Module Code & Module Title

CU6051NP Artificial Intelligence

75% Individual Coursework

Submission: Final Submission

Academic Semester: Autumn Semester 2025

Credit: 15 credit semester long module

Student Name: Santosh Bhandari

London Met ID: 23049001

Assignment Due Date: 07/01/2026.

Assignment Submission Date: 07/01/2026

Submitted To: Mr. Jeevan Prakash Pant

GitHub Link	https://github.com/SantoshBhandari21/AI-coursework-s-application.git

Table of Contents

Table of Figures:

1. Introduction

This report is all about the coursework of Artificial Intelligence(CU6051NP) module. This report carries around 25% of the total marks of this module. In this module, we have learned about Machine Learning, Natural Language Processing, Algorithms and so on. This module really helps students on developing a conceptual solution using Machine Learning practices which helps us to deal with concepts and techniques to tackle real world problems.

1.1 Topic and AI concepts used:

Here, I am utilizing Machine Learning (ML) and Recommendation Systems to design a “Medicine Recommendation System”. The main goal of this project is to predict the possible disease and recommend patients the possible medicine and precautionary measures to prevent that possible disease. Here, this project proposes a machine learning based decision support system which first predicts the most probable disease based on patient’s symptoms and then recommends suitable medicines along with precautionary measures.

In this project, Supervised Learning is employed which trains models on labeled data to map inputs like patient’s symptoms in order to give output of a probable disease. Another reason of selecting supervised learning is because of its interpretability and reliability in healthcare decision support system. Generally, supervised learning provides us a structured and data-driven approach for disease prediction, which directly supports our project.

1.2 Problem Domain:

In context of Nepal, Our Healthcare systems mostly face challenges like accessibility, time limitations and the availability of medical professionals, especially for preliminary medical guidance. Before consulting a doctor, many individuals have to depend on confusing initial guidance and advice from non-medical professionals. And, incorrect interpretation may lead to inappropriate medicine usage that may lead to life threatening side effects. This creates a need for decision support systems that can provide preliminary guidance. So, this project is all about automating the preliminary guidance or support system.

Here are more key points regarding this problem in context of Nepal:

Limited medical resources: Many regions in Nepal have fewer healthcare professionals which causes delays in diagnosis and treatment.

High reliance on self-medication: Due to limited access, people often choose medicines based on symptoms without proper medical consultation.

Need for digital health tools: Mobile and web-based systems can bridge the gap in accessibility which provides initial guidance based on symptom analysis.

Symptom variability and complexity: Many diseases can have same symptoms which makes making manual interpretation challenging.

Potential for supervised learning: Symptoms and prescriptions can be used to train models for disease prediction and medicine recommendation.

Decision support focus: This system assists users by suggesting the most probable disease, relevant medicines and precautionary measures, without replacing professional medical advice as important.

Relevance for Nepalese healthcare: This approach can improve early guidance in remote areas which reduces the risk of inappropriate medication and supports public health awareness.

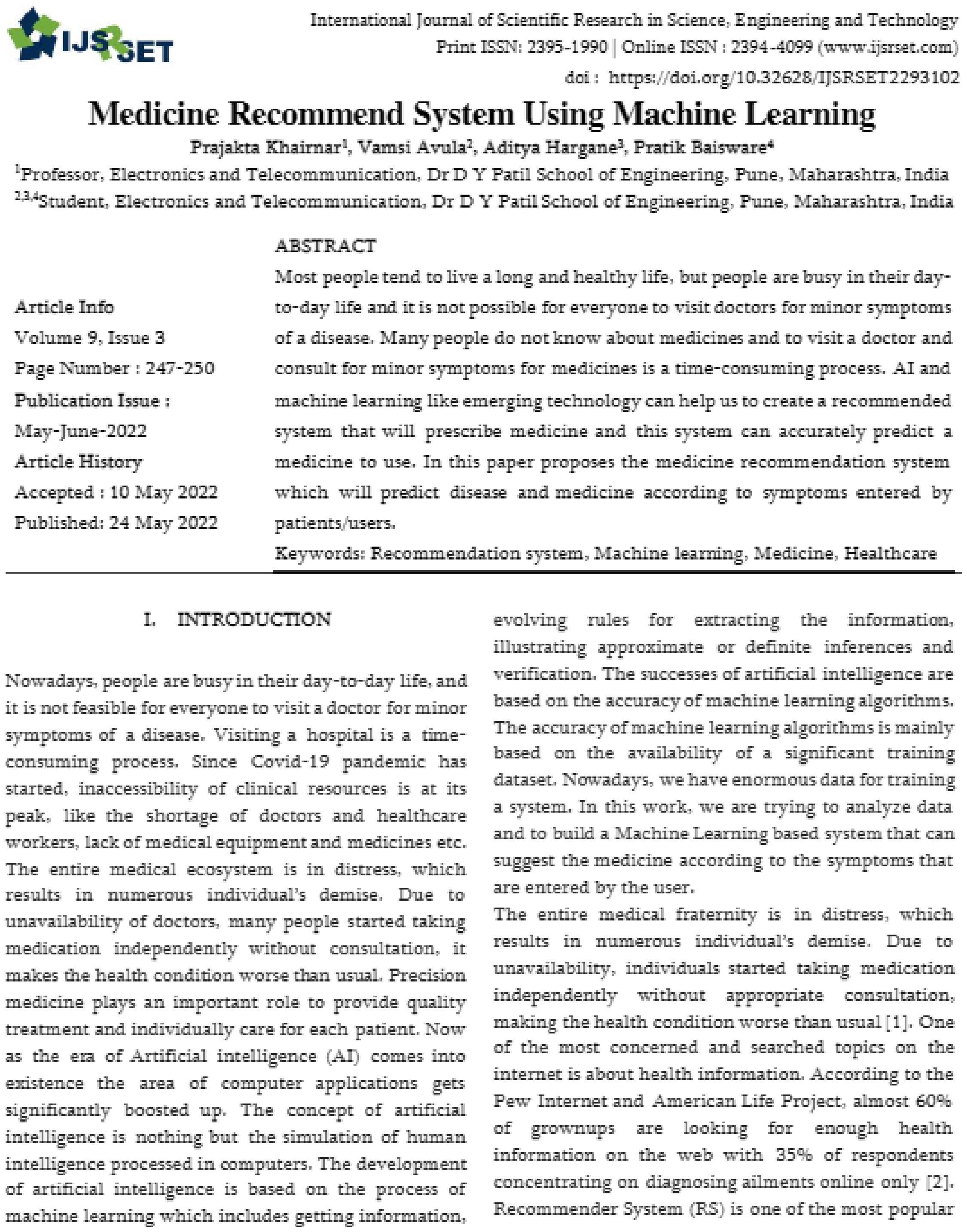
2. Background:

Healthcare is one of the most vital areas where advance technology can make enormous positive impact. With the growth of AI and ML, many researchers and developers have grown their interests on how these technologies can help in disease cure and preventions. In most of the cases due to distance, lack of doctors, or time, patients have suffered. Because of these issues, there is growing need of system that can provide basic and preliminary guidance to the people in need immediately.

Medicine Recommendation System aims to analyze symptoms of a patient and then diagnose the possible disease then recommend medicine and preventive measures. These type if systems are developed to support patients and help them make better decision about their health. The need of these kind of system are very useful in developing countries like ours, Nepal, where healthcare and medication resources are limited.

2.1 Research Work Done:

Many studies have been conducted in past with the utilization of Machine Learning for Medicine recommendation purposes. Mostly, these studies focus on utilizing supervised learning model trained on medical datasets.



International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN: 2395-1990 | Online ISSN : 2394-4099 (www.ijsrset.com)
doi : https://doi.org/10.32628/IJSRSET2293102

Medicine Recommend System Using Machine Learning

Prajakta Khairnar[1], Vamsi Avula[2], Aditya Hargane[3], Pratik Baisware[4]
[1]Professor, Electronics and Telecommunication, Dr D Y Patil School of Engineering, Pune, Maharashtra, India
[2,3,4]Student, Electronics and Telecommunication, Dr D Y Patil School of Engineering, Pune, Maharashtra, India

Article Info
Volume 9, Issue 3
Page Number : 247-250
Publication Issue :
May-June-2022
Article History
Accepted : 10 May 2022
Published: 24 May 2022

ABSTRACT
Most people tend to live a long and healthy life, but people are busy in their day-to-day life and it is not possible for everyone to visit doctors for minor symptoms of a disease. Many people do not know about medicines and to visit a doctor and consult for minor symptoms for medicines is a time-consuming process. AI and machine learning like emerging technology can help us to create a recommended system that will prescribe medicine and this system can accurately predict a medicine to use. In this paper proposes the medicine recommendation system which will predict disease and medicine according to symptoms entered by patients/users.
Keywords: Recommendation system, Machine learning, Medicine, Healthcare

I. INTRODUCTION

Nowadays, people are busy in their day-to-day life, and it is not feasible for everyone to visit a doctor for minor symptoms of a disease. Visiting a hospital is a time-consuming process. Since Covid-19 pandemic has started, inaccessibility of clinical resources is at its peak, like the shortage of doctors and healthcare workers, lack of medical equipment and medicines etc. The entire medical ecosystem is in distress, which results in numerous individual's demise. Due to unavailability of doctors, many people started taking medication independently without consultation, it makes the health condition worse than usual. Precision medicine plays an important role to provide quality treatment and individually care for each patient. Now as the era of Artificial intelligence (AI) comes into existence the area of computer applications gets significantly boosted up. The concept of artificial intelligence is nothing but the simulation of human intelligence processed in computers. The development of artificial intelligence is based on the process of machine learning which includes getting information, evolving rules for extracting the information, illustrating approximate or definite inferences and verification. The successes of artificial intelligence are based on the accuracy of machine learning algorithms. The accuracy of machine learning algorithms is mainly based on the availability of a significant training dataset. Nowadays, we have enormous data for training a system. In this work, we are trying to analyze data and to build a Machine Learning based system that can suggest the medicine according to the symptoms that are entered by the user.
The entire medical fraternity is in distress, which results in numerous individual's demise. Due to unavailability, individuals started taking medication independently without appropriate consultation, making the health condition worse than usual [1]. One of the most concerned and searched topics on the internet is about health information. According to the Pew Internet and American Life Project, almost 60% of grownups are looking for enough health information on the web with 35% of respondents concentrating on diagnosing ailments online only [2]. Recommender System (RS) is one of the most popular

Figure 1: Research Number 1.

Article Source: https://doi.org/10.32628/IJSRSET2293102

This article introduces a project that uses Artificial Intelligence (AI) and Machine Learning (ML) to create a Medicine Recommender System. In simple terms, the authors note that it's hard for people to see a doctor for minor illnesses because hospitals are busy and the pandemic made things worse. This leads many people to search their symptoms online or take random medicine which can be dangerous. The solution they propose is to use the power of computers to help. By feeding a huge amount of existing patient data like symptoms, diagnosis and medicine into a Machine Learning system, they aim to build a smart tool like a website that can instantly look at a user's given symptoms and recommend the most accurate and proper medicine. The main goal is to help people and even nurses or chemists get reliable suggestions when a doctor isn't immediately available, making healthcare faster and safer.

