Medibot Project Report

Artificial Intelligence

Third Year-Computer and Information Systems Engineering

Batch: 2022

Group Members:

Ms. Shiza Ejaz ……………………………………………… (CS-22060)

Ms. Fatima Mehdvi ………………………………………….. (CS-22065)

Mr. Hussain Kazmi …………………………………………... (CS-22090)

Mr. Taha Khan ……………………………………………….… (CS-22134)

Submitted to: Dr. Saad Qasim Khan

Submission Date: 20/11/2024

Table of Contents

Abstract:

The Medibot project is a Complex Engineering Problem (CEP) addressing the challenges in healthcare accessibility by utilizing Artificial Intelligence (AI) to develop a virtual healthcare assistant. Medibot combines machine learning techniques with natural language processing (NLP) to assist users in identifying potential health conditions based on their symptoms. By processing medical data and analyzing user inputs, Medibot provides personalized health predictions and actionable insights in real time.

This project employs supervised learning through Feedforward Neural Networks (FFNs) for disease prediction and integrates NLP for understanding and extracting symptoms from user inputs. A robust data preprocessing pipeline ensures the model is trained effectively on clean and standardized data, leading to reliable outcomes. Additionally, Medibot incorporates fuzzy matching algorithms to enhance symptom recognition, making it user-friendly and capable of understanding natural language interactions.

The achievements of this project include:

1. Developing a highly accurate disease prediction model with over 85% precision on the test data.
2. Designing an interactive and intuitive system for users to communicate symptoms effectively.
3. Creating a scalable framework that can be extended to accommodate more diseases and symptoms.
4. Reducing dependency on immediate human intervention for initial medical assessments, thereby acting as a supportive tool for healthcare professionals and patients alike.

Medibot not only demonstrates the potential of AI in improving healthcare accessibility but also empowers individuals to make informed decisions about their health. By bridging the gap between patients and healthcare resources, this project showcases the transformative role of AI in addressing modern-day healthcare challenges.

Problem Statement:

The Medibot project tackles a critical issue in modern healthcare: the accessibility of timely and accurate diagnostic support. With growing demands on healthcare systems worldwide, patients often face delays in receiving initial assessments and recommendations. This gap highlights the need for innovative solutions to enhance healthcare delivery.

Medibot is designed as a virtual healthcare assistant leveraging Artificial Intelligence (AI) to assist users in identifying potential diseases based on symptoms. By employing a combination of supervised learning and natural language processing (NLP), Medibot facilitates seamless interactions with users, enabling them to communicate their symptoms in natural language.

The problem being addressed can be summarized as follows:

1. Limited availability of healthcare professionals for initial consultations.
2. The need for accessible, real-time diagnostic support to reduce patient stress and delays.
3. Inefficiencies in recognizing symptoms and connecting them to potential conditions.

Medibot aims to bridge this gap by providing users with a quick, reliable, and user-friendly platform to analyze symptoms and receive potential disease predictions. It reduces dependency on immediate human intervention and empowers patients to take the first steps toward healthcare management.

This project demonstrates how AI technologies, when integrated effectively, can offer impactful solutions to enhance everyday healthcare experiences.

Theoretical details of the model:

The Medibot project comprises several key modules, each designed to address specific aspects of symptom analysis, disease prediction, and user interaction. Below is an explanation of these modules and their submodules, accompanied by a flowchart illustrating the algorithm's flow.

1. Data Preparation Module

Purpose: To prepare the medical dataset for training and testing.

- **Submodules**:
  1. **Data Cleaning**: Handling missing values by replacing them with zeros.
  2. **Label Encoding**: Converting disease names into numerical labels using a Label Encoder.
  3. **Feature Scaling**: Standardizing features to ensure consistent input to the machine learning model.

2. Interaction Module (NLP Integration)

Purpose: To facilitate natural language-based interaction with the user.

- **Submodules**:
  1. **Symptom Extraction**: Tokenizing user input and matching tokens with the symptom list using fuzzy matching.
  2. **Suggestion Engine**: Recommending additional symptoms for clarification based on partially matched diseases.

3. Prediction Module

Purpose: To predict diseases based on user-provided symptoms.

