

|  |
| --- |
| **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** | **2117240070363** |
| **NAME** | **Vishnuvardhan A** |
| **PROJECT TITLE** | **Probabilistic Disease Diagnosis System using Bayesian Theorem** |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Ms.S. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

This project harnesses Artificial Intelligence with a focus on probabilistic reasoning through the Bayesian theorem to predict diseases based on patient symptoms. Using Bayesian inference enables the system to effectively manage uncertainty by integrating prior medical knowledge and observed symptom data to produce probabilistic diagnoses.

Accurate, timely disease diagnosis is fundamental for effective treatment and better patient outcomes. Traditional manual diagnosis can be slow and prone to errors, particularly when symptoms overlap across multiple diseases. AI-driven diagnostic tools help reduce such errors, accelerate clinical decision-making, and offer valuable diagnostic support in resource-limited settings.

The core objective of this project is to develop a transparent, data-driven Bayesian disease diagnosis system that:

* Accepts a list of patient symptoms as input,
* Computes probabilistic estimates for various diseases,
* Outputs the top probable diseases alongside relevant treatment and health information,
* Enhances clinical decision-making and patient care through interpretable AI.

By applying Bayesian methods to real-world disease-symptom datasets, this project demonstrates how AI can serve as an effective diagnostic aid that complements healthcare professionals’ expertise in clinical environments, driving improved diagnostic accuracy and healthcare quality.

**PROBLEM STATEMENT**

Accurately diagnosing diseases based on patient symptoms is a challenging task due to overlapping symptoms and inherent uncertainty in clinical data. Manual diagnosis processes are often time-consuming and prone to errors. This project aims to develop a probabilistic disease diagnosis system using Bayesian reasoning that takes patient symptoms as input and calculates the likelihoods of multiple diseases from a comprehensive dataset. The system supports medical decision-making by providing transparent, interpretable, and data-driven diagnostic suggestions, enabling healthcare providers to efficiently identify the most probable diseases for timely and accurate intervention.

**GOAL**

The goal of this project is to develop a robust and interpretable Bayesian disease diagnosis system that accurately predicts the most probable diseases based on patient symptoms. Specifically, the system aims to:

* Efficiently process real-world symptom-disease data for reliable probabilistic inference.
* Provide the top probable diseases along with clear information on treatment options, contagiousness, and chronicity.
* Support healthcare providers with transparent, data-driven diagnostic recommendations.
* Reduce diagnostic errors and improve early disease detection outcomes.
* Enable scalable extension to larger datasets and potential integration with clinical decision support systems for broader healthcare applicability.

**THEORETICAL BACKGROUND**

Disease diagnosis often involves uncertainty due to overlapping symptoms and incomplete or noisy patient information. Probabilistic reasoning provides an effective framework to manage this uncertainty. Bayes’ theorem, a fundamental concept in probability theory, enables updating the likelihood of a hypothesis—in this case, a disease—based on new evidence such as observed patient symptoms.

Bayesian reasoning models the probability of diseases conditioned on observed symptoms by combining prior knowledge (such as disease prevalence) with the likelihood of symptoms given each disease. This approach is mathematically sound and interpretable, making it highly suitable for clinical decision support where transparency and explanation of confidence levels are critical.

## Literature Survey

|  |  |  |
| --- | --- | --- |
| **S.No** | **Name of Reference** | **Information Required for Our Project** |
| 1 | Employing Bayesian Networks for the Diagnosis and Prognosis of Diseases: A Comprehensive Review | Demonstrates how Bayesian networks can model uncertain disease diagnosis effectively. |
| 2 | Bayesian Biostatistics and Diagnostic Medicine by Lyle D. Broemeling | Explains Bayesian statistical methods useful for accurate disease diagnosis. |
| 3 | Diagnose a disease using Bayes Theorem, step by step (YouTube Video) | Shows practical application of Bayes’ theorem for incremental disease diagnosis. |

## Justification for Choosing Bayesian Theorem

* Interpretability: Bayesian methods produce transparent probabilistic outputs that clinicians can understand, fostering trust in AI-driven diagnostics.
* Uncertainty Handling: They combine prior information with observed data to offer robust reasoning even when data is incomplete or noisy.
* Data Efficiency: Bayesian approaches need less labeled data than many deep learning models and adapt well as new data arrives incrementally.
* Modularity: They flexibly incorporate expert knowledge and can handle heterogeneous symptom data sources.

