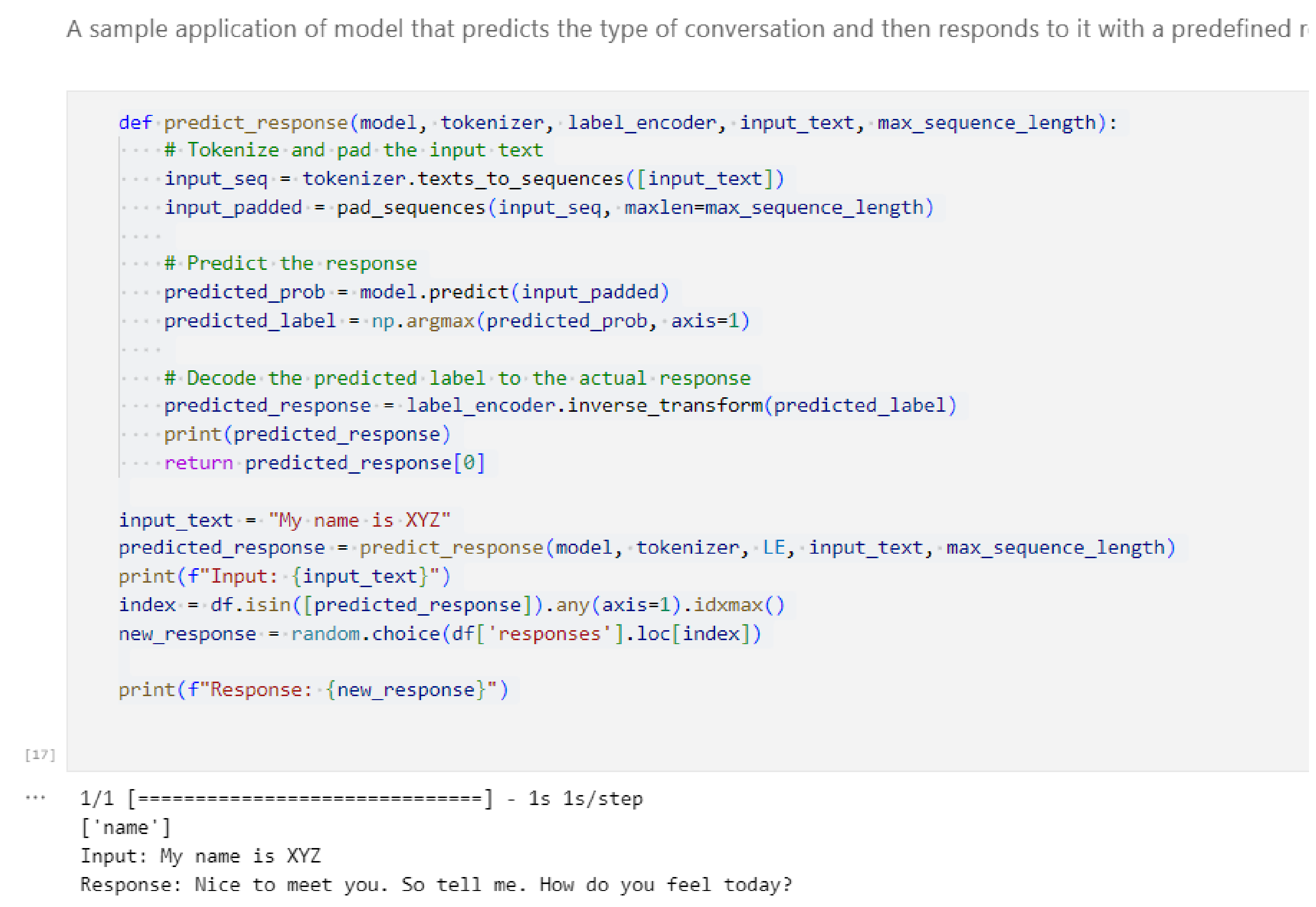
[12]