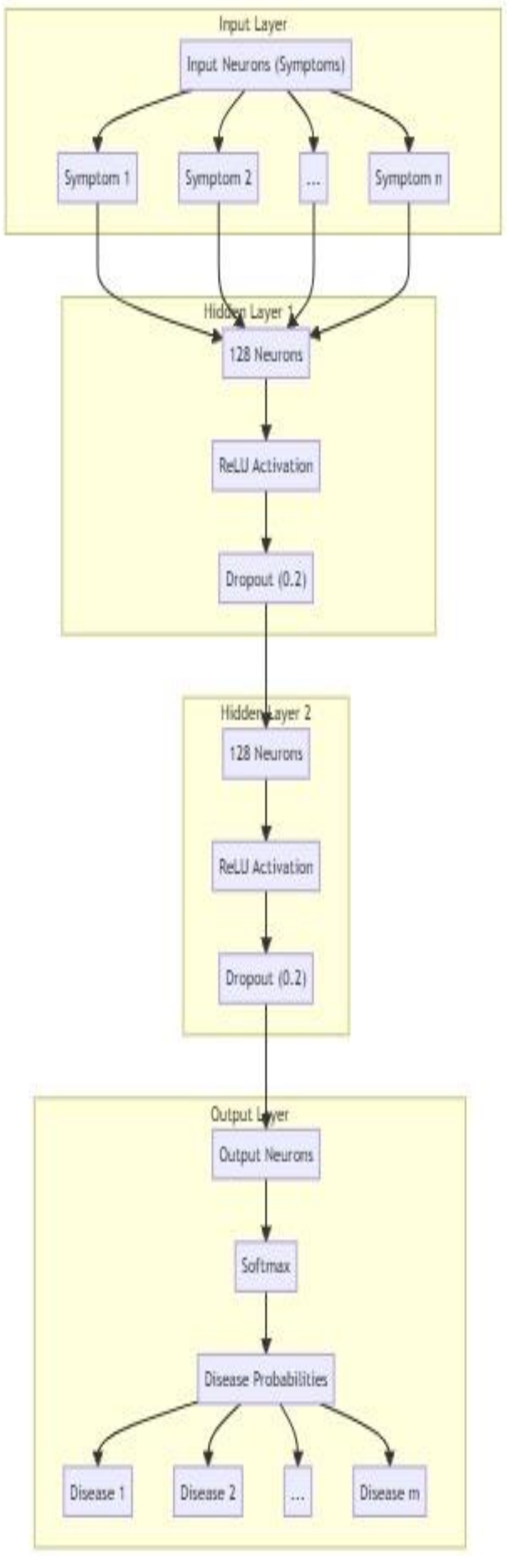
- **Submodules**:
  1. **Symptom Matching**: Using NLP and fuzzy matching to extract symptoms from user input.
  2. **Feature Vector Creation**: Mapping user symptoms to the pre-defined symptom list to create a binary vector.
  3. **Disease Prediction**: Inputting the binary vector into the trained FFN to get probabilities for each disease.

2. Model Training Module

Purpose: To train the Feedforward Neural Network (FFN) for disease prediction.

- **Submodules**:
  1. **Network Architecture**:
     - Input Layer: Accepts scaled symptom data.
     - Hidden Layers: Two fully connected layers with ReLU activation and dropout for regularization.
     - Output Layer: Provides probabilities for potential diseases.
  2. **Loss Function**: Cross-entropy for handling multi-class classification.

3. **Optimizer**: Adam for efficient gradient descent.



Medibot's Feed Forward Network model

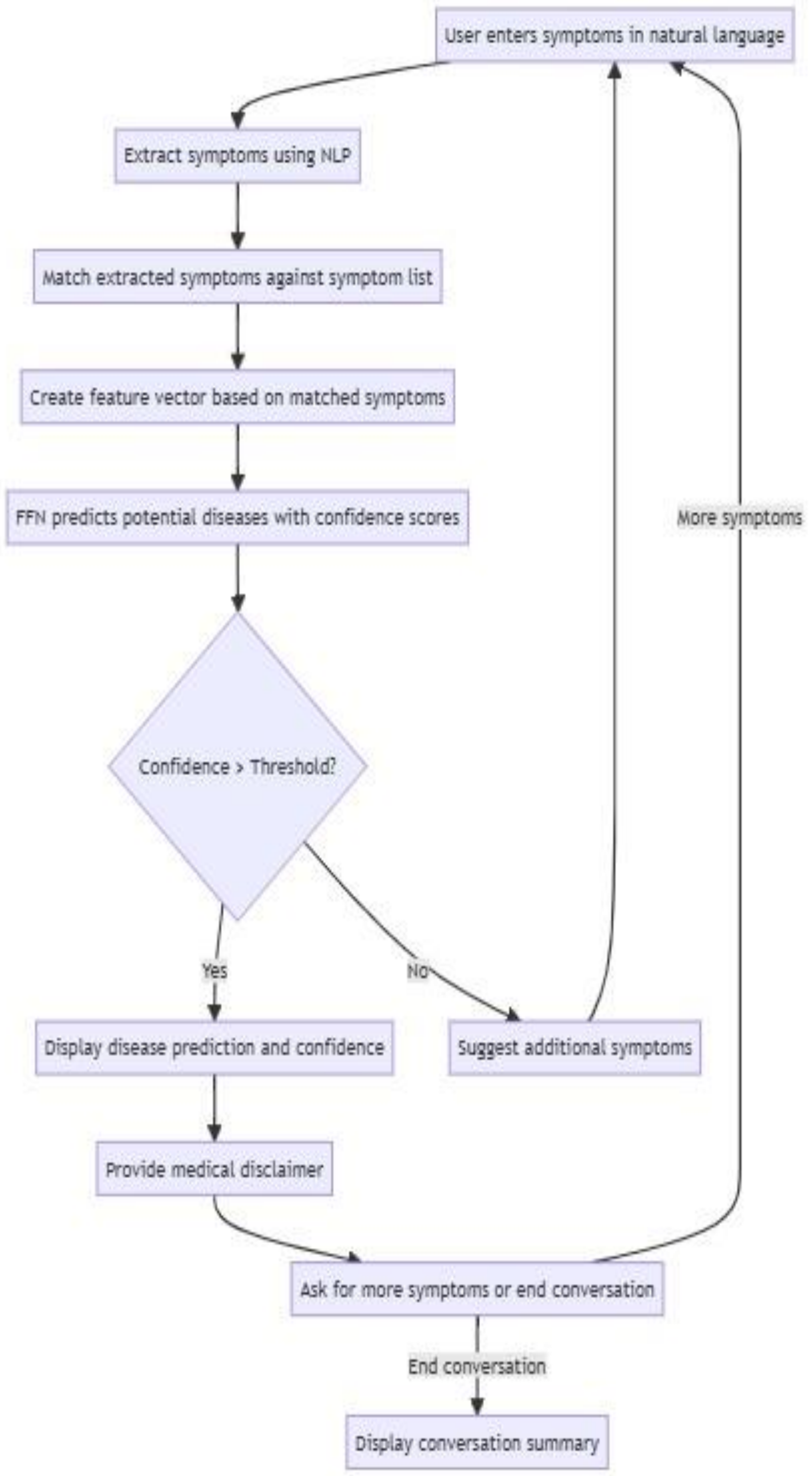
Architecture of the Feedforward Neural Network (FFN)