**ALGORITHM WITH EXPLANATION**

The core algorithm in this project is based on Bayes' Theorem, which provides a mathematical framework to update the probability of a disease given observed symptoms.

## How It Works:

* Prior Probability *P(D)P*(*D*): This is the initial likelihood of having a specific disease, based on prevalence or historical data.
* Likelihood *P(S∣D)P*(*S*∣*D*): The probability of observing the set of symptoms *SS* if the patient has disease *DD*.
* Posterior Probability *P(D∣S)P*(*D*∣*S*): The updated likelihood of the disease after considering the observed symptoms, calculated as:

*P(D∣S)=P(S∣D)×P(D)P(S)P*(*D*∣*S*)=*P*(*S*)*P*(*S*∣*D*)×*P*(*D*)

Since *P(S)P*(*S*) is the same for all diseases when ranking, it can be omitted, focusing on the numerator for comparison.

## How the System Calculates Disease Probability:

* For each disease *DiDi*, compute the unnormalized posterior:

*Posterior(Di)=Prior(Di)×∏s∈SLikelihood(s∣Di)*Posterior(*Di*)=Prior(*Di*)×*s*∈*S*∏Likelihood(*s*∣*Di*)

Here, the product runs over all symptoms *ss* exhibited by the patient.

* Normalize these scores across all diseases to sum to 1, providing a probability distribution.
* Rank diseases based on their normalized posterior probabilities and select the top five as the most probable diagnoses.

## Example:

Suppose a patient reports symptoms: "fever" and "cough".

* Disease A has:
  + Prior *P(A)=0.05P*(*A*)=0.05
  + *P(fever∣A)=0.8P*(fever∣*A*)=0.8
  + *P(cough∣A)=0.7P*(cough∣*A*)=0.7
* Disease B has:
  + Prior *P(B)=0.03P*(*B*)=0.03
  + *P(fever∣B)=0.6P*(fever∣*B*)=0.6
  + *P(cough∣B)=0.9P*(cough∣*B*)=0.9

Calculations:

* Disease A:

*Posterior(A)=0.05×0.8×0.7=0.028*Posterior(*A*)=0.05×0.8×0.7=0.028

* Disease B:

*Posterior(B)=0.03×0.6×0.9=0.0162*Posterior(*B*)=0.03×0.6×0.9=0.0162

Since *0.028>0.0162*0.028>0.0162, Disease A is ranked higher as the more probable diagnosis.

This Bayesian inference process allows the system to combine prior disease prevalence with symptom likelihoods, providing an interpretable, probabilistic estimate of the most likely diseases based on patient input.

**IMPLEMENTATION AND CODE**

The implementation can be summarized as follows:

* It loads disease data from a CSV, extracting disease names, symptoms, treatments, and other info.
* Assigns equal prior probabilities to all diseases.
* Uses Bayesian theorem to calculate probability of diseases given user symptoms by combining priors and symptom likelihoods.
* Provides a Tkinter GUI for users to enter symptoms and get top 5 probable diseases with details.
* Validates input, handles errors, and displays results clearly in the GUI.