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 20, 128)           39040

 lstm (LSTM)                 (None, 128)               131584

 dropout (Dropout)           (None, 128)               0

 dense (Dense)               (None, 80)                10320

=================================================================
Total params: 180,944
Trainable params: 180,944
Non-trainable params: 0
_________________________________________________________________
```

Description:

A deep learning model is built using TensorFlow/Keras.

- **Embedding Layer:** Transforms input tokens into dense vector representations.
- **LSTM Layer:** Captures sequential patterns in the text.
- **Dropout Layer:** Reduces overfitting during training.
- **Dense Layer:** Outputs probabilities for each intent category using the softmax activation function.
  The model is trained on the prepared data for 100 epochs with a batch size of 32, validating against the test set.

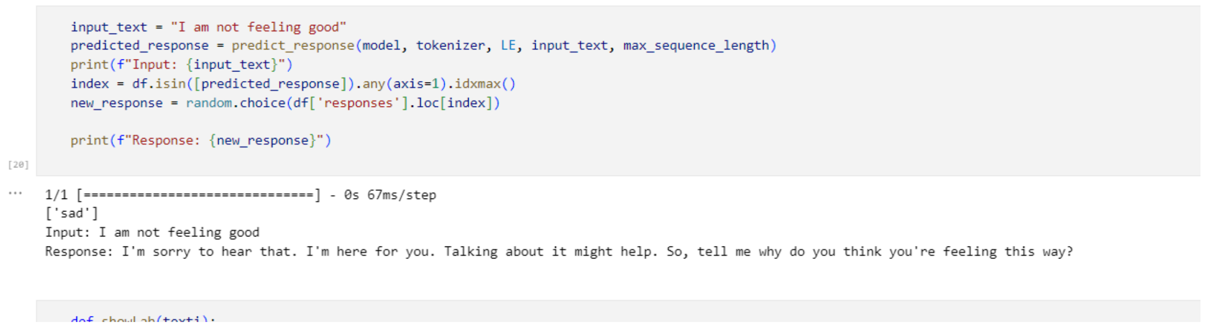

A sample application of model that predicts the type of conversation and then responds to it with a predefined r

```python
def predict_response(model, tokenizer, label_encoder, input_text, max_sequence_length):
    # Tokenize and pad the input text
    input_seq = tokenizer.texts_to_sequences([input_text])
    input_padded = pad_sequences(input_seq, maxlen=max_sequence_length)
    
    # Predict the response
    predicted_prob = model.predict(input_padded)
    predicted_label = np.argmax(predicted_prob, axis=1)
    
    # Decode the predicted label to the actual response
    predicted_response = label_encoder.inverse_transform(predicted_label)
    print(predicted_response)
    return predicted_response[0]

input_text = "My name is XYZ"
predicted_response = predict_response(model, tokenizer, LE, input_text, max_sequence_length)
print(f"Input: {input_text}")
index = df.isin([predicted_response]).any(axis=1).idxmax()
new_response = random.choice(df['responses'].loc[index])

print(f"Response: {new_response}")
```

```
1/1 [==============================] - 1s 1s/step
['name']
Input: My name is XYZ
Response: Nice to meet you. So tell me. How do you feel today?
```

Description:
This function takes user input and:

1. Tokenizes and pads the input text.
2. Predicts the intent using the trained model.
3. Decodes the prediction to find the corresponding response.
4. Randomly selects a suitable response from the dataset.



```python
input_text = "I am not feeling good"
predicted_response = predict_response(model, tokenizer, LE, input_text, max_sequence_length)
print(f"Input: {input_text}")
index = df.isin([predicted_response]).any(axis=1).idxmax()
new_response = random.choice(df['responses'].loc[index])

print(f"Response: {new_response}")
```

```
1/1 [==============================] - 0s 67ms/step
['sad']
Input: I am not feeling good
Response: I'm sorry to hear that. I'm here for you. Talking about it might help. So, tell me why do you think you're feeling this way?
```