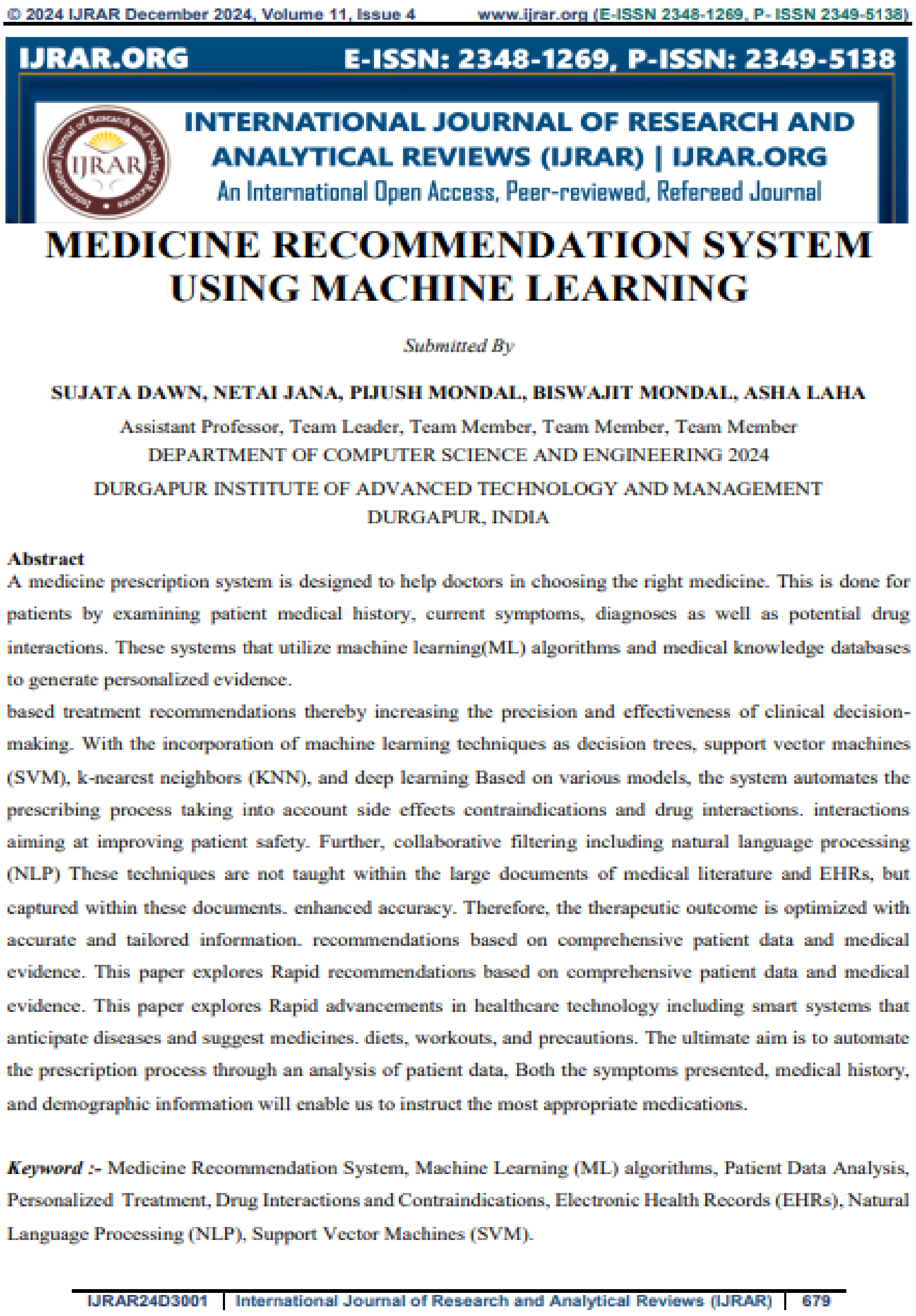
This article describes the closest project to my proposed project. Main difference with my proposed system is that my system also gives precautionary measures and this system does not.

Strength of this work:

- Shows that machine learning can be used for disease prediction.
- Focuses on helping patients when doctors are not easily available.
- Uses symptom-based input, which is simple for users.

Limitations of this work:

- Model accuracy and evaluation details are not clearly explained.
- Precautionary measures are not included.
- Real-world testing is limited.



© 2024 IJRAR December 2024, Volume 11, Issue 4 www.ijrar.org (E-ISSN 2348-1269, P- ISSN 2349-5138)

IJRAR.ORG E-ISSN: 2348-1269, P-ISSN: 2349-5138

INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS (IJRAR) | IJRAR.ORG

An International Open Access, Peer-reviewed, Refereed Journal

MEDICINE RECOMMENDATION SYSTEM USING MACHINE LEARNING

Submitted By

SUJATA DAWN, NETAI JANA, PIJUSH MONDAL, BISWAJIT MONDAL, ASHA LAHA

Assistant Professor, Team Leader, Team Member, Team Member, Team Member

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING 2024

DURGAPUR INSTITUTE OF ADVANCED TECHNOLOGY AND MANAGEMENT

DURGAPUR, INDIA

Abstract

A medicine prescription system is designed to help doctors in choosing the right medicine. This is done for patients by examining patient medical history, current symptoms, diagnoses as well as potential drug interactions. These systems that utilize machine learning(ML) algorithms and medical knowledge databases to generate personalized evidence.

based treatment recommendations thereby increasing the precision and effectiveness of clinical decision-making. With the incorporation of machine learning techniques as decision trees, support vector machines (SVM), k-nearest neighbors (KNN), and deep learning Based on various models, the system automates the prescribing process taking into account side effects contraindications and drug interactions. interactions aiming at improving patient safety. Further, collaborative filtering including natural language processing (NLP) These techniques are not taught within the large documents of medical literature and EHRs, but captured within these documents. enhanced accuracy. Therefore, the therapeutic outcome is optimized with accurate and tailored information. recommendations based on comprehensive patient data and medical evidence. This paper explores Rapid recommendations based on comprehensive patient data and medical evidence. This paper explores Rapid advancements in healthcare technology including smart systems that anticipate diseases and suggest medicines. diets, workouts, and precautions. The ultimate aim is to automate the prescription process through an analysis of patient data, Both the symptoms presented, medical history, and demographic information will enable us to instruct the most appropriate medications.

Keyword **:-** Medicine Recommendation System, Machine Learning (ML) algorithms, Patient Data Analysis, Personalized Treatment, Drug Interactions and Contraindications, Electronic Health Records (EHRs), Natural Language Processing (NLP), Support Vector Machines (SVM).

IJRAR24D3001 | International Journal of Research and Analytical Reviews (IJRAR) | 679

Figure 2: Research Number 2.

Article Source: https://ijrar.org/papers/IJRAR24D3001.pdf

This research paper describes the development of a Medicine Recommendation System that influences Machine Learning (ML) to assist doctors and patients in making better treatment choices.

In simple language, the main point of the project is to build a smart, computer-based tool that can suggest the right medicine for a patient. The system does this by looking at a huge amount of information, including the patient's past illnesses, their current symptoms, and their diagnosis. It uses powerful ML techniques, like Decision Trees and SVM, to automatically analyze this data and make personalized recommendations. A major focus is on improving patient safety by checking for dangerous drug interactions and contraindications. By using these advanced methods, the system aims to make medical decisions more precise, reduce the chance of people taking the wrong medicine and ultimately optimize how well a patient recover.

Strengths of this work:

- Highlights the problem of unsafe self-medication.
- Aims to support doctors with treatment suggestions.
- Discusses broader healthcare decision support systems.

Limitations of this work:

- Uses complex medical data that may not always be available.
- Not easy to use for general users.
- Not focused on rural or low-resource areas.

2.2 Review and Analysis of Existing Work in the Problem Domain:

Many medicine recommendation systems have already been developed in past using machine learning. These systems usually take symptoms and patients attributes as inputs and predict a possible disease. Most already developed system uses supervised learning models like Decision Trees, Support Vector Machines(SVM), Naive Bayes or Random Forest because of how these models works well with labeled medical data.

After reviewing some of existing platforms, these are some strengths which I have observed:

- Existing systems show that machine learning can successfully link symptoms to diseases.
- Many systems help users get quick health-related guidance.
- Some systems support doctors by reducing the time needed for initial diagnosis.

But they also have some limitations which are given below:

- Many systems do not clearly explain their accuracy and performance.
- Some systems are too complex for general users.
- Most systems do not focus on rural areas.
- Precautionary measures are not included.

Based on this analysis, we can understand that these systems are useful but there is still room for improvement. Including medicine suggestions, precautionary measures and clear advice to consult a doctor can make such a system more reliable and helpful to general users. So, a system which is easy to use, simple, provides factual medicines and guidance and which is much more practical in regions like Nepal is still in need.

3. Solution:

The proposed solution is a Machine Learning based Medicine Recommendation System that works steps by step. The main motive of this system is to help users get the early guidance by predicting the probable disease based on the inputted symptoms and then suggest suitable medicines and precautionary measures. Our system will first read the multiple datasets provided.

3.1 Approach to solve the problem:

Then the following approach will be implemented to solve the problem:

a. Data Preprocessing:

Firstly, the raw medical related datasets will be loaded then cleaned and prepared by handling the missing values, removing duplicate records and eliminating irrelevant attributes before training the machine learning model. Then the important features such as symptoms will be selected. Since symptoms are usually represented in textual or categorical form, they are converted into numerical values using encoding techniques such as label encoding or binary encoding. This step ensures that the data is in a suitable format for training supervised learning models.

b. Model Selection:

After preprocessing the data, we need to select an appropriate machine learning model to predict the disease. This disease prediction task is treated as a classification problem where the aim is to predict a disease from set of possible diseases. So, I have considered the following supervised learning algorithms:

Naïve Bayes: It works well with categorical data and probabilities, so it is suitable for my symptoms based prediction.

Decision Tree: it is a simple and interpretable model which splits data based on symptoms values. In a decision tree, it is easier to understand and handle both numerical and categorical data.

Random Forest: it's a collaborative method which improves prediction accuracy by combining multiple decision trees and handling non-linear relationships. And this is great for handling complex and large datasets with non-linear relations like ours.

c. Training the Model:

Once the models are selected then the given dataset will be divided into training and testing sets by usually using a 70-30 split. Then the supervised learning models will be trained using the training data where symptoms act as an input features and disease as an output label. This training will help the model to learn patterns that link symptoms to diseases. Then the model will be evaluated on the basis of their accuracy and capability to predict the possible disease.

d. Model Evaluation:

This stage comes after training the model. Here, the models are evaluated using the testing dataset. Then, the performance will be measured using evaluation metrics such as accuracy, precision, recall and F1-score. These metrics help in comparing different models and understanding their strengths and weaknesses. At the end, the model which provides us the most reliable disease prediction will be selected for the final system.

e. Disease Prediction and Recommendation:

After using the best performing model, it will predict the most probable disease when users enter their symptoms in our system. Based in these predicted disease, the system then will recommend the user the commonly used medicines and precautionary measures of that possible disease. This recommendation data will be useful for providing the initial guidance for the user but the medical professional's counselling must be taken.

f. Analysis:

Finally, our system's predictions are evaluated by comparing the predicted disease labels with the actual disease labels that are present in the dataset. These analysis helps users to determine the reliability and effectiveness of this disease prediction model. Then the quality of medicine and precautionary recommendations are also reviewed to ensure that our system provides meaningful and practical guidance, especially for users in areas with limited access to healthcare services.

Key features of my proposed solution:

- This system does not replace a doctor rather it provides guidance in early stages.
- It reduces confusion caused by overlapping symptoms.
- It is suitable for use in areas with limited access to healthcare services such as rural regions of Nepal.
- This system can be improved in future by adding more diseases or improving model accuracy.

3.2 AI algorithms used:

This proposed Medicine Recommendation System mainly depends on Supervised Machine Learning algorithms that predicts the most probable disease. Since the output disease is already known while training data so, supervised learning is the most suitable approach. To ensure accuracy, reliability and easy interpretation I have used different algorithms.

Naïve Bayes Algorithm:

- It Uses probability to predict the most probable disease based on the symptoms entered by the user.

- This algorithm assumes that each symptoms contributes independently to the disease prediction.
- Works well with categorical data such as presence or absence of symptoms.
- It is fast to train and suitable for early-stage disease prediction systems.

Decision Tree Algorithm

- This algorithm splits the symptom data into branches based on feature values to reach a disease prediction.
- Easy to understand and interpret which makes it suitable for healthcare decision support.
- It can handle both numerical and categorical symptom data effectively.
- This Helps to explain how specific symptoms leads to a particular disease outcome.

Random Forest Algorithm

- This algorithm can combine multiple decision trees to improve disease prediction accuracy.
- It Reduces overfitting by training each tree on a random subset of the data.
- It can handle complex and non-linear relationships between symptoms and diseases.
- This algorithm provides far more reliable predictions compared to a single decision tree.

3.3 Pseudocode:

Pseudocode is a process of writing codes of a program in simple terms excluding the complexity of the logical interpretation like in codes. They are written more in human like language without following any complex rules. They are very helpful for making programming more understandable and enjoyable (Mahr, 2025).

The pseudocode of this proposed system is given below:

```
START
IMPORT Libraries

a. Load Dataset
   - Load training dataset containing symptoms and disease labels
   - Load supporting datasets:
       • Symptom severity
       • Disease descriptions
       • Medicines
       • Precautionary measures

b. Data Preprocessing
   - Remove unnecessary or duplicate records
   - Check and handle missing values
   - Separate input features (symptoms) and target variable (disease)
   - Encode disease labels into numerical form
```

```
c. Split Dataset
   - Divide dataset into training set and testing set
   - Use a fixed ratio (e.g., 70% training, 30% testing)

d. Train Machine Learning Models
   FOR each selected supervised learning algorithm DO
       - Train the model using training data
       - Predict disease labels using testing data
       - Calculate accuracy and confusion matrix
   END FOR

e. Select Best Model
   - Compare model performances
   - Select the model with the highest accuracy (SVM classifier)

f. Save Trained Model
   - Store the trained model in a file for later use

g. Disease Prediction
   - Accept symptoms as user input
   - Convert input symptoms into required feature format
   - Load trained model
   - Predict the most probable disease
```

```
h. Medicine and Precaution Recommendation
   - Retrieve medicines related to the predicted disease
   - Retrieve precautionary measures for the predicted disease
   - Display disease name, medicines, and precautions to the user

i. Output Results
   - Show predicted disease
   - Show recommended medicines
   - Show precautionary measures
END
```

3.4 Flowchart:

A flowchart is a graphical representation of processes and workflow of a system. It has different shapes for different processes and they are collected by lines and represented on a step by step manner to help in decision making (Belcic, 2025).