1. **Input Layer**:
   - The input layer represents the symptoms, with each input neuron corresponding to a symptom in the dataset.
   - The total number of input neurons (**n**) matches the number of symptoms in the predefined symptom list.
2. **Hidden Layer 1**:
   - This layer contains 128 neurons.
   - It performs a linear transformation on the input data, followed by a ReLU activation function to introduce non-linearity.
   - A dropout mechanism with a rate of 0.2 is applied to prevent overfitting by randomly deactivating neurons during training.
3. **Hidden Layer 2**:
   - Similar to the first hidden layer, this layer also comprises 128 neurons.
   - It applies the same operations as Hidden Layer 1, including a ReLU activation function and a dropout with a rate of 0.2.
4. **Output Layer**:
   - The number of output neurons (**m**) corresponds to the total number of possible diseases in the dataset.
   - This layer transforms the output from Hidden Layer 2 into raw scores for each disease.
5. **Softmax Activation**:
   - The raw scores from the output layer are passed through a Softmax activation function.
   - This converts the scores into probabilities, ensuring they sum up to 1, making it easier to interpret the likelihood of each disease.

Data Flow Through the Network

1. The input layer receives a binary vector representing the presence (1) or absence (0) of each symptom.
2. This data is passed through the first hidden layer, where it undergoes a linear transformation, ReLU activation, and dropout.
3. The output of the first hidden layer is then passed through the second hidden layer, following the same process.
4. Finally, the data reaches the output layer, where it's transformed into raw scores for each disease.
5. The softmax function is applied to these scores, producing a probability distribution over all possible diseases.
6. These probabilities are the final output, representing the likelihood of each disease given the input symptoms.

Flow of Medibot Algorithm


User enters symptoms in natural language
Extract symptoms using NLP
Match extracted symptoms against symptom list
Create feature vector based on matched symptoms
FFN predicts potential diseases with confidence scores
More symptoms
Confidence > Threshold?
Yes
No
Display disease prediction and confidence
Suggest additional symptoms
Provide medical disclaimer
Ask for more symptoms or end conversation
End conversation
Display conversation summary

Program for implementation:

The implementation of the Medibot system combines Python programming with advanced libraries for AI, natural language processing, and machine learning. Below is an explanation of the core components along with representative code snippets.

