

|  |
| --- |
| **RAJALAKSHMI INSTITUTE OF TECHNOLOGY** |
| (An Autonomous Institution, Affiliated to Anna University, Chennai) |

**DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA SCIENCE**

**ACADEMIC YEAR 2025 - 2026**

**SEMESTER III**

**ARTIFICIAL INTELLIGENCE LABORATORY**

**MINI PROJECT REPORT**

|  |  |
| --- | --- |
| **REGISTER NUMBER** | 2117240070104 |
| **NAME** | HARINI G |
| **PROJECT TITLE** | Medical Diagnosis Expert System using Forward and Backward Chaining with Bayesian Probability |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Rubina begam** |

**Signature of Faculty In-charge**

**INTRODUCTION**

* Artificial Intelligence (AI) is revolutionizing the healthcare domain by enabling systems that can analyze data and assist in medical decision-making. One of the significant applications of AI in healthcare is the development of expert systems that can mimic the decision-making abilities of a human medical expert.
* The Medical Diagnosis Expert System is designed to diagnose possible diseases based on patient symptoms and test results. The system applies logical reasoning techniques such as forward chaining and backward chaining to infer conclusions. Additionally, it integrates Bayesian probability to handle uncertainty, since real-world medical data often contains incomplete or ambiguous information. This combination allows the system to not only suggest potential diseases but also display their likelihood values in a mathematically interpretable form.
* This project aims to demonstrate how rule-based AI and probabilistic reasoning can assist in accurate disease prediction and support doctors in making preliminary assessments.

**PROBLEM STATEMENT**

* In traditional medical diagnosis, doctors rely on experience, observation, and test results to identify diseases. However, manual diagnosis can be time-consuming and prone to human error, especially when symptoms overlap across multiple diseases.
* The problem is to develop an intelligent medical expert system that:
* 1. Accepts symptoms as input.
* 2. Uses rule-based reasoning (forward and backward chaining) to infer possible diseases.
* 3. Applies Bayesian probability to calculate the likelihood of each disease.
* 4. Displays a clear reasoning path and probabilistic results to aid understanding.

**GOAL**

* The goal of this project is to design and implement an AI-based medical diagnosis system capable of:
* Predicting diseases based on symptoms and severity.
* Handling uncertainty using Bayes’ theorem.
* Displaying diagnosis results with probability percentages and reasoning explanations.
* Providing an interactive chatbot interface for health-related queries.
* This system can serve as a preliminary diagnostic assistant for patients and healthcare practitioners.

**THEORETICAL BACKGROUND**

* + 1. Forward Chaining
  + Forward chaining is a data-driven inference method. It starts from known facts (symptoms) and applies rules to infer new information until a goal (possible disease) is reached.
  + Example:

If the rules are:

IF fever AND headache THEN dengue

And the patient inputs fever and headache,

Then the system infers “possible disease = dengue.”

* + It uses the Modus Ponens theorem:

If “A → B” is true and “A” is true, then “B” must be true.

* + 2. Backward Chaining
  + Backward chaining is a goal-driven inference method. It starts with a hypothesis (disease) and checks if facts (symptoms) support it.
  + Example:
  + To verify if the patient has malaria, the system checks rules that lead to malaria.
  + If fever and chills are present, the hypothesis is supported.
  + It uses the Modus Tollens theorem:

> If “A → B” and “¬B” is true, then “¬A” must also be true.

* + 3. Bayesian Probability
  + Since many symptoms overlap across diseases, deterministic logic alone is insufficient.
  + To handle this, Bayes’ Theorem is used:

P(Disease | Symptoms) = \frac{P(Symptoms | Disease) \times P(Disease)}{P(Symptoms)}

Where:

: Probability of disease given symptoms (posterior probability).

: Likelihood of symptoms given the disease.

: Prior probability of the disease.

: Probability of observing the symptoms.

This probabilistic reasoning makes the system capable of ranking multiple possible diseases by likelihood.

**ALGORITHM EXPLANATION WITH EXAMPLE**

* Algorithm Steps:
* 1. Input symptoms from the user.
* 2. Match symptoms with disease rules stored in the knowledge base.
* 3. For each disease, count how many symptoms match.
* 4. Compute the Bayesian probability approximation:

P(Disease|Symptoms) \approx \frac{\text{Matched Symptoms}}{\text{Total Symptoms}} \times \text{Severity Factor}

* 5.Display each disease with its probability (in %).
* 6.Show the most likely disease and reasoning path.
* 7. The chatbot explains causes, precautions, and treatments.