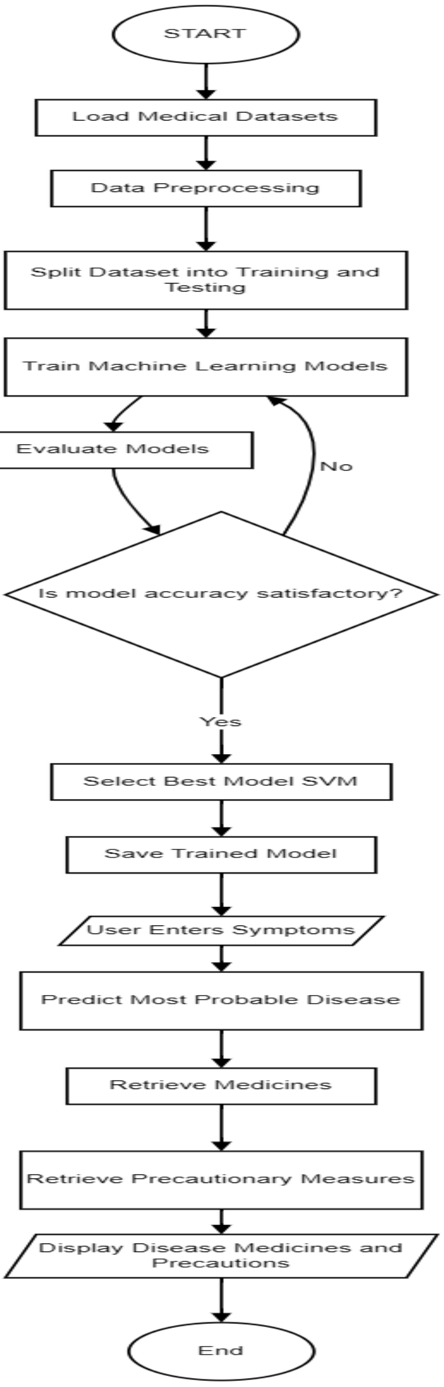

START
Load Medical Datasets
Data Preprocessing
Split Dataset into Training and Testing
Train Machine Learning Models
Evaluate Models
No
Is model accuracy satisfactory?
Yes
Select Best Model SVM
Save Trained Model
User Enters Symptoms
Predict Most Probable Disease
Retrieve Medicines
Retrieve Precautionary Measures
Display Disease Medicines and Precautions
End

Figure 3: Flowchart of my system.

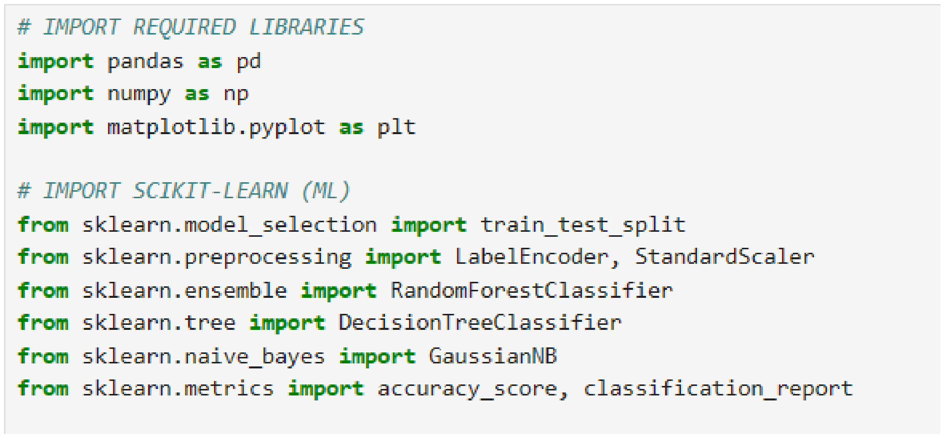
3.5 Development Process:

The step by step development process of my system is given below:

Step 1: Importing required libraries

Firstly, I have imported all the python libraries required in my project.

- Pandas was used to load and handle my CSV files.
- NumPy was used to work with arrays and to develop the symptoms input vector
- Matplotlip was simply used to work on different visualizations like bar graphs, pie charts, heat map etc.
- Scikit-learn was used to train the machine learning models, pre-process our data and evaluate their performance.



```
# IMPORT REQUIRED LIBRARIES
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# IMPORT SCIKIT-LEARN (ML)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score, classification_report
```

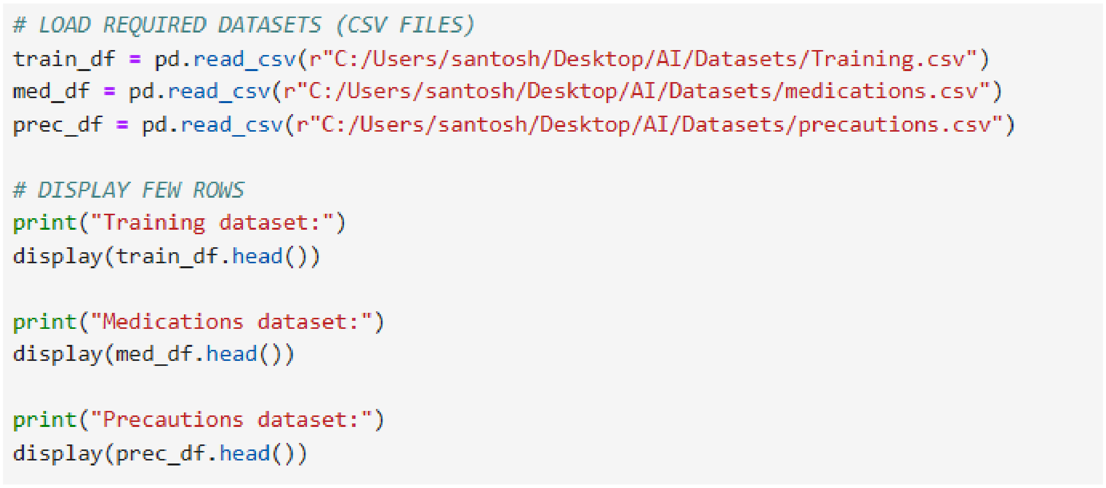
Figure 4: Importing libraries while developing my system.

Step 2: Importing datasets

For my project, I have to load 3 different datasets.

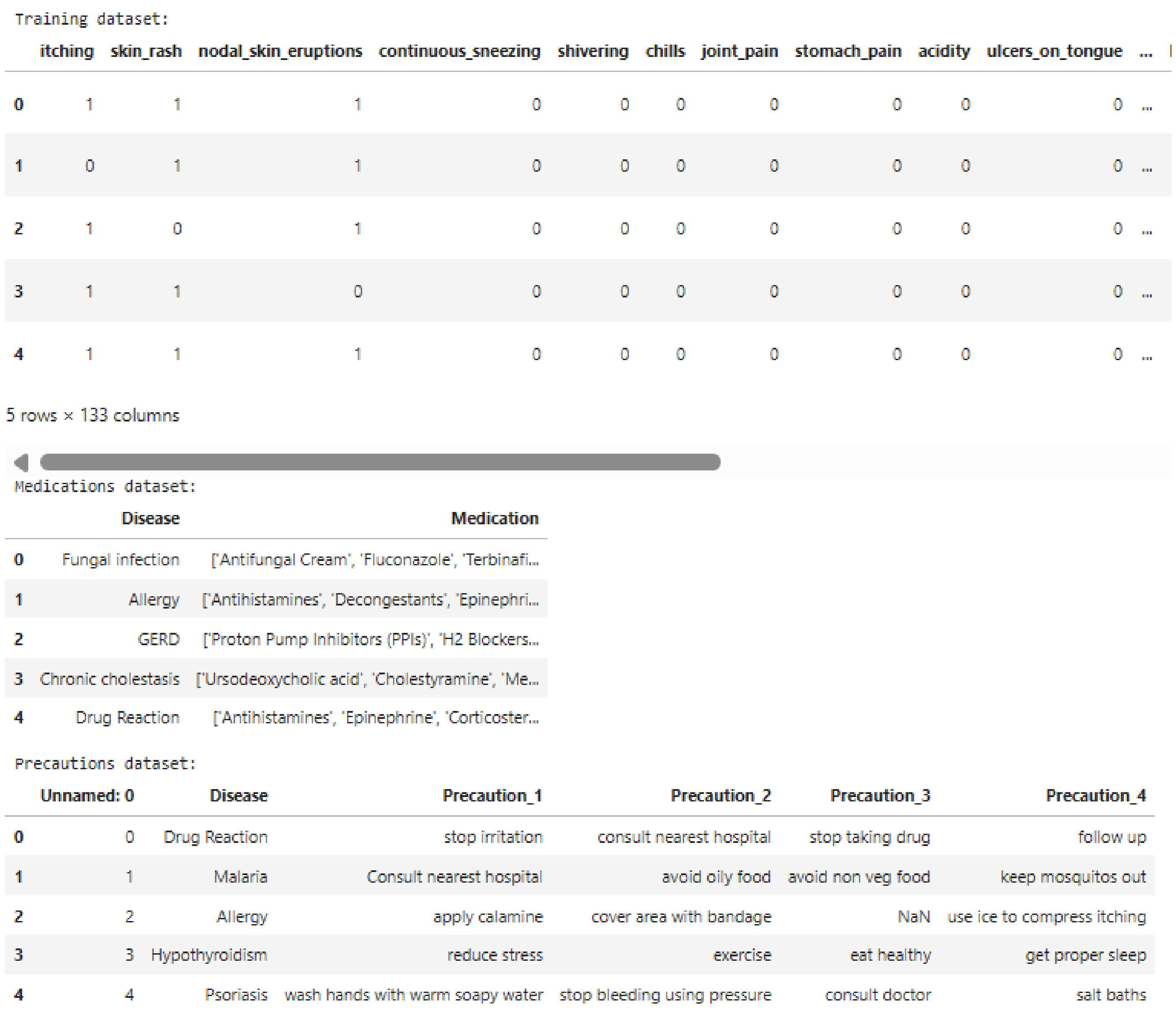
- **Training.csv**: it contains many symptoms columns and one target column ‘disease’. This was used to train the ML models.
- **medications.csv:** it has Disease and Medication. This was used to recommend medicine after prediction.

- **precautions.csv**: it contains Disease and Precaution_1 to Precaution_4. This was used to recommend precautions after prediction.



```
# LOAD REQUIRED DATASETS (CSV FILES)
train_df = pd.read_csv(r"C:/Users/santosh/Desktop/AI/Datasets/Training.csv")
med_df = pd.read_csv(r"C:/Users/santosh/Desktop/AI/Datasets/medications.csv")
prec_df = pd.read_csv(r"C:/Users/santosh/Desktop/AI/Datasets/precautions.csv")

# DISPLAY FEW ROWS
print("Training dataset:")
display(train_df.head())

print("Medications dataset:")
display(med_df.head())

print("Precautions dataset:")
display(prec_df.head())
```

Figure 5: Importing datasets and displaying top 5 rows.



Training dataset:

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...
0	1	1	1	0	0	0	0	0	0	0	...
1	0	1	1	0	0	0	0	0	0	0	...
2	1	0	1	0	0	0	0	0	0	0	...
3	1	1	0	0	0	0	0	0	0	0	...
4	1	1	1	0	0	0	0	0	0	0	...

5 rows × 133 columns

Medications dataset:

	Disease	Medication
0	Fungal infection	['Antifungal Cream', 'Fluconazole', 'Terbinafi...
1	Allergy	['Antihistamines', 'Decongestants', 'Epinephri...
2	GERD	['Proton Pump Inhibitors (PPIs)', 'H2 Blockers...
3	Chronic cholestasis	['Ursodeoxycholic acid', 'Cholestyramine', 'Me...
4	Drug Reaction	['Antihistamines', 'Epinephrine', 'Corticoster...