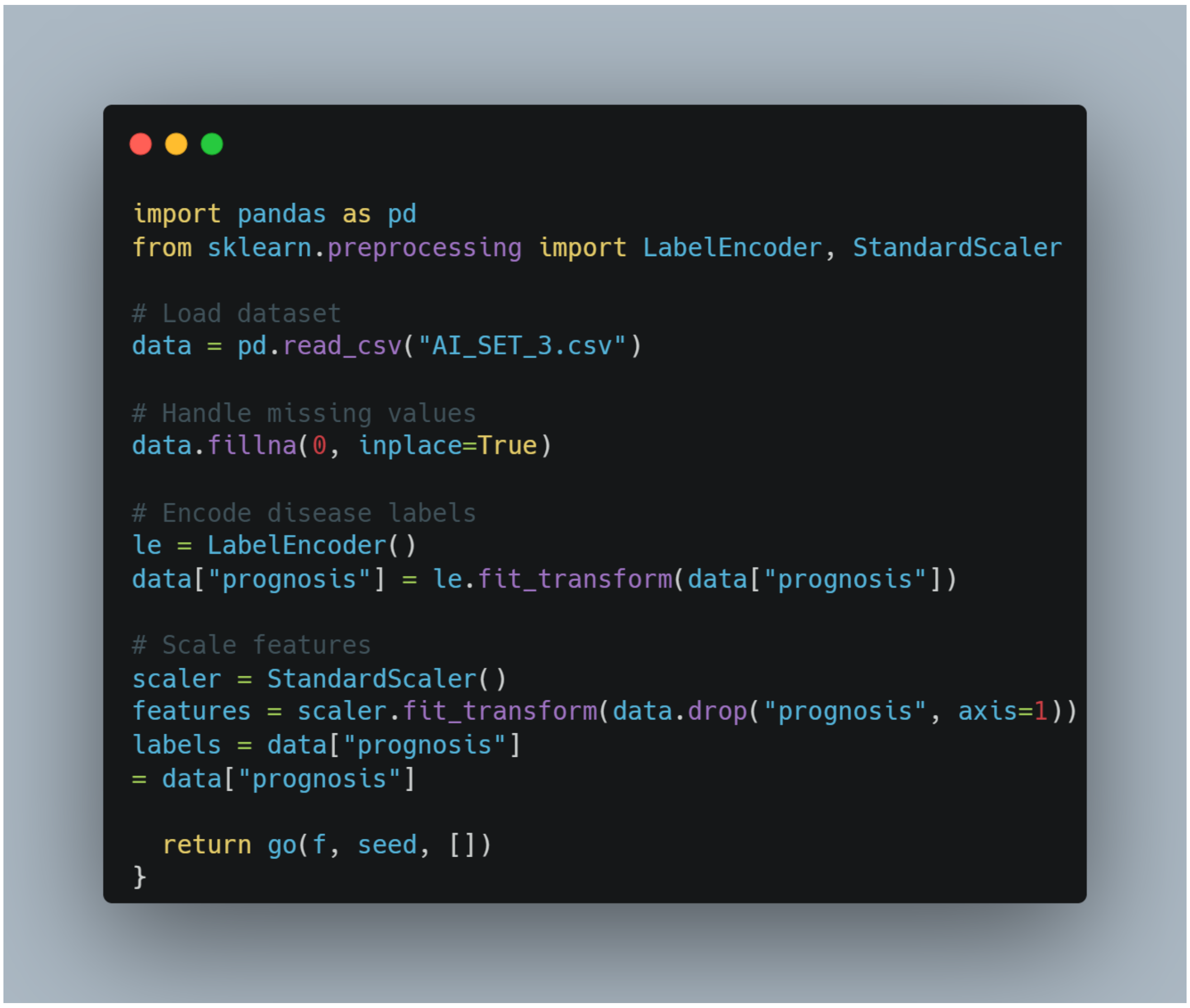
Language Used: Python

Core Libraries:

- **PyTorch**: For building and training the Feedforward Neural Network (FFN).
- **Scikit-learn**: For preprocessing, including label encoding and scaling.
- **SpaCy and FuzzyWuzzy**: For natural language processing and symptom matching.
- **Pandas**: For data manipulation and analysis.

1. Data Preprocessing:

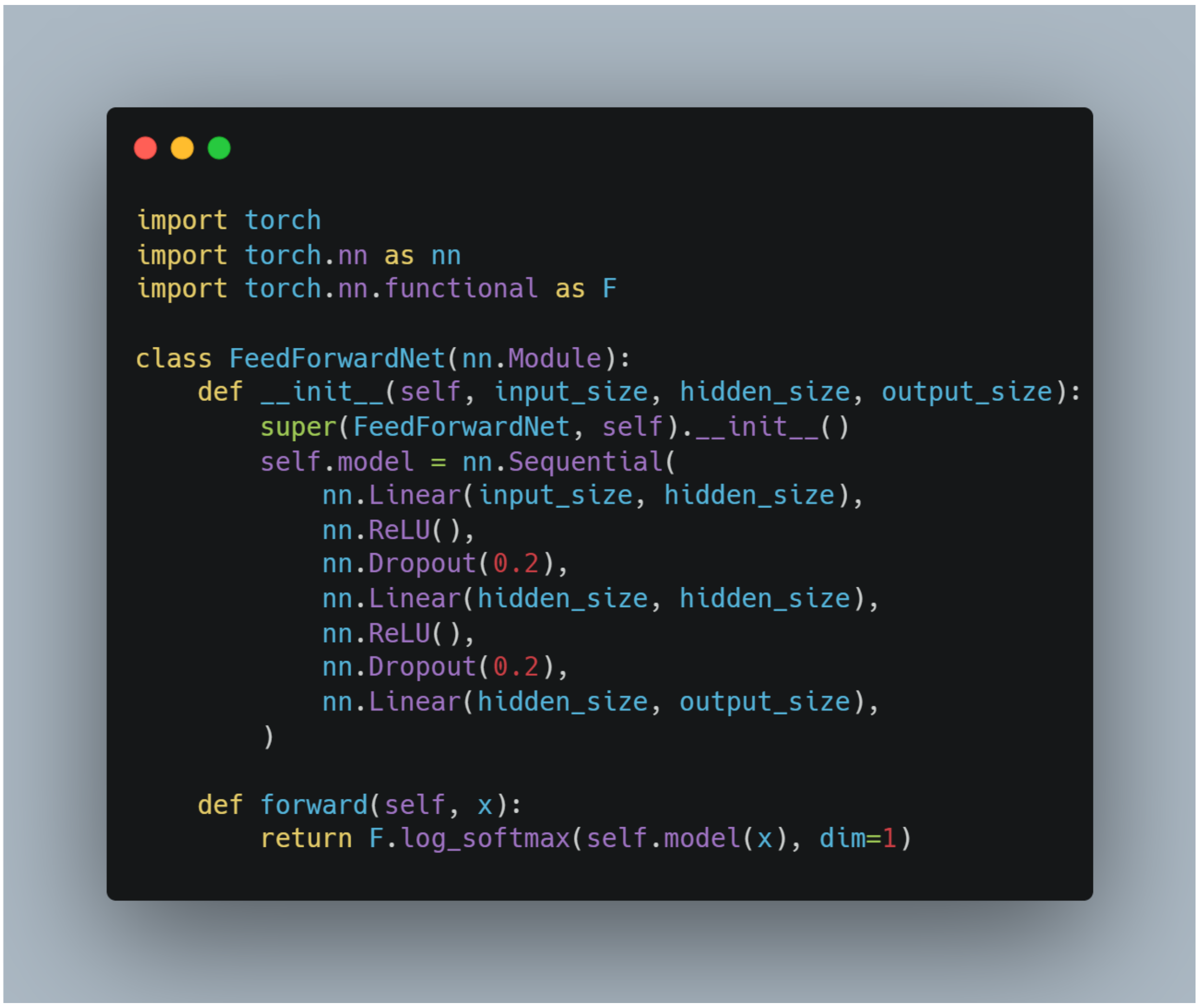
This module processes the medical dataset by filling missing values, encoding diseases, and scaling symptom data for use in the FFN.



```
import pandas as pd
from sklearn.preprocessing import LabelEncoder, StandardScaler

# Load dataset
data = pd.read_csv("AI_SET_3.csv")

# Handle missing values
data.fillna(0, inplace=True)

# Encode disease labels
le = LabelEncoder()
data["prognosis"] = le.fit_transform(data["prognosis"])

# Scale features
scaler = StandardScaler()
features = scaler.fit_transform(data.drop("prognosis", axis=1))
labels = data["prognosis"]
= data["prognosis"]

  return go(f, seed, [])
}
```