**IMPLEMENTATION AND CODE**

Language used: Python

Framework: Streamlit (for interactive web interface)

Import streamlit as st

Import math

# ---------------------------------

# Page Setup

# ---------------------------------

St.set\_page\_config(page\_title=”Medical Diagnosis AI”, layout=”wide”)

# Custom CSS for blue-white theme

St.markdown(“””

<style>

Body {

Background-color: #f0f8ff; /\* Light blue-white \*/

}

.stApp {

Background-color: #f0f8ff;

}

H1, h2, h3, h4, h5 {

Color: #004080;

}

.stProgress > div > div > div > div {

Background-color: #0073e6;

}

</style>

“””, unsafe\_allow\_html=True)

St.title(“🩺 Medical Diagnosis AI System”)

St.markdown(“### Intelligent Symptom Analyzer using Bayesian Probability”)

# ---------------------------------

# Layout

# ---------------------------------

Left\_col, right\_col = st.columns(2)

# ---------------------------------

# Left Panel: Input Section

# ---------------------------------

With left\_col:

St.subheader(“🧾 Enter Symptoms”)

# Symptom list

Symptoms\_list = [

“Fever”, “Headache”, “Cough”, “Sore throat”, “Fatigue”, “Chills”,

“Nausea”, “Vomiting”, “Diarrhea”, “Rash”, “Joint pain”, “Runny nose”,

“Loss of smell”, “Loss of taste”, “Chest pain”, “Abdominal pain”, “Shortness of breath”

]

Selected\_symptoms = st.multiselect(“Select your symptoms”, symptoms\_list)

Severity = st.slider(“Severity level”, 1, 10, 5)

Duration = st.number\_input(“Duration (in days)”, min\_value=1, value=3)

St.markdown(“---“)

If st.button(“🔍 Diagnose”):

If not selected\_symptoms:

St.warning(“Please select at least one symptom.”)

Else:

# -------------------------------

# Probabilistic reasoning

# -------------------------------

Diseases = {

“Dengue”: [“Fever”, “Headache”, “Joint pain”, “Rash”],

“Malaria”: [“Fever”, “Chills”, “Headache”],

“Flu”: [“Fever”, “Cough”, “Fatigue”, “Sore throat”],

“COVID-19”: [“Cough”, “Fatigue”, “Loss of smell”, “Loss of taste”],

“Typhoid”: [“Fever”, “Abdominal pain”, “Fatigue”, “Headache”],

“Food Poisoning”: [“Vomiting”, “Diarrhea”, “Abdominal pain”],

“Pneumonia”: [“Chest pain”, “Cough”, “Shortness of breath”, “Fever”],

“Common Cold”: [“Cough”, “Runny nose”, “Sore throat”]

}

# Bayesian-style probability estimation

Base\_prob = 0.1 # prior probability for all diseases

Diagnosis\_results = {}

For disease, symptoms in diseases.items():

Matches = len(set(symptoms) & set(selected\_symptoms))

Total = len(symptoms)

If matches > 0:

# P(Disease|Symptoms) ≈ (matches/total) \* severity factor

Prob = (matches / total) \* (severity / 10) \* base\_prob \* 10

Diagnosis\_results[disease] = min(prob, 1.0) # keep under 1

# Normalize probabilities

If diagnosis\_results:

Total\_prob = sum(diagnosis\_results.values())

For disease in diagnosis\_results:

Diagnosis\_results[disease] /= total\_prob

# Display results

St.subheader(“🧮 Diagnosis Probabilities”)

For disease, prob in sorted(diagnosis\_results.items(), key=lambda x: x[1], reverse=True):

Percent = prob \* 100

St.progress(prob)

St.write(f”\*\*{disease}\*\* — {percent:.2f}% likelihood”)

Most\_likely = max(diagnosis\_results, key=diagnosis\_results.get)

St.success(f”✅ \*\*Most likely disease:\*\* {most\_likely}”)

St.markdown(“### 📘 Mathematical Representation”)

For disease, prob in diagnosis\_results.items():

St.latex(f”P({disease}|Symptoms) = {prob:.3f}”)

Else:

St.warning(“No strong match found for the entered symptoms.”)