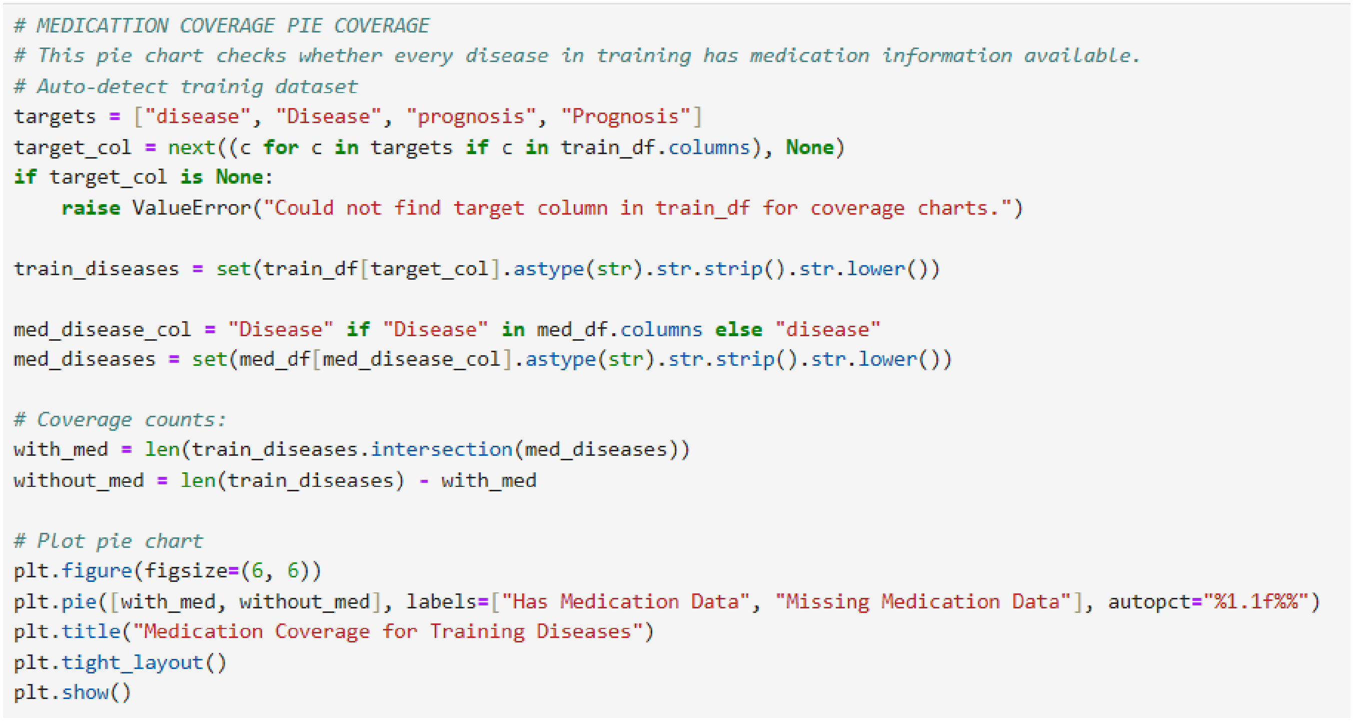
Precautions dataset:

	Unnamed: 0	Disease	Precaution_1	Precaution_2	Precaution_3	Precaution_4
0	0	Drug Reaction	stop irritation	consult nearest hospital	stop taking drug	follow up
1	1	Malaria	Consult nearest hospital	avoid oily food	avoid non veg food	keep mosquitos out
2	2	Allergy	apply calamine	cover area with bandage	NaN	use ice to compress itching
3	3	Hypothyroidism	reduce stress	exercise	eat healthy	get proper sleep
4	4	Psoriasis	wash hands with warm soapy water	stop bleeding using pressure	consult doctor	salt baths

Figure 6: Top 5 rows of my datasets.

Step 3: Checking missing values and coverage

Here, I have checked if there are any missing values in our datasets. I also checked if all diseases in the training dataset have medicine and precaution information. This is important because if a disease is missing in medications or precautions, then the system cannot recommend anything for it. Then a pie chart is created to visualize this information.



```
# MEDICATTION COVERAGE PIE COVERAGE
# This pie chart checks whether every disease in training has medication information available.
# Auto-detect trainig dataset
targets = ["disease", "Disease", "prognosis", "Prognosis"]
target_col = next((c for c in targets if c in train_df.columns), None)
if target_col is None:
    raise ValueError("Could not find target column in train_df for coverage charts.")

train_diseases = set(train_df[target_col].astype(str).str.strip().str.lower())

med_disease_col = "Disease" if "Disease" in med_df.columns else "disease"
med_diseases = set(med_df[med_disease_col].astype(str).str.strip().str.lower())

# Coverage counts:
with_med = len(train_diseases.intersection(med_diseases))
without_med = len(train_diseases) - with_med

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie([with_med, without_med], labels=["Has Medication Data", "Missing Medication Data"], autopct="%1.1f%%")
plt.title("Medication Coverage for Training Diseases")
plt.tight_layout()
plt.show()
```

Figure 7: Checking missing values of every disease.

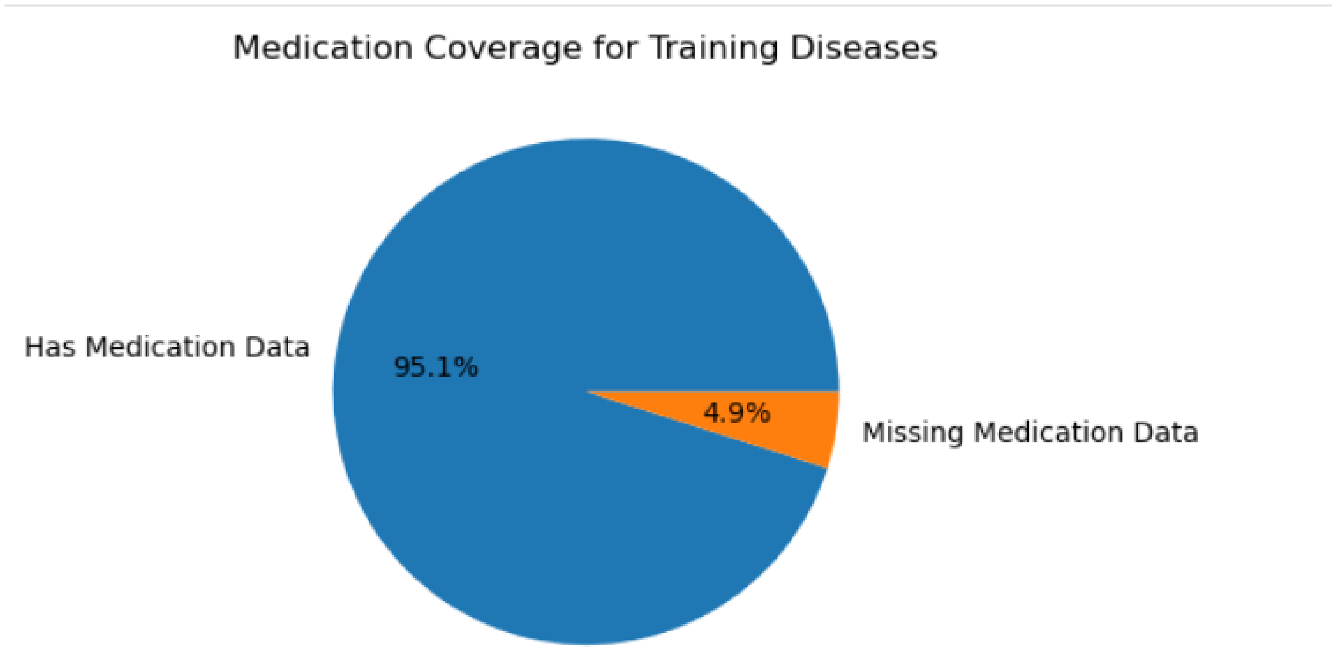
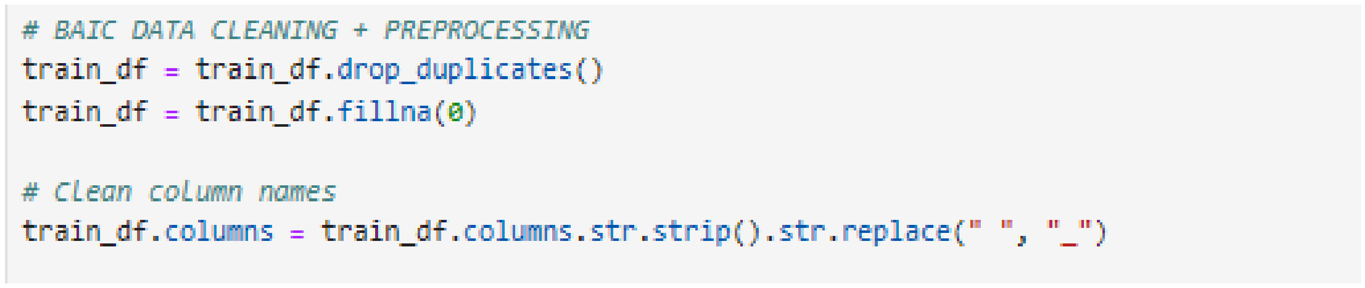


Figure 8: Pie chart of missing data and non-missing data.

Step 4: Data cleaning

I started cleaning the dataset by:

- Removing duplicate rows
- Handling missing values which are filled with 0.
- Cleaning column names by removing extra spaces and replacing spaces with underscores.

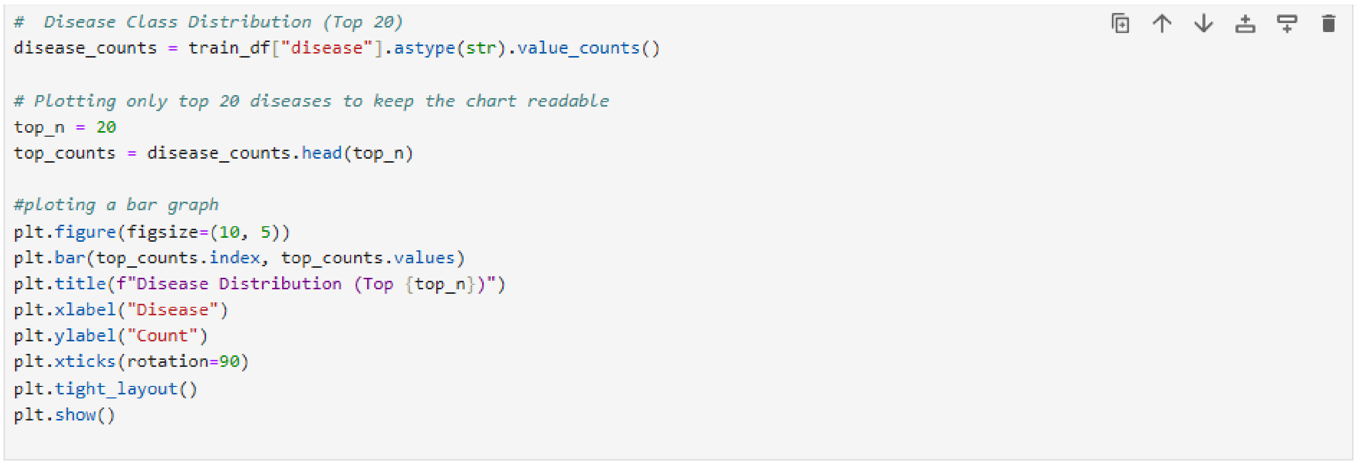


```
# BAIC DATA CLEANING + PREPROCESSING
train_df = train_df.drop_duplicates()
train_df = train_df.fillna(0)

# Clean column names
train_df.columns = train_df.columns.str.strip().str.replace(" ", "_")
```

Figure 9: Basic data cleaning process.