2. Feedforward Neural Network (FFN):

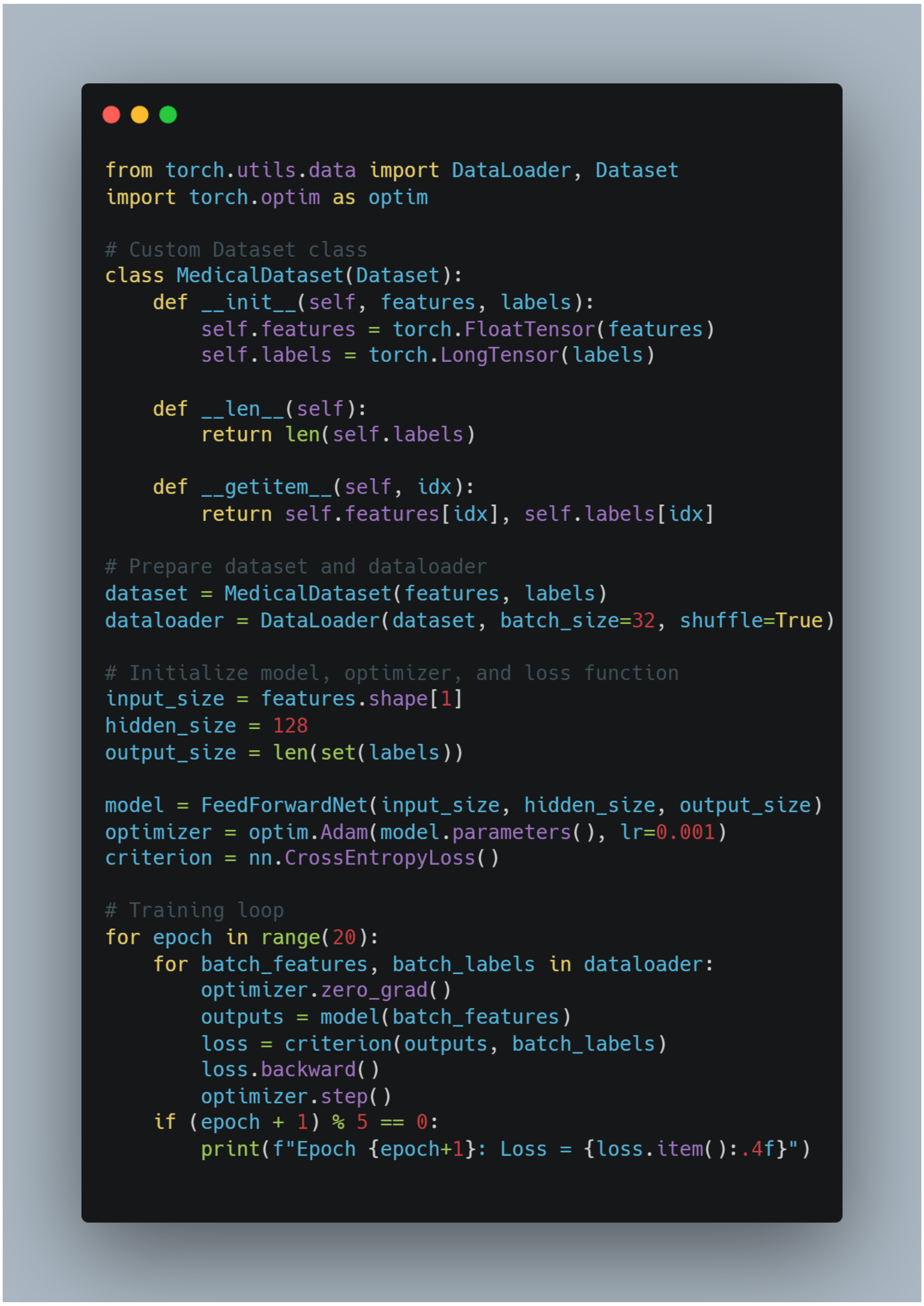
The FFN is the core model used for disease prediction. It consists of multiple dense layers with ReLU activation and dropout regularization.



```python
import torch
import torch.nn as nn
import torch.nn.functional as F

class FeedForwardNet(nn.Module):
    def __init__(self, input_size, hidden_size, output_size):
        super(FeedForwardNet, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(input_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(hidden_size, hidden_size),
            nn.ReLU(),
            nn.Dropout(0.2),
            nn.Linear(hidden_size, output_size),
        )

    def forward(self, x):
        return F.log_softmax(self.model(x), dim=1)
```

