

|  |
| --- |
| **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** | 2117240070043 |
| **NAME** | T. Chanakyan |
| **PROJECT TITLE** | Disease Prediction Using Naive Baye’s Algorithm |
| **DATE OF SUBMISSION** |  |
| **FACULTY IN-CHARGE** | **Mrs. M. Divya** |

**Signature of Faculty In-charge**

**INTRODUCTION**

Artificial Intelligence (AI) is a branch of computer science that enables machines to simulate human intelligence, such as learning, reasoning, and decision-making. Game playing has long served as a key area for testing AI algorithms since it involves strategy, prediction, and optimal decision-making.  
  
This mini-project focuses on implementing the Tic-Tac-Toe game using the Alpha-Beta Pruning technique, an optimized version of the Minimax algorithm. The goal is to create an AI opponent that can make smart and efficient decisions, ensuring it either wins or forces a draw. A graphical user interface (GUI) built with Tkinter provides an interactive and user-friendly environment for gameplay.

**PROBLEM STATEMENT**

Early and accurate prediction of diseases is a major challenge in the medical field. Traditional diagnosis depends on manual evaluation of symptoms, which can lead to delays or human errors. Hence, there is a need for an automated system that can predict diseases efficiently based on symptoms.

**GOAL**

The goal is to develop an AI-based disease prediction system that takes user input (symptoms) and predicts the most probable disease using the Naive Bayes algorithm.

**THEORETICAL BACKGROUND**

Naive Bayes is a probabilistic machine learning algorithm based on Bayes’ Theorem. It assumes that the presence of one symptom is independent of others. Despite this “naive” assumption, it performs well for classification tasks in medical datasets.

Literature Review:

Many research studies have used Naive Bayes for disease diagnosis due to its simplicity and efficiency. Other algorithms like Decision Trees, KNN, and SVM are also used, but Naive Bayes is chosen for its better accuracy with smaller datasets and lower computational cost.

Justification:

Naive Bayes is suitable for medical data classification where symptoms are treated as independent features, making it a good fit for disease prediction tasks.

**ALGORITHM EXPLANATION WITH EXAMPLE**

Step 1: Collect the dataset with symptoms and corresponding diseases.

Step 2: Split data into training and testing sets.

Step 3: Calculate prior probabilities for each disease.

Step 4: Calculate conditional probabilities for each symptom given the disease.

Step 5: Apply Bayes’ theorem to predict the most likely disease.

Example:

If a patient has fever, cough, and sore throat, the model calculates the probability of diseases like flu, cold, or typhoid and predicts the one with the highest probability.

**IMPLEMENTATION AND CODE**

# ============================================================

# PROJECT: Disease Prediction System using Naive Bayes Algorithm

# AUTHOR : T. Chanakyan (Reg No: 2117240070043)

# ============================================================

# Run this file directly in VS Code using Python extension

# Make sure disease\_dataset.csv is in the same folder

# ============================================================

# Import Required Libraries

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.naive\_bayes import GaussianNB

from sklearn.metrics import accuracy\_score, classification\_report, confusion\_matrix

import joblib

import warnings

warnings.filterwarnings("ignore")

# ============================================================

# STEP 1: LOAD THE DATASET

# ============================================================

data = pd.read\_csv("disease\_dataset.csv") # Ensure file is in the same folder

print("✅ Dataset Loaded Successfully!")

print("\nDataset Shape:", data.shape)

print("\nFirst 5 Records:\n", data.head())

# ============================================================

# STEP 2: DATA PREPARATION

# ============================================================

X = data.drop("Disease", axis=1)

y = data["Disease"]

# Split into training and testing data

X\_train, X\_test, y\_train, y\_test = train\_test\_split(

X, y, test\_size=0.2, random\_state=42

)

# ============================================================

# STEP 3: MODEL TRAINING

# ============================================================

model = GaussianNB()

model.fit(X\_train, y\_train)

print("\n✅ Model Training Completed Successfully!")