Step 5: Understanding disease distribution

I created a top 20 disease distribution graph to see how many records exist for each disease. This helped me understand if some diseases appear more than others in the dataset.



```
#  Disease Class Distribution (Top 20)
disease_counts = train_df["disease"].astype(str).value_counts()

# Plotting only top 20 diseases to keep the chart readable
top_n = 20
top_counts = disease_counts.head(top_n)

#ploting a bar graph
plt.figure(figsize=(10, 5))
plt.bar(top_counts.index, top_counts.values)
plt.title(f"Disease Distribution (Top {top_n})")
plt.xlabel("Disease")
plt.ylabel("Count")
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

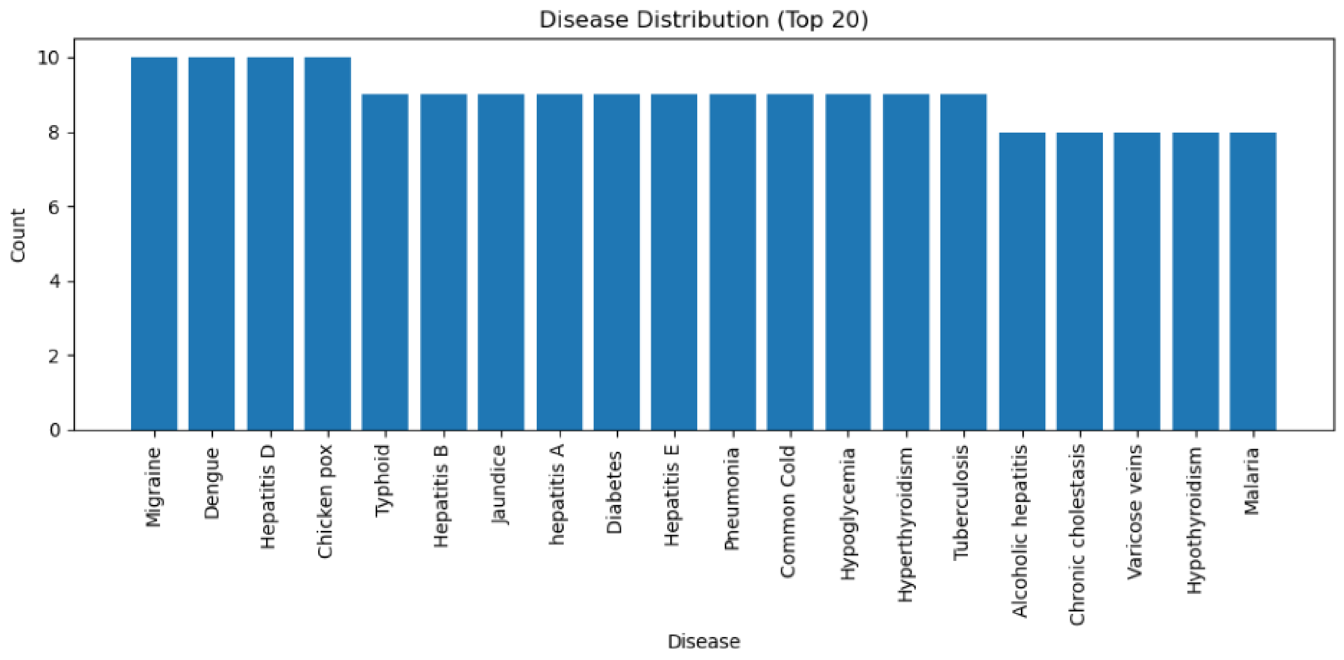


Figure 10: Visualizing the top 20 disease distributions.

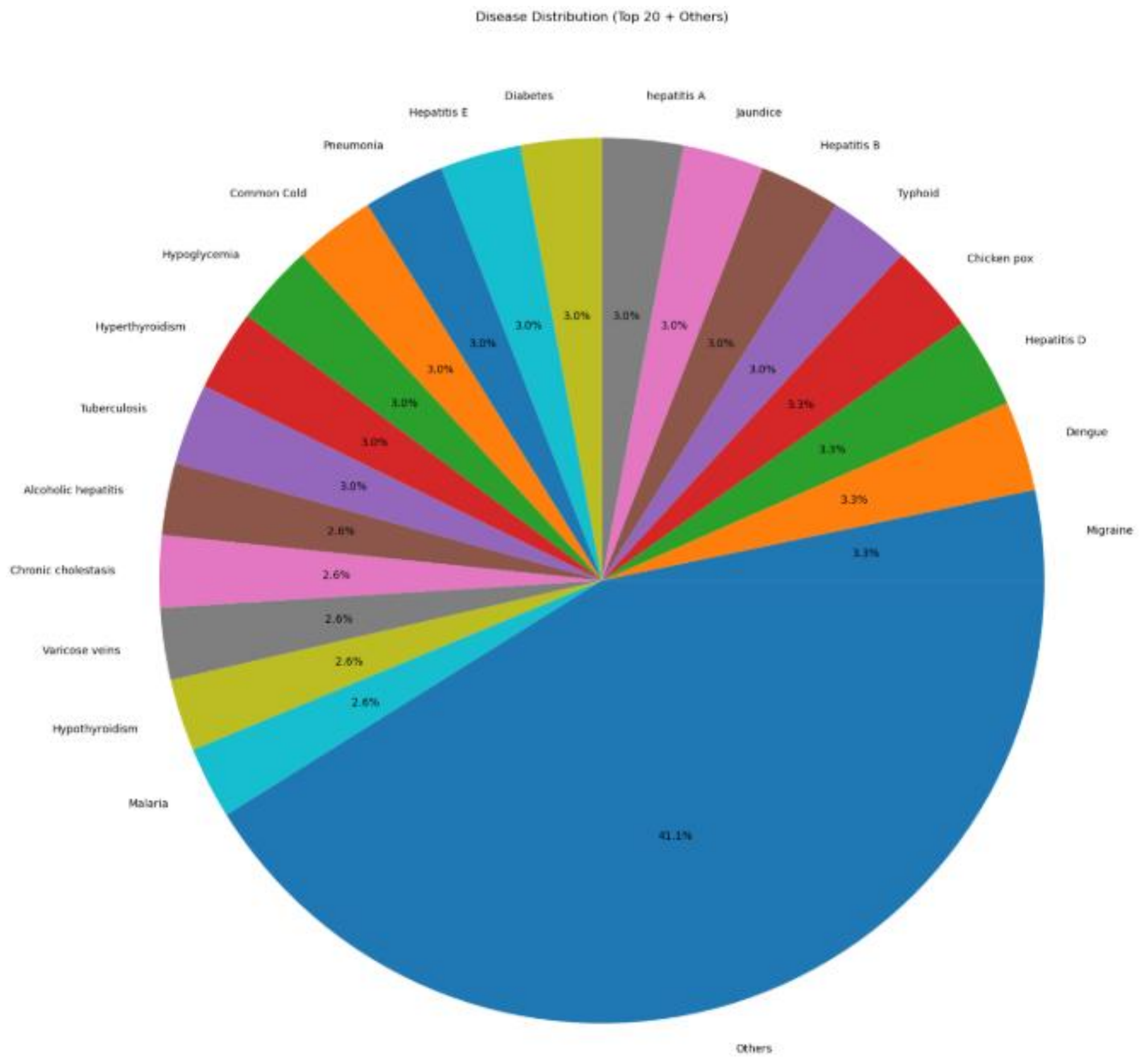


Figure 11: Visualizing disease distribution in a pie chart.

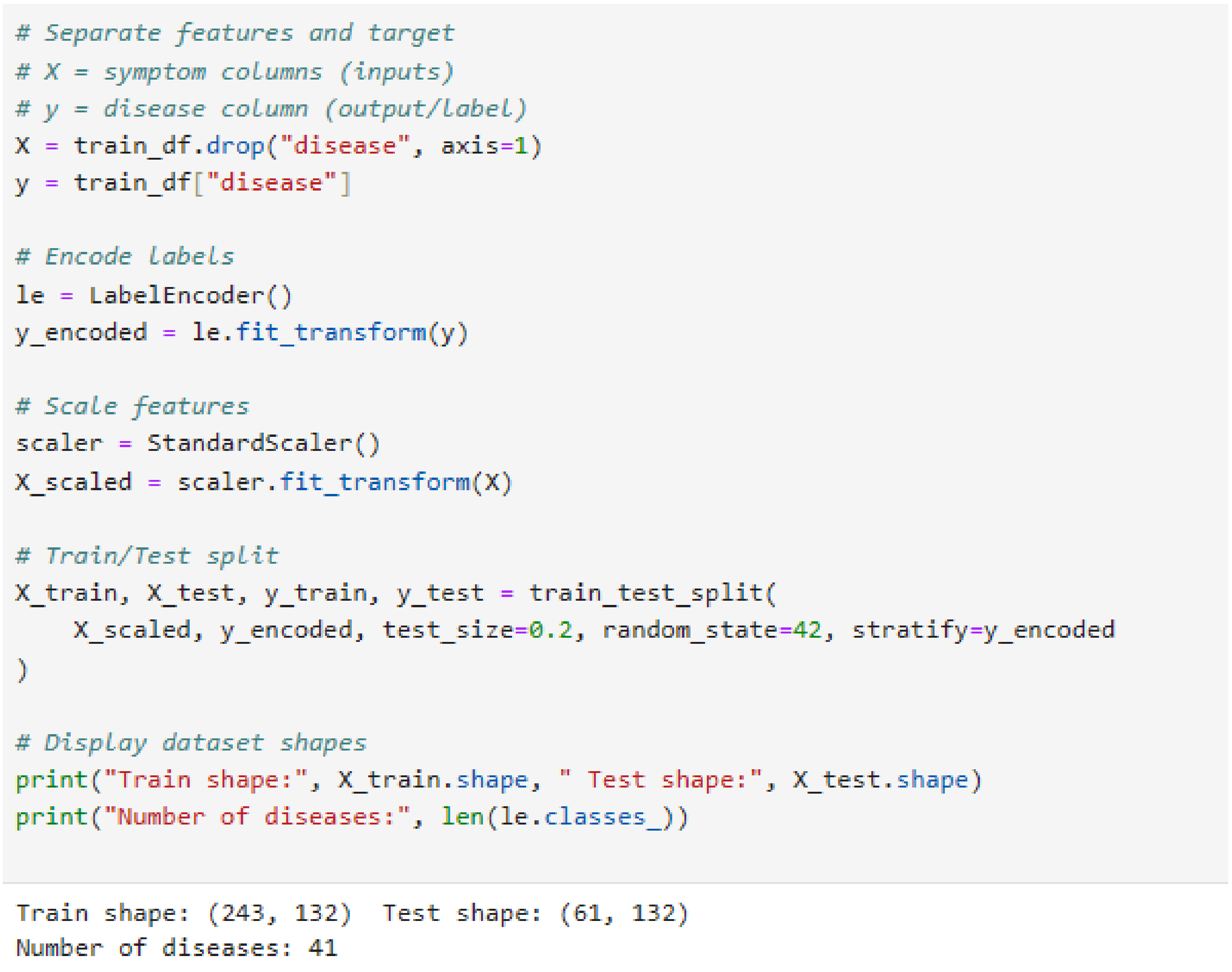
Step 6: Selecting features and target, Encoding the target and Scaling the features

I separated the dataset into:

- X (features) = all symptom columns
- y (target) = the disease column

Since machine learning models cannot work directly with text labels, I used LabelEncoder to convert disease names into numbers.

Then, I applied StandardScaler to normalize the symptom values so the model can learn better and prediction stays consistent.



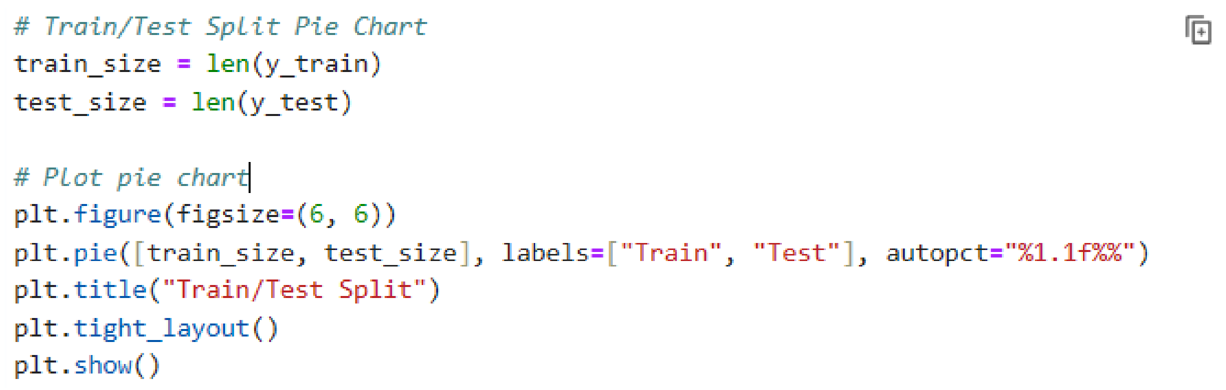
```python
# Separate features and target
# X = symptom columns (inputs)
# y = disease column (output/label)
X = train_df.drop("disease", axis=1)
y = train_df["disease"]

# Encode labels
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Scale features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Train/Test split
X_train, X_test, y_train, y_test = train_test_split(
    X_scaled, y_encoded, test_size=0.2, random_state=42, stratify=y_encoded
)

# Display dataset shapes
print("Train shape:", X_train.shape, " Test shape:", X_test.shape)
print("Number of diseases:", len(le.classes_))
```

```
Train shape: (243, 132)  Test shape: (61, 132)
Number of diseases: 41
```

Figure 12: More data cleaning processes.

Step 7: Splitting the data

I split the dataset into training and testing parts using 80% training and 20% testing. This helped to test the model on new/unseen data.