3. Model Training:

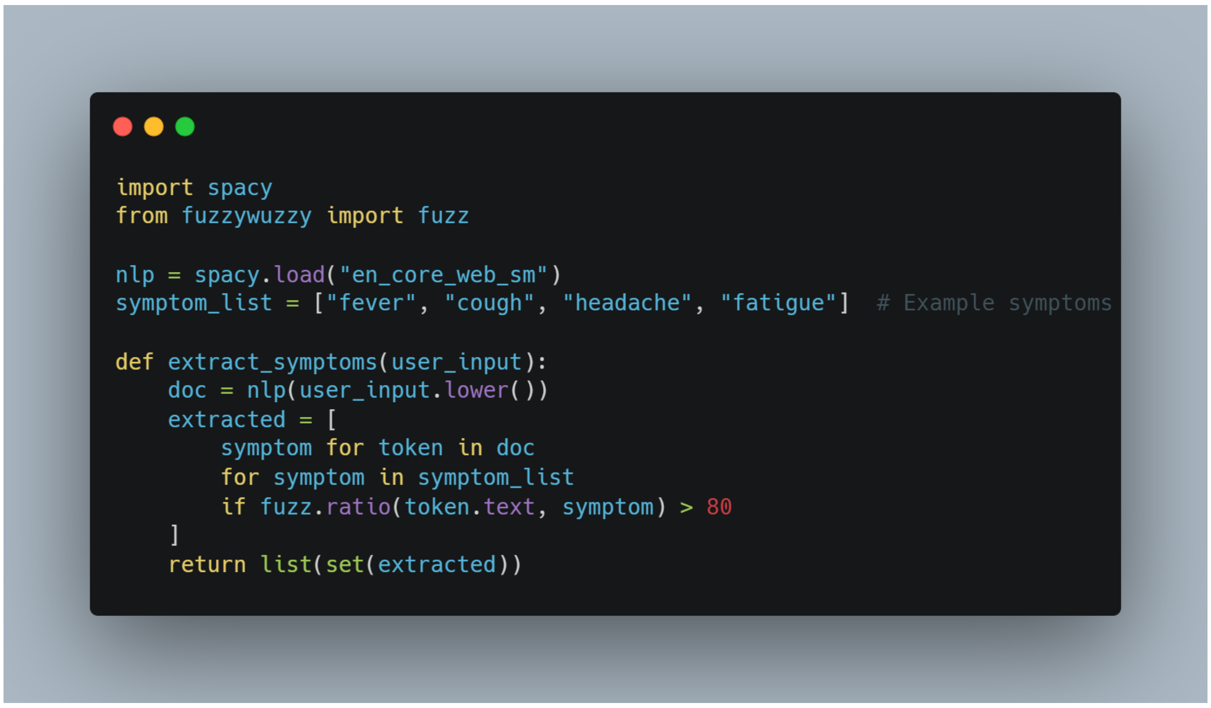
The model is trained using cross-entropy loss and the Adam optimizer.



```python
from torch.utils.data import DataLoader, Dataset
import torch.optim as optim

# Custom Dataset class
class MedicalDataset(Dataset):
    def __init__(self, features, labels):
        self.features = torch.FloatTensor(features)
        self.labels = torch.LongTensor(labels)

    def __len__(self):
        return len(self.labels)

    def __getitem__(self, idx):
        return self.features[idx], self.labels[idx]

# Prepare dataset and dataloader
dataset = MedicalDataset(features, labels)
dataloader = DataLoader(dataset, batch_size=32, shuffle=True)

# Initialize model, optimizer, and loss function
input_size = features.shape[1]
hidden_size = 128
output_size = len(set(labels))

model = FeedForwardNet(input_size, hidden_size, output_size)
optimizer = optim.Adam(model.parameters(), lr=0.001)
criterion = nn.CrossEntropyLoss()

# Training loop
for epoch in range(20):
    for batch_features, batch_labels in dataloader:
        optimizer.zero_grad()
        outputs = model(batch_features)
        loss = criterion(outputs, batch_labels)
        loss.backward()
        optimizer.step()
    if (epoch + 1) % 5 == 0:
        print(f"Epoch {epoch+1}: Loss = {loss.item():.4f}")
```