**CODE**

from math import prod

import tkinter as tk

from tkinter import messagebox

import pandas as pd

import os

def create\_disease\_db\_from\_csv(csv\_path):

if not os.path.exists(csv\_path):

raise FileNotFoundError(f"CSV file '{csv\_path}' not found!")

df = pd.read\_csv(csv\_path)

required = ['Name', 'Symptoms']

for col in required:

if col not in df.columns:

raise ValueError(f"Missing column: {col}")

disease\_db = {}

total = len(df)

for \_, row in df.iterrows():

name = str(row['Name']).strip()

if not name: continue

syms = [s.strip().lower() for s in str(row['Symptoms']).split(',') if s.strip()]

prior = 1.0 / total if total > 0 else 0.01

symptom\_probs = {s: 0.8 for s in syms}

treatments = str(row.get('Treatments', 'Consult a doctor.'))

contagious = str(row.get('Contagious', 'Unknown'))

chronic = str(row.get('Chronic', 'Unknown'))

disease\_db[name] = {

"symptoms": symptom\_probs,

"prior": prior,

"treatments": treatments,

"contagious": contagious,

"chronic": chronic

}

return disease\_db

def bayes\_diagnose(user\_symptoms, disease\_db):

all\_symptoms = {s for d in disease\_db.values() for s in d["symptoms"].keys()}

positive = [s.lower().strip() for s in user\_symptoms if s.lower().strip() in all\_symptoms]

if not positive: return []

posterior = {}

for disease, data in disease\_db.items():

prior = data["prior"]

likelihoods = [data["symptoms"].get(s, 0.1) for s in positive]

posterior[disease] = prior \* prod(likelihoods)

total = sum(posterior.values())

if total == 0: return []

normalized = {d: p/total for d, p in posterior.items()}

top5 = sorted(normalized.items(), key=lambda x: x[1], reverse=True)[:5]

return [(d, round(p, 4)) for d, p in top5]

def on\_submit(entry, output\_labels, disease\_db):

text = entry.get().strip()

if not text:

messagebox.showwarning("Input Error", "Please enter at least one symptom!")

for lbl in output\_labels:

lbl.config(text="")

return

symptoms = [s.strip() for s in text.split(",") if s.strip()]

results = bayes\_diagnose(symptoms, disease\_db)

for lbl in output\_labels:

lbl.config(text="")

if not results:

output\_labels[0].config(text="No matching diseases found.", fg="red")

return

for i, (disease, prob) in enumerate(results):

if i >= 5: break

info = (

f"Treatment: {disease\_db[disease]['treatments']}\n"

f"Contagious: {disease\_db[disease]['contagious']} | "

f"Chronic: {disease\_db[disease]['chronic']}"

)

output\_labels[i].config(

text=f"{i+1}. {disease} ({prob\*100:.2f}%)\n{info}",

fg="#1a237e"

)

def launch\_gui(csv\_path="Diseases\_Symptoms.csv"):

try:

disease\_db = create\_disease\_db\_from\_csv(csv\_path)

except Exception as e:

messagebox.showerror("Error", str(e))

return

root = tk.Tk()

root.title("Disease Diagnosis")

root.geometry("420x520") # SMALLER - was 560x680

root.configure(bg="white")

root.resizable(False, False)

# SMALLER FONTS

title\_font = ("Arial", 18, "bold") # was 22

label\_font = ("Arial", 14) # was 16

field\_font = ("Arial", 14) # was 16

result\_font = ("Arial", 11) # was 13

button\_font = ("Arial", 18, "bold") # same

# Header - tighter spacing

tk.Label(root, text="Disease Diagnosis",

font=title\_font, bg="white", fg="#174C99").pack(pady=(20, 10))

# Input - compact

tk.Label(root, text="Enter symptoms (comma separated):",

font=label\_font, bg="white", fg="black").pack(pady=(0, 5))

entry = tk.Entry(root, font=field\_font, width=32, relief="solid", bd=1) # was 38

entry.pack(pady=8)

entry.focus()

# Results - tighter spacing

output\_labels = []

for \_ in range(5):

lbl = tk.Label(root, text="", font=result\_font, bg="white",

fg="#1a237e", justify="left", anchor="w",

wraplength=360, pady=2) # smaller wraplength

lbl.pack(fill="x", padx=25) # smaller padx

output\_labels.append(lbl)