```python
# Train/Test Split Pie Chart
train_size = len(y_train)
test_size = len(y_test)

# Plot pie chart
plt.figure(figsize=(6, 6))
plt.pie([train_size, test_size], labels=["Train", "Test"], autopct="%1.1f%%")
plt.title("Train/Test Split")
plt.tight_layout()
plt.show()
```

Figure 13: Code to Train/Test split.

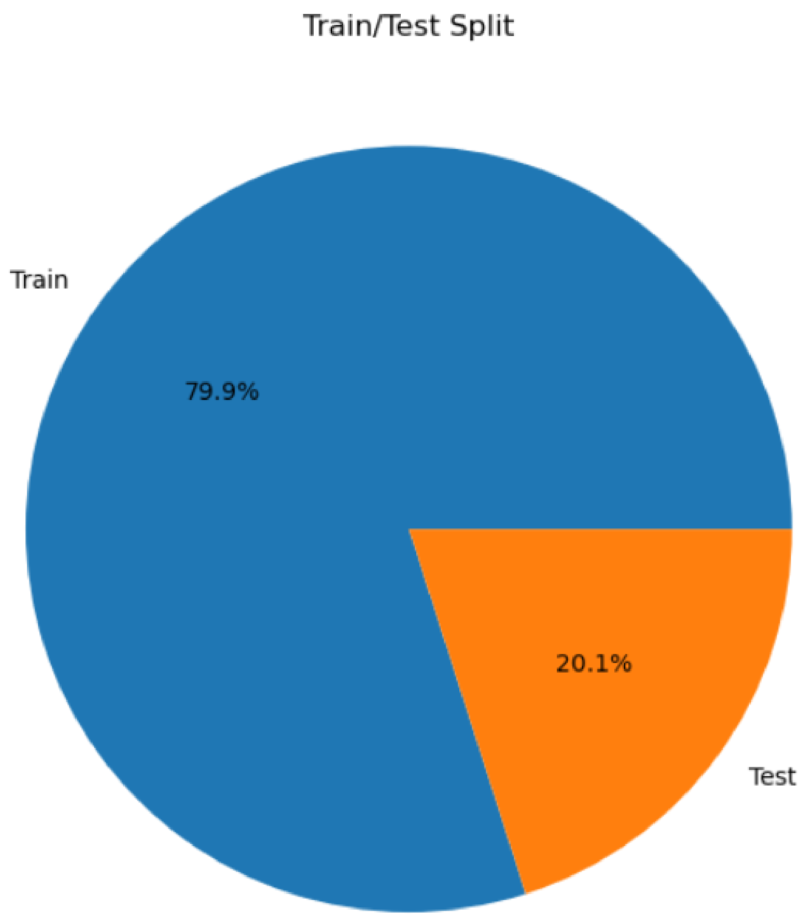
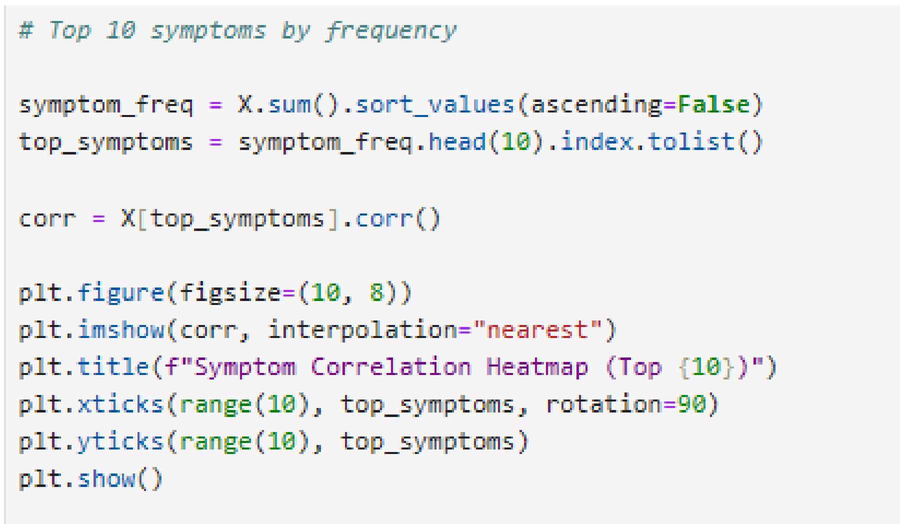


Figure 14: 80/20 Train/Test split on Pie chart.

Step 8: Symptoms Correlation Heatmap

A symptoms Correlation Heatmap was then visualized to gain better understandings of different symptoms in our datasets.



```
# Top 10 symptoms by frequency

symptom_freq = X.sum().sort_values(ascending=False)
top_symptoms = symptom_freq.head(10).index.tolist()

corr = X[top_symptoms].corr()

plt.figure(figsize=(10, 8))
plt.imshow(corr, interpolation="nearest")
plt.title(f"Symptom Correlation Heatmap (Top {10})")
plt.xticks(range(10), top_symptoms, rotation=90)
plt.yticks(range(10), top_symptoms)
plt.show()
```

Figure 15: Code to find symptoms correlation.

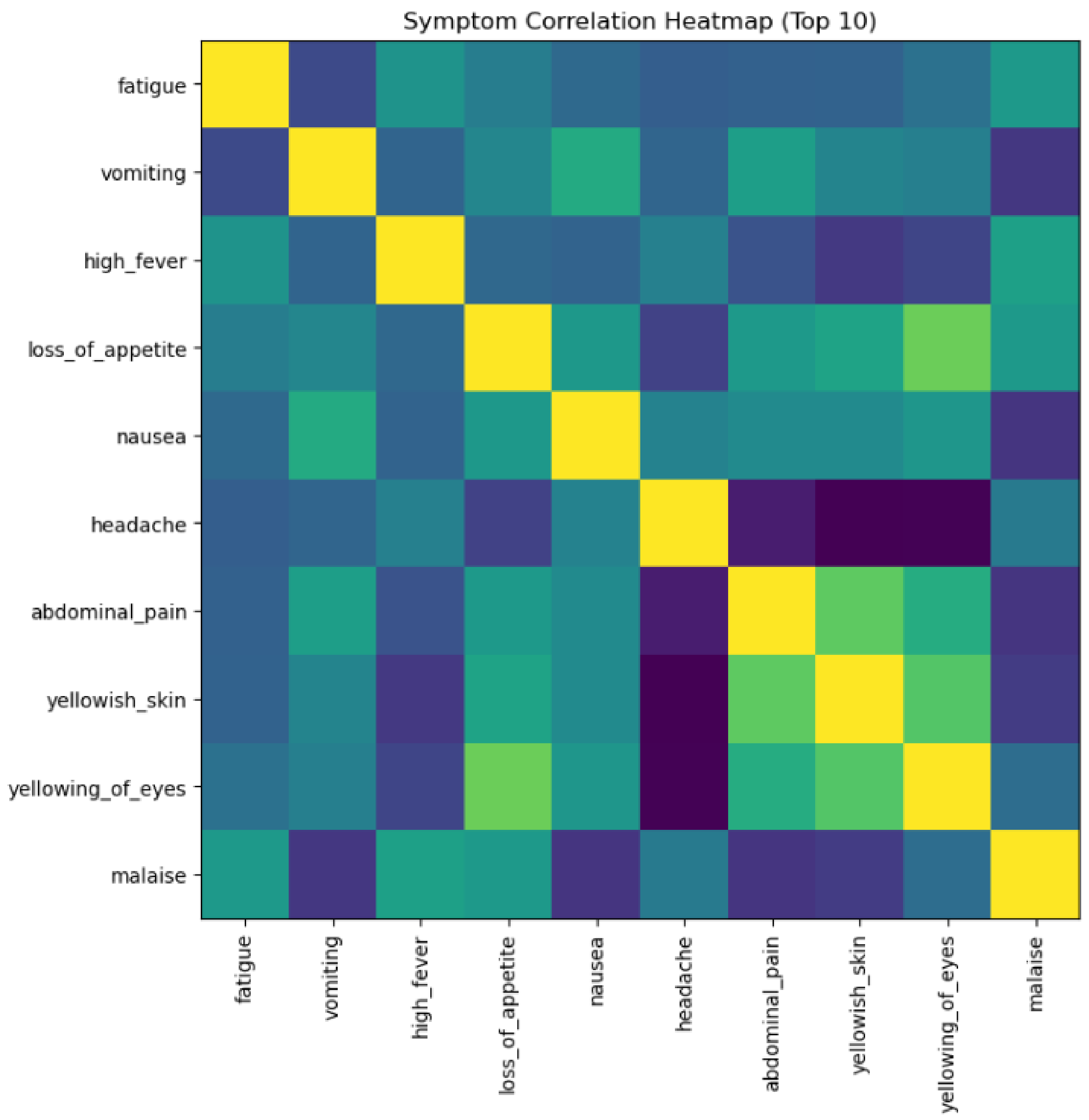


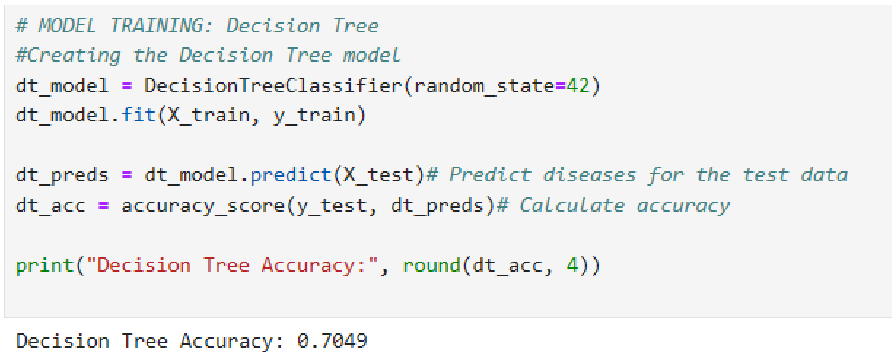
Figure 16: Symptoms correlation Heatmap.

Step 9: Training models

I trained three machine learning models:

- Decision Tree, Random Forest, Naive Bayes

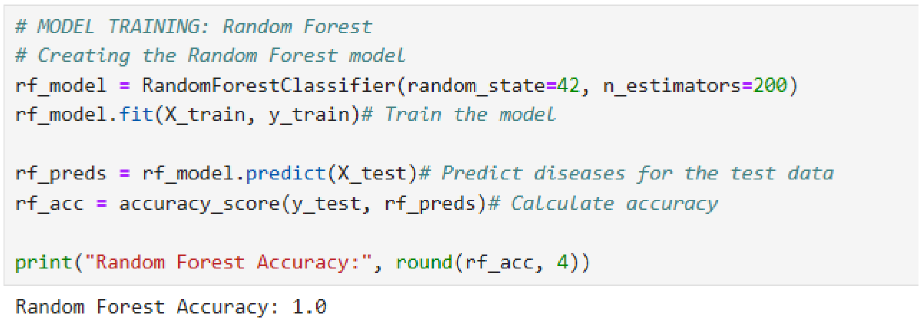
Then I tested each model and calculated their accuracy.



```
# MODEL TRAINING: Decision Tree
#Creating the Decision Tree model
dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

dt_preds = dt_model.predict(X_test)# Predict diseases for the test data
dt_acc = accuracy_score(y_test, dt_preds)# Calculate accuracy

print("Decision Tree Accuracy:", round(dt_acc, 4))
```

```
Decision Tree Accuracy: 0.7049
```

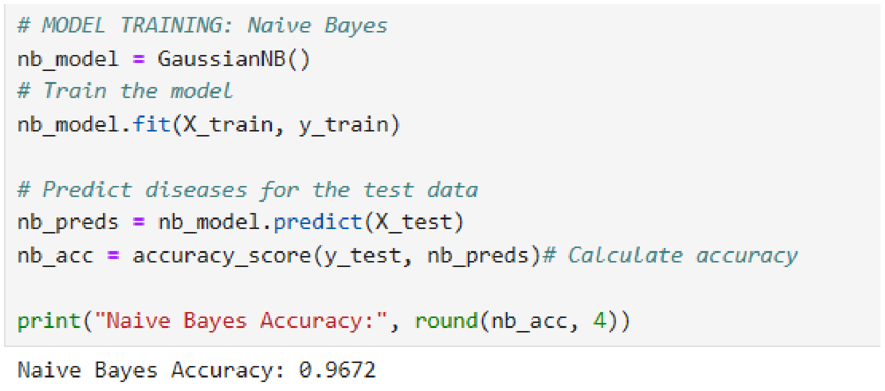
Figure 17: Decision Tree model Training with its accuracy.