4. Symptom Extraction and Matching:

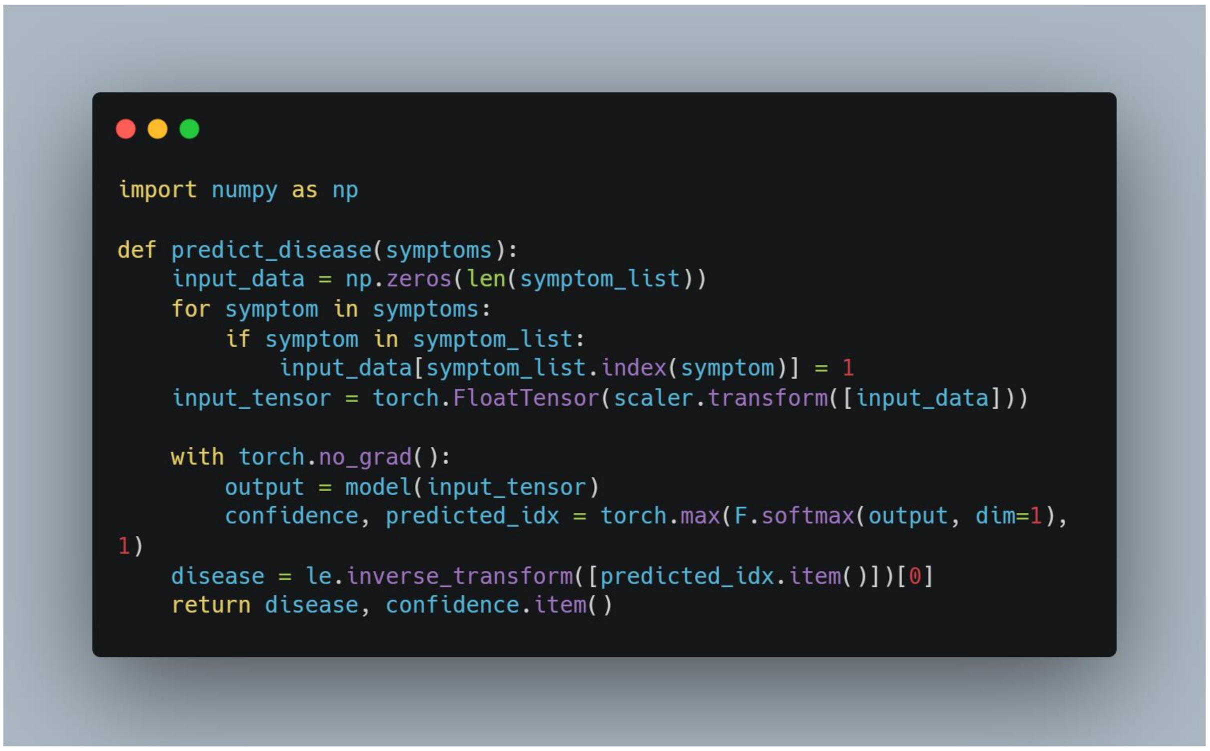
NLP techniques are used to extract symptoms from user input and match them with predefined symptoms.



```python
import spacy
from fuzzywuzzy import fuzz

nlp = spacy.load("en_core_web_sm")
symptom_list = ["fever", "cough", "headache", "fatigue"]  # Example symptoms

def extract_symptoms(user_input):
    doc = nlp(user_input.lower())
    extracted = [
        symptom for token in doc
        for symptom in symptom_list
        if fuzz.ratio(token.text, symptom) > 80
    ]
    return list(set(extracted))
```

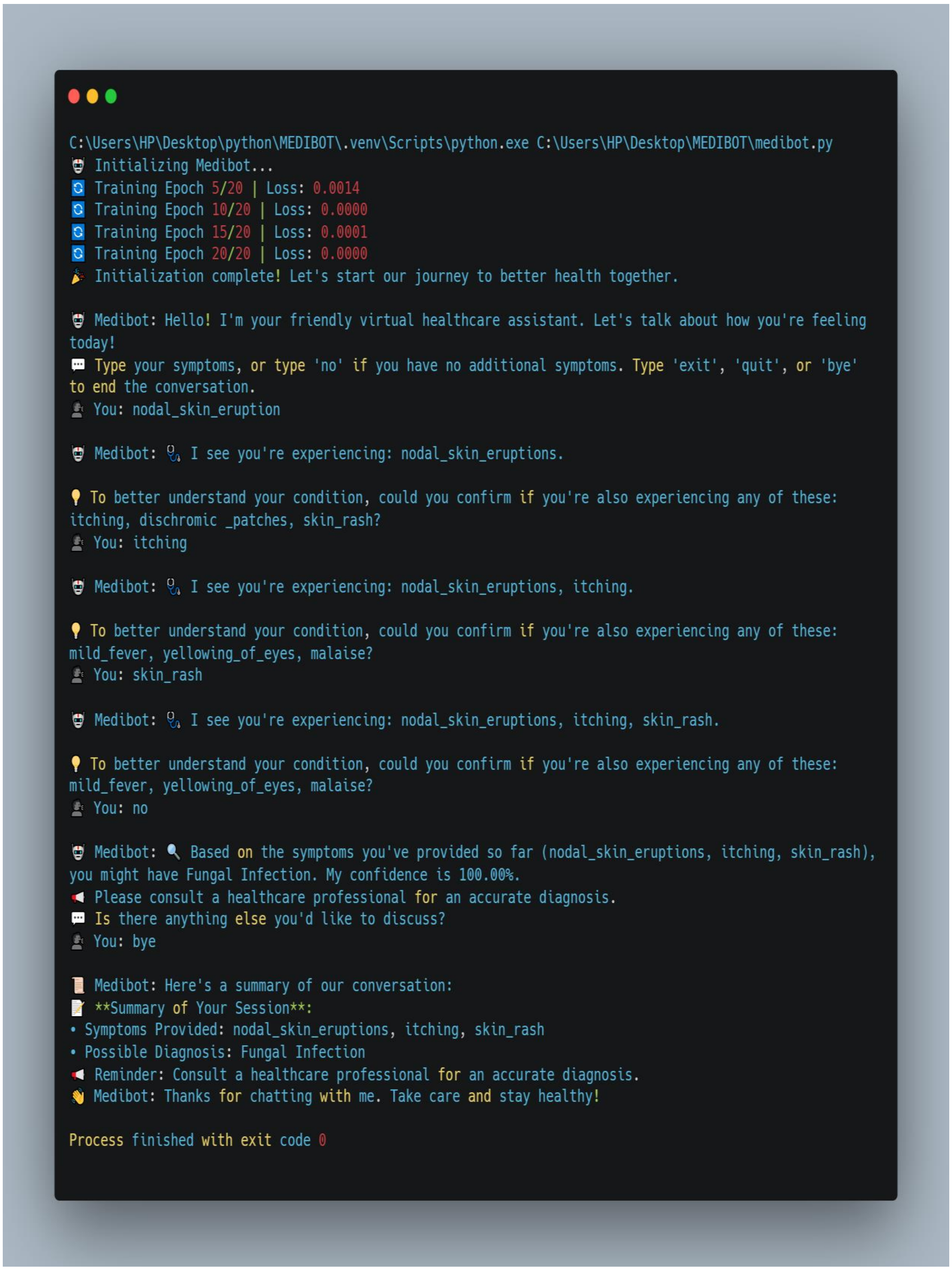
5. Disease Prediction:

Based on the extracted symptoms, the FFN predicts the most likely diseases.



```python
import numpy as np

def predict_disease(symptoms):
    input_data = np.zeros(len(symptom_list))
    for symptom in symptoms:
        if symptom in symptom_list:
            input_data[symptom_list.index(symptom)] = 1
    input_tensor = torch.FloatTensor(scaler.transform([input_data]))

    with torch.no_grad():
        output = model(input_tensor)
        confidence, predicted_idx = torch.max(F.softmax(output, dim=1),
1)
    disease = le.inverse_transform([predicted_idx.item()])[0]
    return disease, confidence.item()
```

Simulation Results:



```
C:\Users\HP\Desktop\python\MEDIBOT\.venv\Scripts\python.exe C:\Users\HP\Desktop\MEDIBOT\medibot.py
🤖 Initializing Medibot...
🔄 Training Epoch 5/20 | Loss: 0.0014
🔄 Training Epoch 10/20 | Loss: 0.0000
🔄 Training Epoch 15/20 | Loss: 0.0001
🔄 Training Epoch 20/20 | Loss: 0.0000
🎉 Initialization complete! Let's start our journey to better health together.

🤖 Medibot: Hello! I'm your friendly virtual healthcare assistant. Let's talk about how you're feeling
today!
💬 Type your symptoms, or type 'no' if you have no additional symptoms. Type 'exit', 'quit', or 'bye'
to end the conversation.
👤 You: nodal_skin_eruption

🤖 Medibot: 🩺 I see you're experiencing: nodal_skin_eruptions.

💡 To better understand your condition, could you confirm if you're also experiencing any of these:
itching, dischromic _patches, skin_rash?
👤 You: itching

🤖 Medibot: 🩺 I see you're experiencing: nodal_skin_eruptions, itching.

💡 To better understand your condition, could you confirm if you're also experiencing any of these:
mild_fever, yellowing_of_eyes, malaise?
👤 You: skin_rash

🤖 Medibot: 🩺 I see you're experiencing: nodal_skin_eruptions, itching, skin_rash.

💡 To better understand your condition, could you confirm if you're also experiencing any of these:
mild_fever, yellowing_of_eyes, malaise?
👤 You: no

🤖 Medibot: 🔍 Based on the symptoms you've provided so far (nodal_skin_eruptions, itching, skin_rash),
you might have Fungal Infection. My confidence is 100.00%.
📢 Please consult a healthcare professional for an accurate diagnosis.
💬 Is there anything else you'd like to discuss?
👤 You: bye

📜 Medibot: Here's a summary of our conversation:
📝 **Summary of Your Session**:
• Symptoms Provided: nodal_skin_eruptions, itching, skin_rash
• Possible Diagnosis: Fungal Infection
📢 Reminder: Consult a healthcare professional for an accurate diagnosis.
👋 Medibot: Thanks for chatting with me. Take care and stay healthy!

Process finished with exit code 0
```

Conclusion:

The Medibot project successfully demonstrates the potential of Artificial Intelligence in addressing healthcare accessibility challenges. By integrating Feedforward Neural Networks (FFNs) and Natural Language Processing (NLP), the system provides a seamless and user-friendly solution for identifying potential diseases based on symptoms.

The key achievements of the project include:

1. Development of a robust disease prediction model with high accuracy.
2. Implementation of an NLP-based symptom extraction mechanism, allowing natural language interaction.
3. A modular and scalable design, enabling future integration of additional symptoms and diseases.

Medibot empowers individuals by offering a preliminary diagnostic tool that bridges the gap between patients and healthcare professionals. It not only reduces the burden on medical practitioners but also encourages proactive healthcare management among users.

Future Scope:
To enhance the effectiveness of Medibot, the following improvements can be explored:

1. Expanding the dataset to include a broader range of diseases and symptoms.
2. Enhancing the NLP capabilities for more nuanced symptom extraction and conversation.
3. Incorporating a mobile application to improve accessibility further.

Medibot showcases how AI-driven tools can positively transform healthcare delivery, fostering a more inclusive and efficient approach to managing health.