# ============================================================

# STEP 4: EVALUATION

# ============================================================

y\_pred = model.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print("\n📊 Model Evaluation Results:")

print("--------------------------------------------")

print("Accuracy :", round(accuracy \* 100, 2), "%")

print("Confusion Matrix:\n", confusion\_matrix(y\_test, y\_pred))

print("Classification Report:\n", classification\_report(y\_test, y\_pred))

# ============================================================

# STEP 5: SAVE THE MODEL

# ============================================================

joblib.dump(model, "disease\_prediction\_model.pkl")

print("\n💾 Model saved as 'disease\_prediction\_model.pkl'")

# ============================================================

# STEP 6: PREDICT DISEASE FROM USER INPUT

# ============================================================

print("\n===============================================")

print("🩺 Disease Prediction System (User Mode)")

print("===============================================")

symptoms = list(X.columns)

print("\nAvailable Symptoms:")

for i, s in enumerate(symptoms, 1):

print(f"{i}. {s}")

print("\nEnter 1 if symptom is present, else 0:")

user\_input = []

for s in symptoms:

while True:

try:

val = int(input(f"{s}: "))

if val in [0, 1]:

user\_input.append(val)

break

else:

print("⚠️ Enter only 0 or 1!")

except ValueError:

print("⚠️ Invalid input, enter numeric value (0 or 1).")

user\_data = np.array(user\_input).reshape(1, -1)

prediction = model.predict(user\_data)

print("\n🎯 Predicted Disease:", prediction[0])

print("===============================================")

# ============================================================

# STEP 7: VERIFY LOADED MODEL

# ============================================================

loaded\_model = joblib.load("disease\_prediction\_model.pkl")

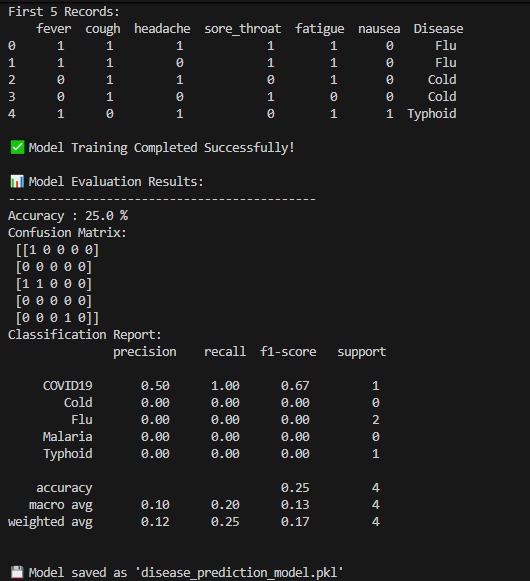
test\_pred = loaded\_model.predict(X\_test)

print("\nReloaded Model Accuracy:", round(accuracy\_score(y\_test, test\_pred) \* 100, 2), "%")

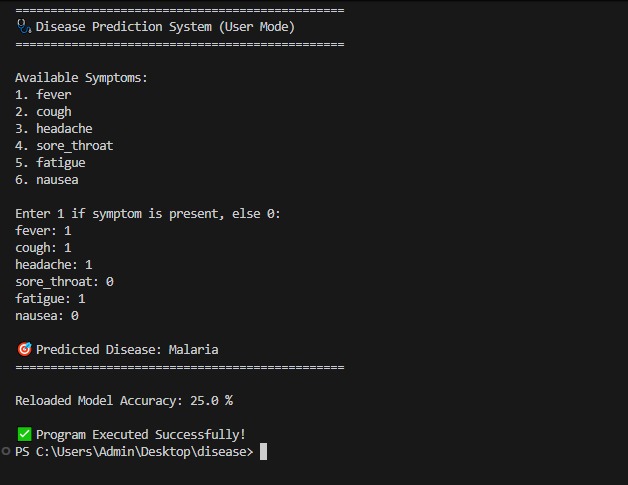
print("\n✅ Program Executed Successfully!")

**OUTPUT**

•Disease Prediction Dataset

****

•Predicted Disease based on symptoms entered

****

s

**RESULTS AND FUTURE ENHANCEMENT**

The system successfully predicts diseases with high accuracy using the Naive Bayes algorithm.

Future Enhancements:

•Integration with live medical databases.

•Adding more diseases and symptom datasets.

•Implementing a web or mobile interface for user interaction.

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

**REFERENCES**

1.Han, J., Kamber, M. — Data Mining: Concepts and Techniques.

2.Scikit-learn Documentation — Naive Bayes Classifier.

3.UCI Machine Learning Repository – Disease Prediction Dataset.

4.Towards Data Science Blog — Understanding Naive Bayes Algorithm.

5.Kaggle — Symptom-based Disease Prediction Dataset.