```
# MODEL TRAINING: Random Forest
# Creating the Random Forest model
rf_model = RandomForestClassifier(random_state=42, n_estimators=200)
rf_model.fit(X_train, y_train)# Train the model

rf_preds = rf_model.predict(X_test)# Predict diseases for the test data
rf_acc = accuracy_score(y_test, rf_preds)# Calculate accuracy

print("Random Forest Accuracy:", round(rf_acc, 4))
```

```
Random Forest Accuracy: 1.0
```

Figure 18: Random Forest model Training with its accuracy.



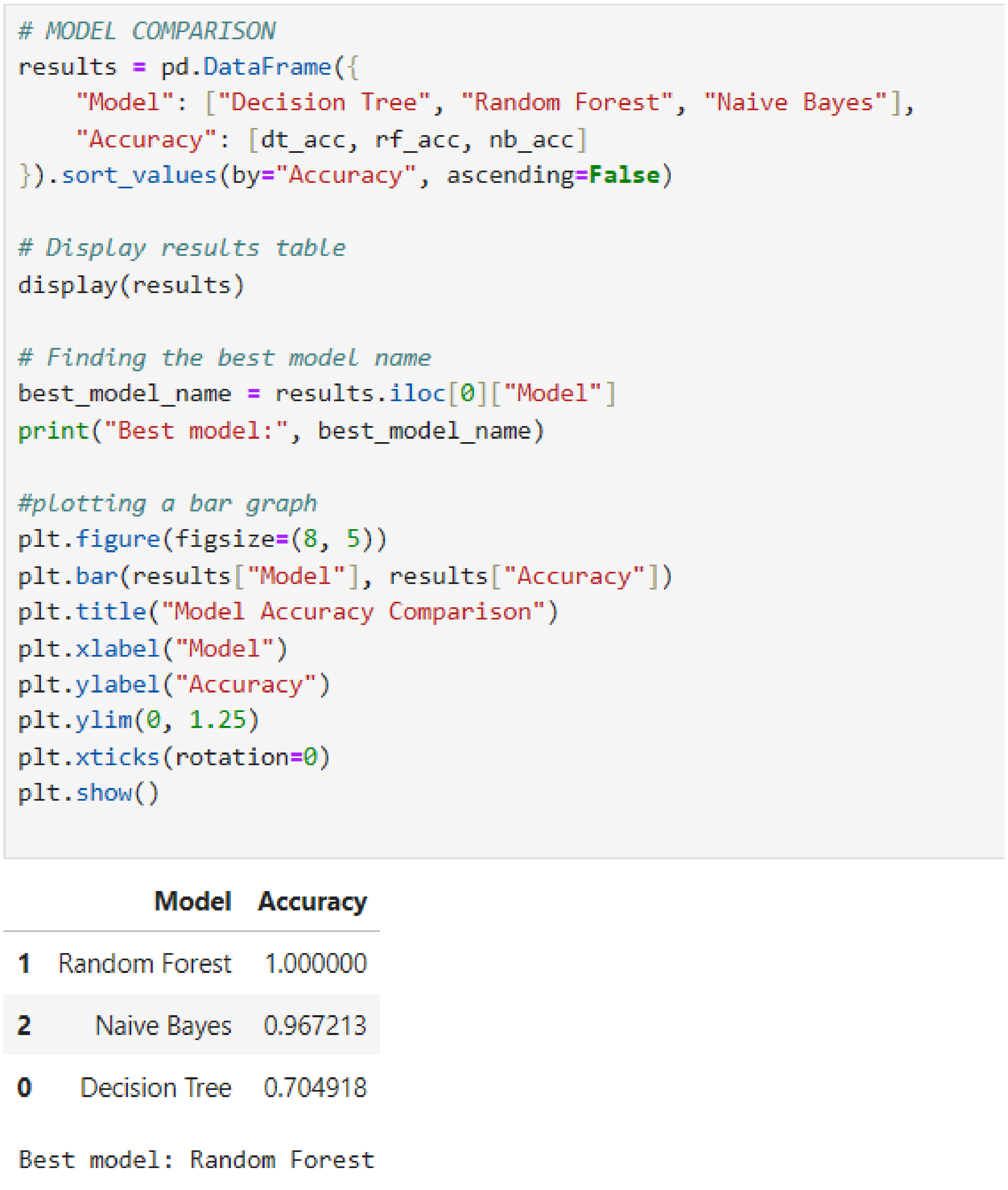
```
# MODEL TRAINING: Naive Bayes
nb_model = GaussianNB()
# Train the model
nb_model.fit(X_train, y_train)

# Predict diseases for the test data
nb_preds = nb_model.predict(X_test)
nb_acc = accuracy_score(y_test, nb_preds)# Calculate accuracy

print("Naive Bayes Accuracy:", round(nb_acc, 4))
```

```
Naive Bayes Accuracy: 0.9672
```

Figure 19: Naive Bayes model Training with its accuracy.

Step 10: Comparing models

I compared the accuracy of all these models using a table and bar graph. Random Forest gave the best accuracy, so I selected it as my final model.



```
# MODEL COMPARISON
results = pd.DataFrame({
    "Model": ["Decision Tree", "Random Forest", "Naive Bayes"],
    "Accuracy": [dt_acc, rf_acc, nb_acc]
}).sort_values(by="Accuracy", ascending=False)

# Display results table
display(results)

# Finding the best model name
best_model_name = results.iloc[0]["Model"]
print("Best model:", best_model_name)

#plotting a bar graph
plt.figure(figsize=(8, 5))
plt.bar(results["Model"], results["Accuracy"])
plt.title("Model Accuracy Comparison")
plt.xlabel("Model")
plt.ylabel("Accuracy")
plt.ylim(0, 1.25)
plt.xticks(rotation=0)
plt.show()
```

	Model	Accuracy
1	Random Forest	1.000000
2	Naive Bayes	0.967213
0	Decision Tree	0.704918

```
Best model: Random Forest
```

Figure 20: Comparing three model's accuracy.

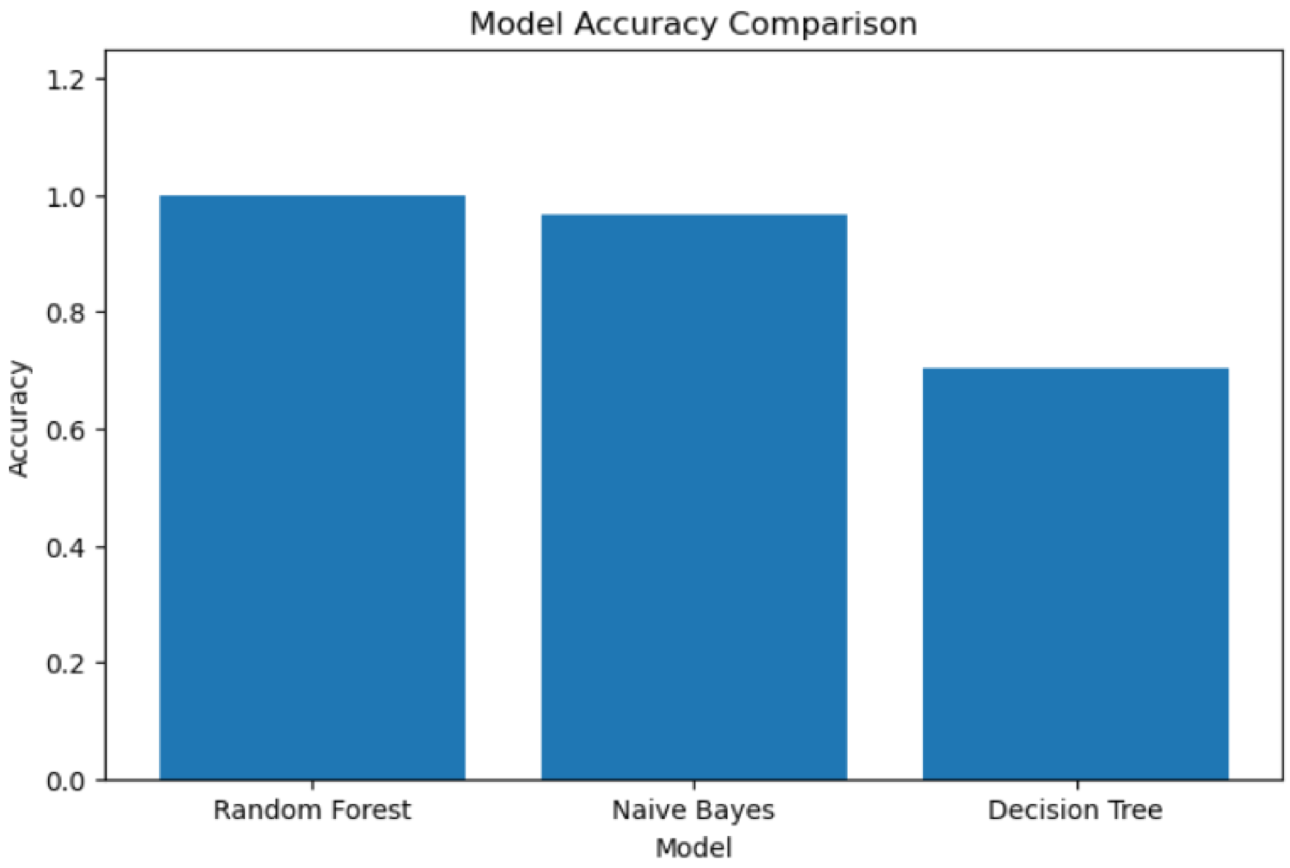
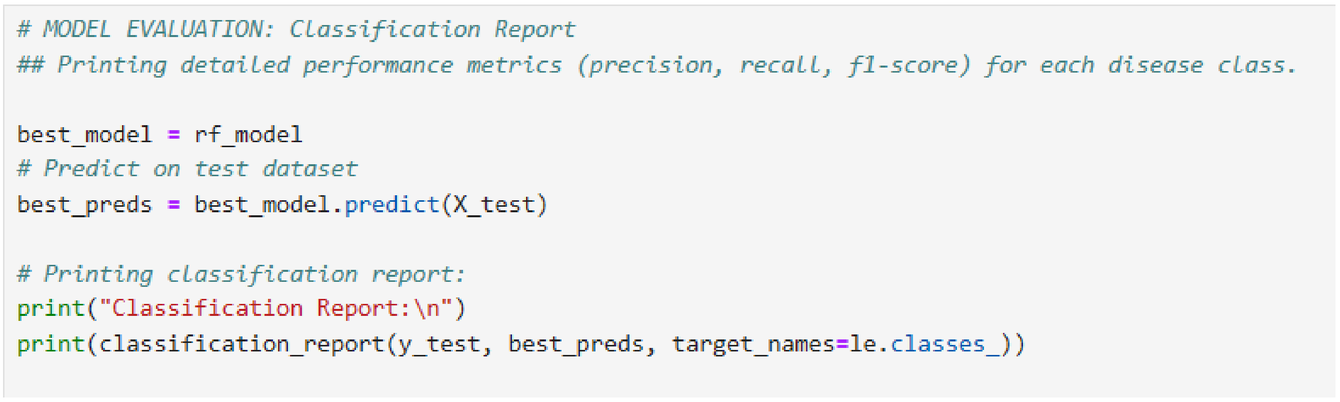


Figure 21: Model's accuracy bar graph.

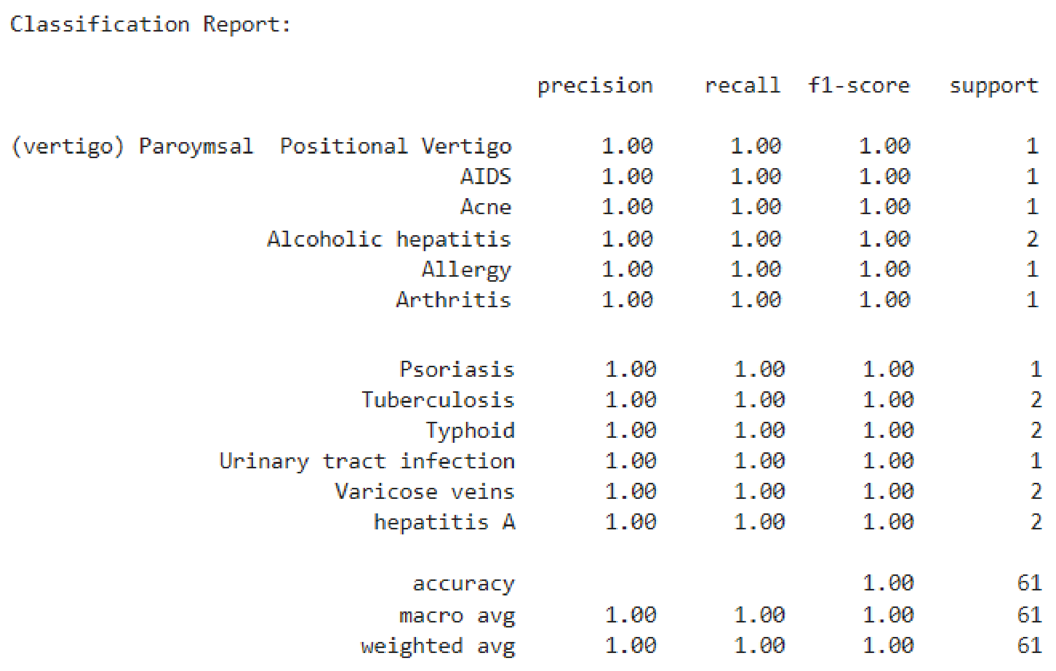
Step 11: Model evaluation

I generated the classification report to check how well the best model performs. I also compared Precision, Recall, and F1 score for all models.