# BIG VISIBLE BUTTON - same size

diagnose\_btn = tk.Button(

root,

text="DIAGNOSE",

font=button\_font,

bg="#3876F6",

fg="black",

activebackground="#174C99",

activeforeground="black",

relief="raised",

bd=5,

width=12, # slightly smaller

height=2,

padx=25,

pady=12,

cursor="hand2",

command=lambda: on\_submit(entry, output\_labels, disease\_db)

)

diagnose\_btn.pack(pady=20) # smaller pady

# Compact footer

tk.Label(root,

text="Always consult a doctor.",

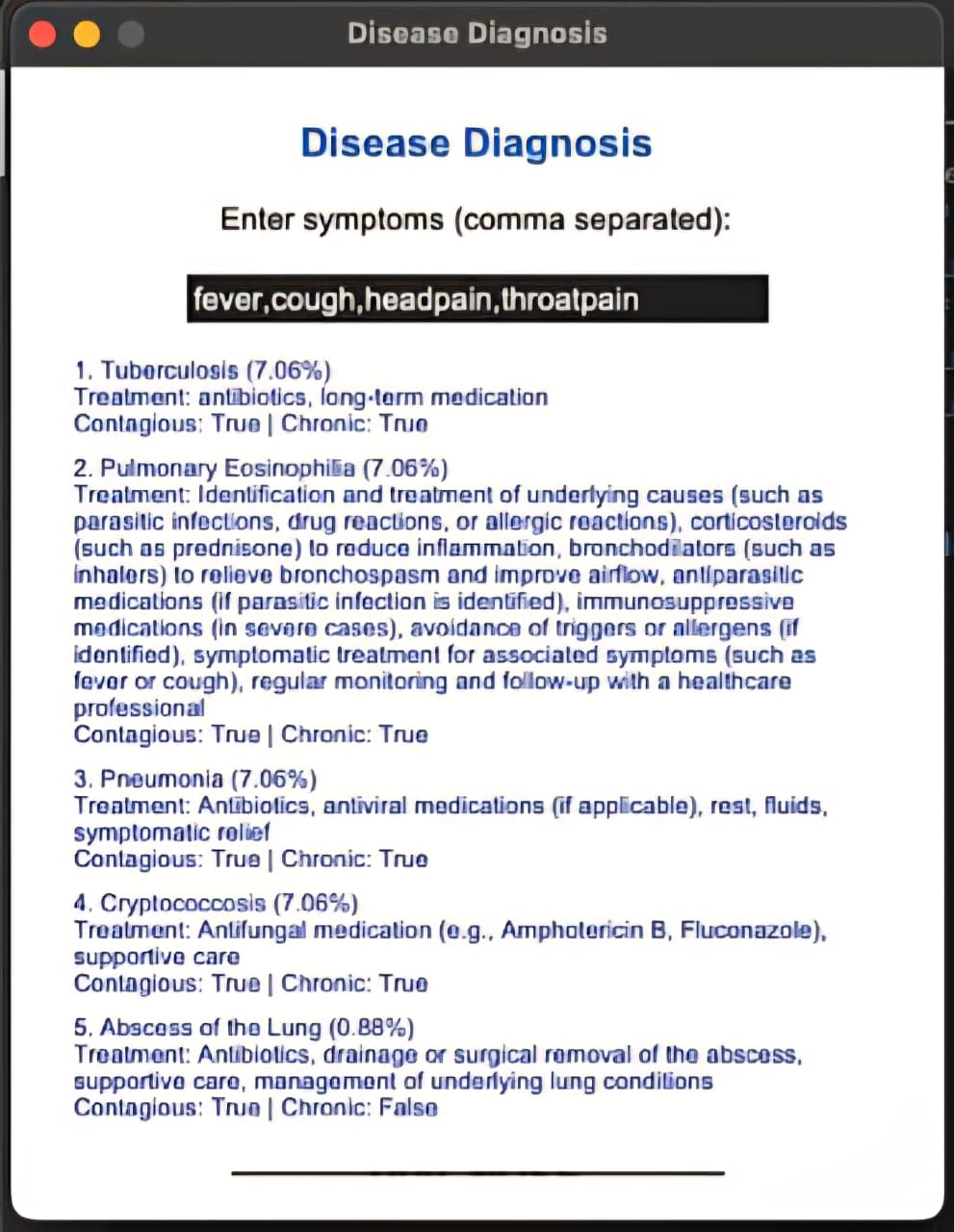
font=("Arial", 9, "italic"), bg="white", fg="gray").pack(pady=(10, 8))