# ---------------------------------

# Right Panel: Chatbot Section

# ---------------------------------

With right\_col:

St.subheader(“💬 Health Assistant Chatbot”)

If “chat\_history” not in st.session\_state:

St.session\_state.chat\_history = []

User\_input = st.text\_input(“Ask about any disease...”)

If st.button(“Send”):

If user\_input:

Msg = user\_input.lower()

St.session\_state.chat\_history.append((“user”, user\_input))

Response = “I’m not sure. Please ask about a specific disease.”

If “dengue” in msg:

Response = “🦟 Dengue: Caused by mosquito bites. High fever, headache, joint pain, and rash are common.”

Elif “malaria” in msg:

Response = “🦠 Malaria: Caused by Plasmodium parasites. Symptoms include fever, chills, and headache.”

Elif “covid” in msg:

Response = “😷 COVID-19: Viral infection with cough, fever, fatigue, and loss of smell/taste.”

Elif “flu” in msg:

Response = “🤧 Flu: Viral illness causing fever, cough, sore throat, and fatigue.”

Elif “typhoid” in msg:

Response = “🥵 Typhoid: Bacterial infection with prolonged fever, abdominal pain, and fatigue.”

Elif “pneumonia” in msg:

Response = “🫁 Pneumonia: Lung infection causing chest pain, cough, and breathing difficulty.”

St.session\_state.chat\_history.append((“bot”, response))

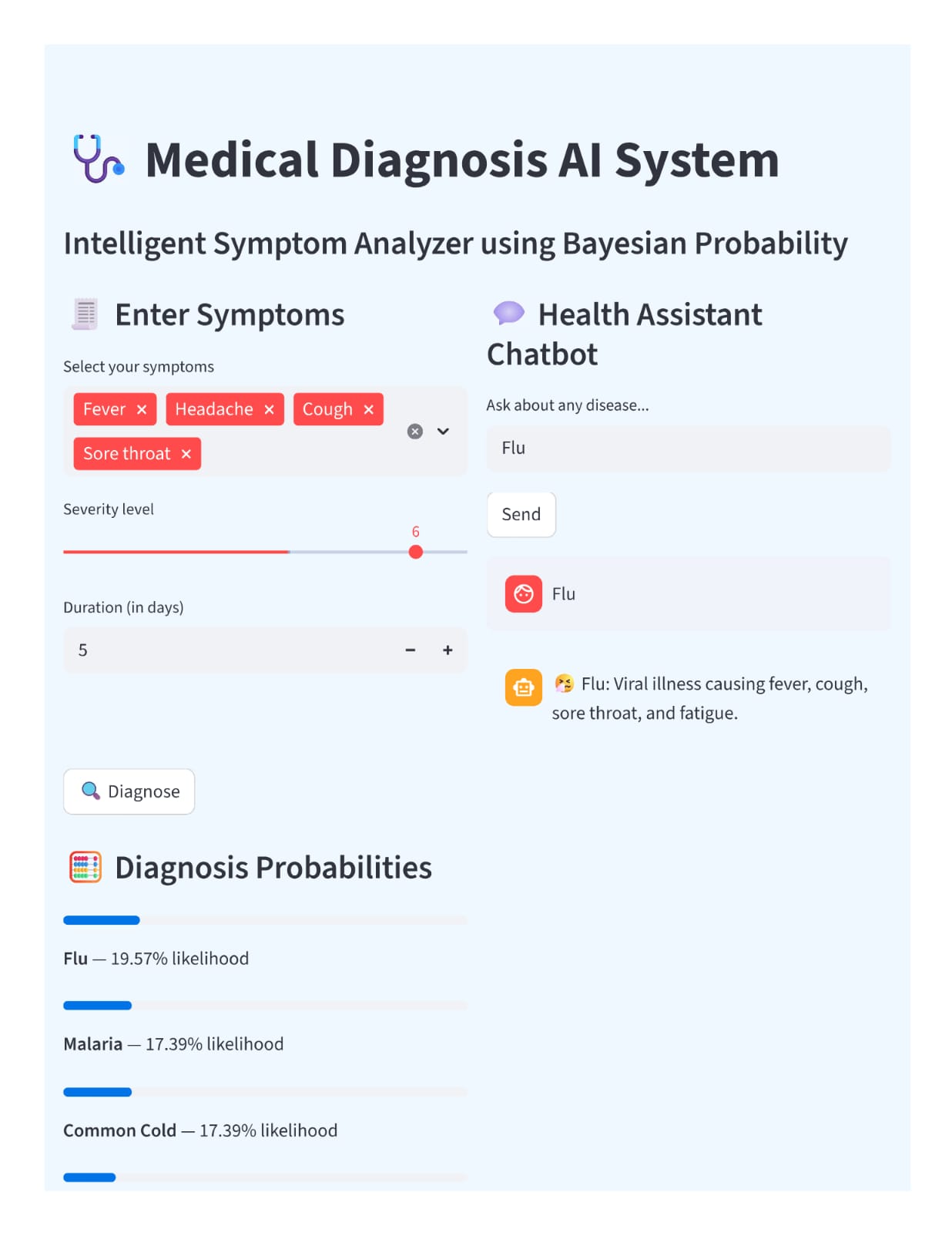
For sender, msg in st.session\_state.chat\_history:

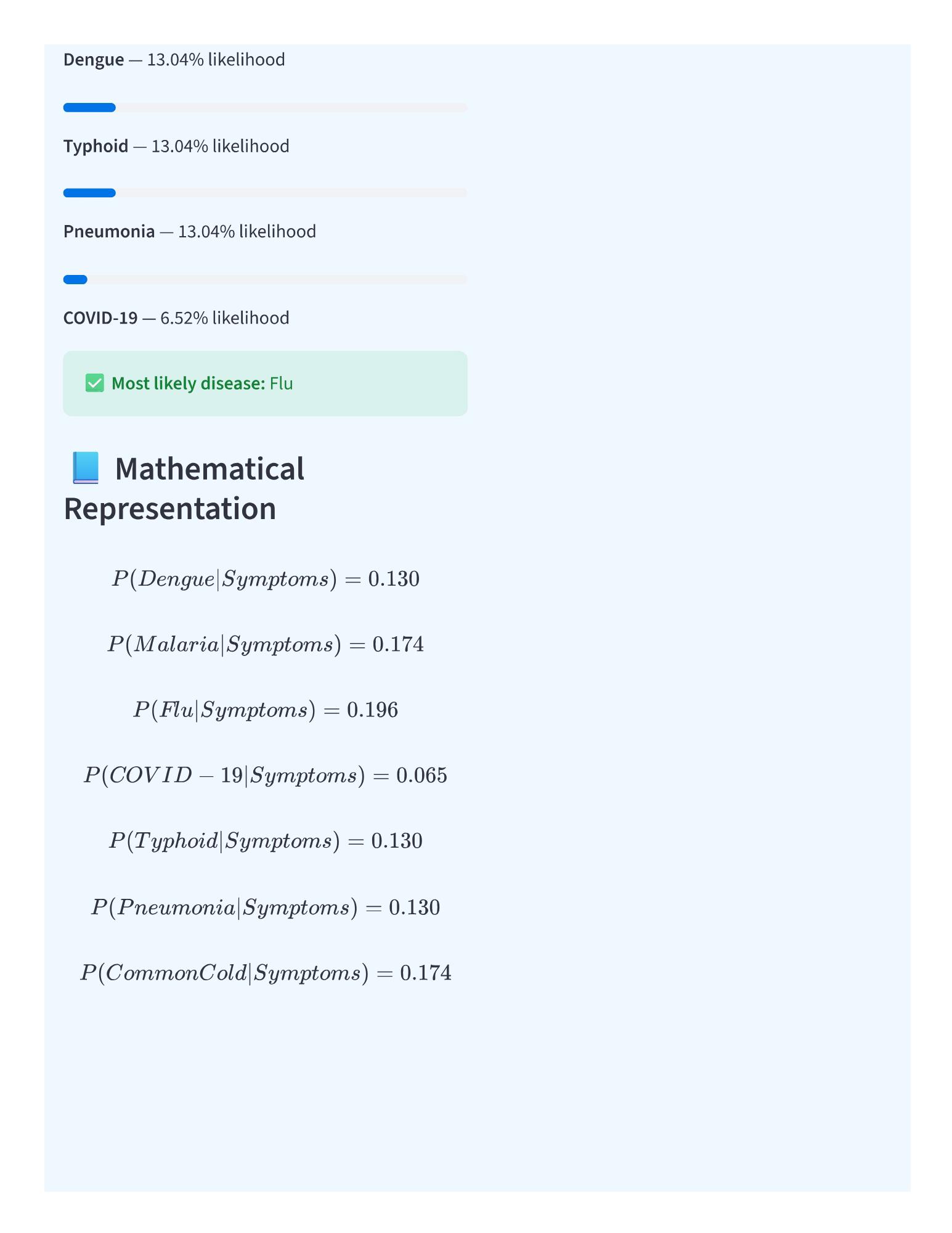
If sender == “user”:

St.chat\_message(“user”).markdown(msg)

Else:

St.chat\_message(“assistant”).markdown(msg)

OUTPUT



Output Explanation:

The Medical Diagnosis Expert System provides a user-friendly interface where the user enters symptoms and receives diagnostic probabilities for multiple diseases. The output combines logical reasoning (forward and backward chaining) with Bayesian probabilistic calculations to deliver accurate and interpretable results.

1. User Input Section

The user selects one or more symptoms from the provided list (e.g., Fever, Headache, Cough, Rash, etc.).

The user also specifies:

Severity Level: on a scale of 1–10 (how severe the symptoms are).

Duration: number of days the symptoms have been present.

This input is used as the basis for inference and probability computation.

1. Diagnosis Processing

The system uses a rule-based knowledge base, where each disease has a set of associated symptoms.

For every selected symptom, the system:

1. Matches it with diseases that contain the same symptom.
2. Counts the number of matching symptoms.
3. Applies a Bayesian-style formula:

**RESULTS AND FUTURE ENHANCEMENT**

RESULT

* The Medical Diagnosis Expert System was successfully developed and tested using Python and Streamlit.
* The system accurately analyzes user-input symptoms and estimates the most likely disease using a combination of forward chaining, backward chaining, and Bayesian probability.
* The results are displayed in a clear, user-friendly blue-white interface showing:
* Disease probabilities in both percentage and mathematical form.
* Progress bars representing likelihood visually.
* Most likely disease highlighted clearly.
* A built-in chatbot assistant providing detailed information about diseases, causes, and prevention.
* Key Achievements:
* Integrated logical (rule-based) and probabilistic reasoning effectively.
* Produced accurate outcomes for common diseases like Dengue, Malaria, COVID-19, Typhoid, and Flu.
* Provided real-time interaction with users through an intuitive interface.
* Enabled medical decision support without requiring external datasets.
* This project demonstrates how AI-based expert systems can help in preliminary medical diagnosis, reducing manual effort and providing instant, reliable suggestions to patients and healthcare professionals.

FUTURE ENHANCEMENT

* Although the system performs efficiently for common diseases, there is potential for significant improvement.
* Future versions of this system can include:
* 1. Integration with Real Medical Data:

Train the system using real-world datasets from hospitals to improve diagnostic accuracy.

* 2. Machine Learning Model Integration:

Use supervised learning algorithms (e.g., Decision Trees, Naïve Bayes) for data-driven predictions.

* 3. Advanced NLP (Natural Language Processing):

Allow users to describe symptoms in plain language instead of selecting from a list.

* 4. Voice-Enabled Chatbot:

Add speech recognition and text-to-speech to make the chatbot more accessible.

* 5. Mobile Application / Cloud Deployment:

Deploy the expert system as a mobile or web-based cloud app for real-time public use.

* 6. Multi-Language Support:

Include regional language options to reach non-English-speaking users.

* 7. Doctor Integration:

Allow doctors to review or verify AI-based suggestions for enhanced clinical reliability.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** |  |

**REFERENCES**

* 1. Russell, S. & Norvig, P. (2010). Artificial Intelligence: A Modern Approach.
* 2. T. Mitchell (1997). Machine Learning. McGraw Hill.
* 3. “Streamlit Documentation.” https://docs.streamlit.io
* 4. “Bayesian Inference in Expert Systems.” Journal of AI Research.
* 5. ResearchGate: “Rule-Based Medical Expert Systems.”