```python
def showLab(texti):
```

For example:

- **Input:** "I am not feeling good"
- **Output:** "I am sorry to hear that, I am here for you. Why do you think you are feeling this way?"

This example demonstrates the foundational steps in developing a chatbot, including data processing, model training, and response generation. While functional, the code is only a prototype and will require enhancements to handle scalability, diverse inputs, and complex interactions. Future improvements may include integrating advanced transformers or pre-trained language models for better performance.

Mental Stability Score (MSS): Concept and Implementation

The **Mental Stability Score (MSS)** is a critical feature of the AI Consultant Application, designed to assess a user's emotional and mental well-being during interactions with the chatbot. The MSS acts as a dynamic metric, calculated based on various user inputs and contextual factors. This score enables the application to determine whether the user's mental state requires additional support, such as professional therapy, and triggers a recommendation when specific thresholds are crossed.

KEY COMPONENTS FOR ACHIEVING THE MSS

The calculation of the MSS can be achieved through a combination of the following methodologies:

1. **Sentiment Analysis**

- **Objective:** Detect the emotional tone of user messages to classify them as positive, neutral, or negative.
- **Implementation:**
  - Use NLP techniques like VADER (Valence Aware Dictionary for Sentiment Reasoning) for basic sentiment analysis.
  - For more accuracy, pre-trained transformer models like **BERT** or **DistilBERT** can be fine-tuned on labeled sentiment datasets.
  - Sentiments such as sadness, anger, and anxiety can reduce the MSS, while positive emotions can increase it.

2. Emotion Detection

- **Objective:** Identify specific emotions conveyed in user responses.
- **Implementation:**
  - Train a deep learning model (e.g., LSTM, CNN) on a dataset like **ISEAR (International Survey on Emotion Antecedents and Reactions)** or use pre-trained emotion classification models.
  - Recognize key emotions such as fear, frustration, guilt, or contentment.
  - Assign weights to these emotions to calculate their impact on the MSS (e.g., anger may lower the score by 10%, while hope may increase it by 5%).

3. Behavioral Patterns and Linguistic Features

- **Objective:** Monitor changes in communication patterns over time to detect distress signals.
- **Implementation:**
  - Track parameters like message length, frequency of negative words, or abrupt tone shifts.
  - Build time-series models to observe trends, such as a consistent use of negative language over multiple sessions.

4. Survey and Questionnaire Integration

- **Objective:** Collect structured responses from users to supplement conversational analysis.
- **Implementation:**
  - Periodically prompt users with short mental health surveys or mood check-ins.
  - Use validated psychological scales, such as the **GAD-7 (Generalized Anxiety Disorder)** or **PHQ-9 (Patient Health Questionnaire)**, to gather quantitative data.
  - Normalize the survey scores to integrate them into the overall MSS computation.

5. Historical Data and Contextual Awareness

- **Objective:** Incorporate user history to provide a comprehensive understanding of mental stability trends.
- **Implementation:**
  - Store anonymized data from previous sessions, including sentiment scores and recurring issues.

  - Calculate rolling averages of past MSS values to detect prolonged distress patterns.
  - Weight historical scores less heavily than real-time inputs to prioritize the present state.

Triggering the Recommendation System

The MSS serves as a real-time decision-making tool for the chatbot. Once the score is calculated during a session, it guides the application to take appropriate actions.

THRESHOLDS AND TRIGGERS

1. **Safe Zone:**
   - MSS > 70: Indicates that the user is in a positive or neutral mental state.
   - **Response:** The chatbot continues providing general advice, conversation, or calming activities.
2. **Warning Zone:**
   - MSS between 40 and 70: Suggests mild to moderate mental distress.
   - **Response:** The chatbot offers targeted remedies such as breathing exercises, motivational quotes, or self-care routines. It also gently inquires about the user's willingness to seek further assistance.
3. **Critical Zone:**
   - MSS < 40: Indicates significant emotional distress or potential mental health risks.
   - **Response:** The chatbot immediately recommends professional therapy services.
     - A curated list of therapists is presented based on the user's location and preferences.
     - If the user agrees, the app transitions to the **appointment booking interface**, ensuring a seamless handover to professional care.

Business Model

REVENUE STREAMS:

1. **Commission-Based Referral System:**
    - The app earns a commission for every successful therapy booking facilitated through the platform.
    - Example: For a $100 therapy session, a 10% commission generates $10 in revenue.
2. **Freemium Model:**
    - Basic chatbot access is free, with optional premium features (e.g., personalized wellness plans) available via subscription.
3. **Advertisements and Sponsorships:**
    - Revenue from in-app advertisements, targeting wellness products and services.
    - Sponsored listings for therapists seeking prominent placement on the platform.

PARTNERSHIPS:

- Collaboration with licensed therapists and therapy centers to expand the referral network.
- Potential integration with healthcare organizations for broader reach.

Financial Modeling

REVENUE EQUATION:

Developing the financial equation

- Y as total profit
- Price of product (suppose m)
- Total sale as a function of time (x(t))
- Total production and maintenance cost (c)

$$Y = m\,x(t) - c$$

Estimating the cost of production of application is:

- Production cost: estimating 15 – 20 lakhs including tools and post production for a startup level business
- Estimating monthly maintenance cost : 1 – 2 lakhs
- Maintenance cost for a year , 12 – 15 lakh

Pricing of the product:

- 10 – 15 percent as commission
- If average therapy session costs 1000 – 5000, taking median 3000
- Assuming 50,000 user base, with 10,000 active users
- Conversion rate: 5 % = 500 users
- Revenue generated in a month = 500 x 450 (15 % of 3000) = 2,25,000

Total sale as a function of time:

- For x(t) =100, y= –55,000 (loss).
- For x(t)=200, y= –10,000 (loss).
- For x(t)=300, y= 35,000 (profit).
- For x(t)=400, y= 80,000 (profit).
- For x(t)=500, y= 125,000 (profit).

Revised Revenue Equation:

For a monthly period of time including the production cost:

- M = 450rs on 3000rs (average session pricing)
- X(t)= 500
- C= 1,00,000rs for a month

$Y = 450 * 500 - 1,00,000$

$Y = 1,25,000$

Privacy and Regulatory Compliance

- **Data Protection:** Compliance with India’s **IT Act, 2000**, and the upcoming **Digital Personal Data Protection Act, 2023**.
- **User Consent:** Explicit consent required for collecting sensitive mental health data.
- **Licensing:** Partner only with certified therapists and adhere to ethical referral practices.

Conclusion

The AI Consultant Application redefines mental health support by offering a unique combination of personalized AI-driven conversations and seamless therapist referrals. Its innovative features, such as MSS evaluation and a commission-based referral model, position it as a game-changing solution in the mental health industry. By addressing both mild and severe cases, the app aims to promote early intervention, reduce stigma, and improve mental health outcomes on a global scale.