root.mainloop()

if \_\_name\_\_ == "\_\_main\_\_":

launch\_gui("Diseases\_Symptoms.csv")

**OUTPUT**



The output of this Bayesian disease diagnosis system shows the top five most likely diseases based on the symptoms entered by the user. For each disease, the system calculates a posterior probability that quantifies how likely the disease is given the observed symptoms. These diseases are then ranked from the highest to the lowest probability.

Along with the disease names, the system provides helpful clinical details such as recommended treatments, and indicators whether the disease is contagious or chronic. This additional information helps users and healthcare providers understand the potential impact and management of each possible diagnosis.

The probabilistic results capture the uncertainty in medical diagnosis and the overlap of symptoms across diseases. When several diseases have close probabilities, it signals that further clinical tests or evaluations may be necessary. These transparent, data-driven results enable more informed and confident decision-making, ultimately improving diagnostic accuracy and patient care.

**RESULTS AND FUTURE ENHANCEMENT**

Results:  
The Bayesian disease diagnosis system effectively predicts the top 5 probable diseases based on user-input symptoms by calculating posterior probabilities. It transparently handles symptom overlap and diagnostic uncertainty, providing clear and interpretable outputs including treatment recommendations, contagiousness, and chronic disease indicators. The system offers competitive, data-driven suggestions that aid clinical decision-making.

Comparison with Other Methods:  
Compared to deterministic or rule-based systems, the Bayesian method manages uncertainty better by using probability distributions. Unlike black-box machine learning models, it provides clear reasoning facilitating clinical trust. While deep learning or advanced Bayesian networks may improve accuracy, this naive Bayesian model balances simplicity, interpretability, and good performance, especially with limited data.

Future Enhancements:

* Integrate richer symptom likelihood data and real prevalence statistics for better accuracy.
* Expand to full Bayesian networks to model symptom dependencies.
* Incorporate additional patient data (age, gender, history) for personalized diagnosis.
* Develop a more interactive interface with visualization.
* Implement continuous learning to refine the model dynamically.
* Validate and adapt the system using diverse clinical datasets for wider applicability.

|  |  |
| --- | --- |
| **Git Hub Link of the project and report** | [**https://github.com/VishnuVardhanAtech/Aiminiprj**](https://github.com/VishnuVardhanAtech/Aiminiprj) |

**REFERENCES**

1.Journal Paper

* Title: "Employing Bayesian Networks for the Diagnosis and Prognosis of Diseases: A Comprehensive Review"
* Link: <https://arxiv.org/abs/2304.06400>
* Authors: Carlos Segundo Muñoz-Valencia et al.

2.Book

* Title: "Bayesian Biostatistics and Diagnostic Medicine"
* Author: Lyle D. Broemeling
* Link: <https://www.barnesandnoble.com/w/bayesian-biostatistics-and-diagnostic-medicine-lyle-d-broemeling/1132542227>

3.YouTube Video

* Title: "Diagnose a disease using Bayes Theorem, step by step upon evidence and lab test result arrives."
* Channel: Spice Logic
* Link: <https://www.youtube.com/watch?v=C9D6hcY6-Es>