```
# MODEL EVALUATION: Classification Report
## Printing detailed performance metrics (precision, recall, f1-score) for each disease class.

best_model = rf_model
# Predict on test dataset
best_preds = best_model.predict(X_test)

# Printing classification report:
print("Classification Report:\n")
print(classification_report(y_test, best_preds, target_names=le.classes_))
```

Figure 22: Code to evaluate models.

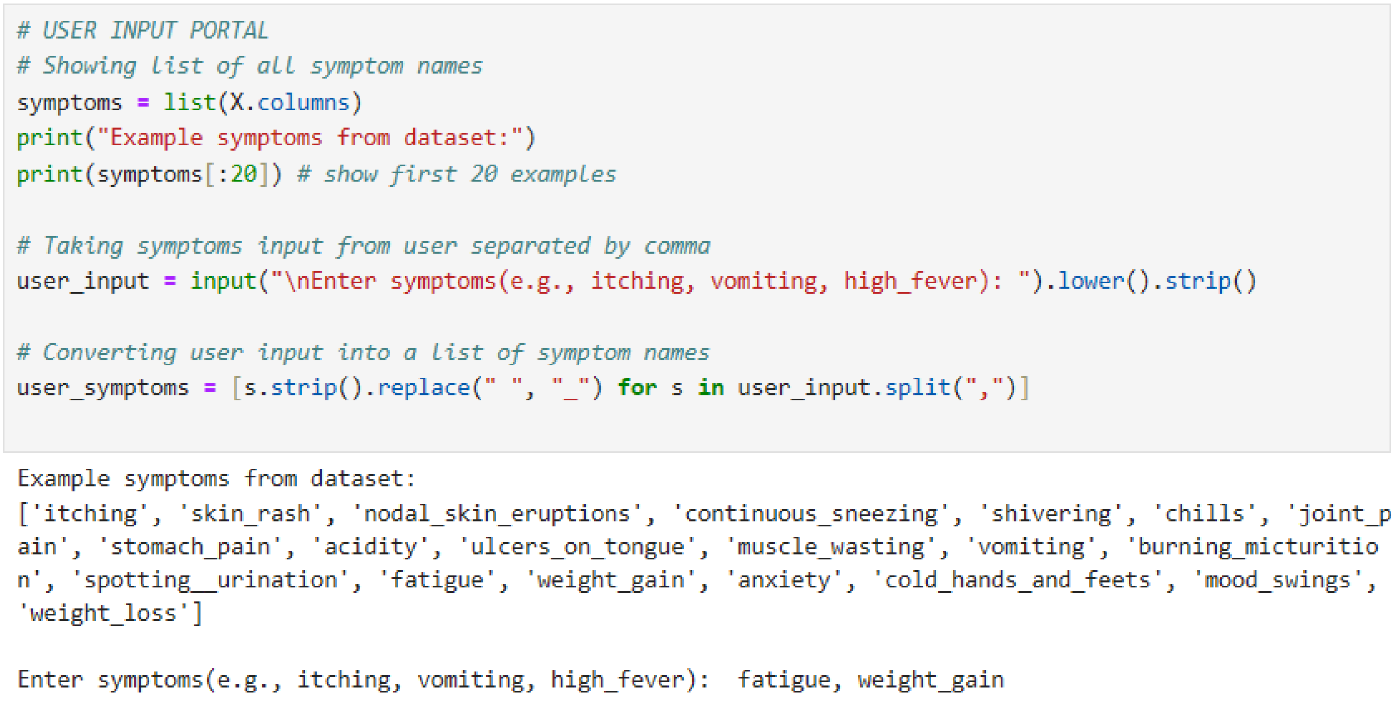


```
Classification Report:

                                        precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00         1
                                   AIDS       1.00      1.00      1.00         1
                                   Acne       1.00      1.00      1.00         1
                    Alcoholic hepatitis       1.00      1.00      1.00         2
                                Allergy       1.00      1.00      1.00         1
                              Arthritis       1.00      1.00      1.00         1

                              Psoriasis       1.00      1.00      1.00         1
                           Tuberculosis       1.00      1.00      1.00         2
                                Typhoid       1.00      1.00      1.00         2
                Urinary tract infection       1.00      1.00      1.00         1
                         Varicose veins       1.00      1.00      1.00         2
                            hepatitis A       1.00      1.00      1.00         2

                               accuracy                           1.00        61
                              macro avg       1.00      1.00      1.00        61
                           weighted avg       1.00      1.00      1.00        61
```

Figure 23: Model evaluation result.

Step 12: User Input

Finally, this system takes symptoms from the users which will predict the disease. Symptoms examples are also given so that user do not get confused on how to input symptoms.



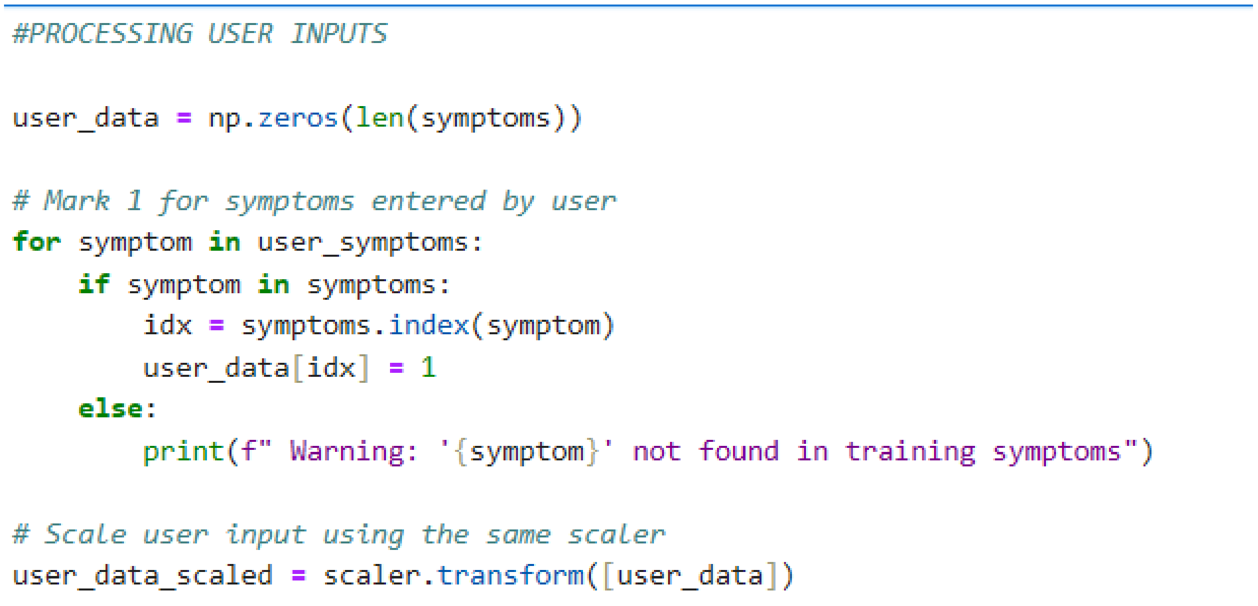
```python
# USER INPUT PORTAL
# Showing list of all symptom names
symptoms = list(X.columns)
print("Example symptoms from dataset:")
print(symptoms[:20]) # show first 20 examples

# Taking symptoms input from user separated by comma
user_input = input("\nEnter symptoms(e.g., itching, vomiting, high_fever): ").lower().strip()

# Converting user input into a list of symptom names
user_symptoms = [s.strip().replace(" ", "_") for s in user_input.split(",")]
```

```
Example symptoms from dataset:
['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shivering', 'chills', 'joint_p
ain', 'stomach_pain', 'acidity', 'ulcers_on_tongue', 'muscle_wasting', 'vomiting', 'burning_micturitio
n', 'spotting__urination', 'fatigue', 'weight_gain', 'anxiety', 'cold_hands_and_feets', 'mood_swings',
'weight_loss']

Enter symptoms(e.g., itching, vomiting, high_fever):  fatigue, weight_gain
```

Figure 24: System taking User inputs.

Step 13: Processing user inputs

The user inputted symptoms then are stored on user_data a binary feature vector. Then if a symptom exists in the training list symptoms, it finds its index and sets that position to 1 (“present”) else prints a warning message.



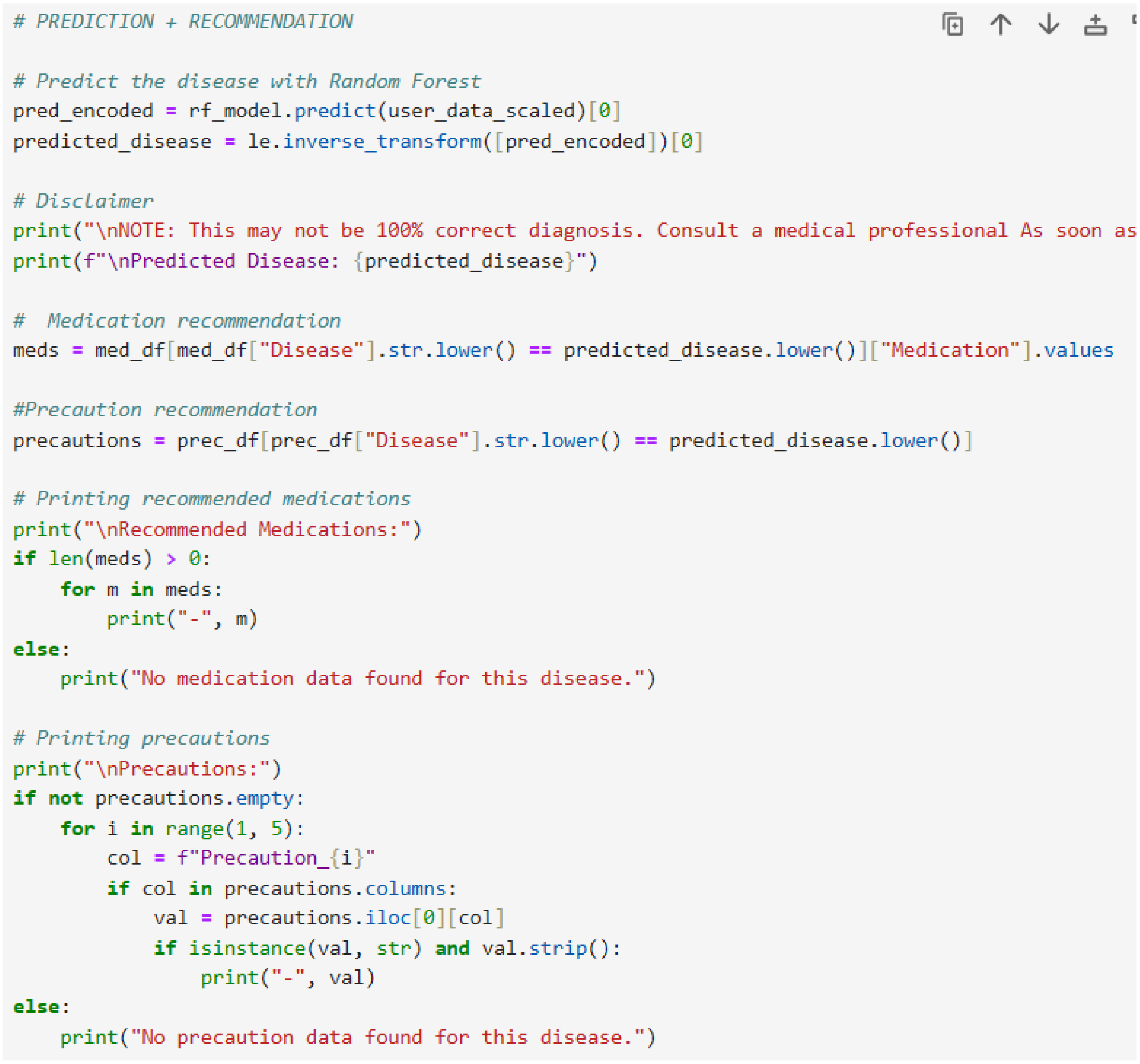
```
#PROCESSING USER INPUTS

user_data = np.zeros(len(symptoms))

# Mark 1 for symptoms entered by user
for symptom in user_symptoms:
    if symptom in symptoms:
        idx = symptoms.index(symptom)
        user_data[idx] = 1
    else:
        print(f" Warning: '{symptom}' not found in training symptoms")

# Scale user input using the same scaler
user_data_scaled = scaler.transform([user_data])
```

Figure 25: Code to process user inputs.

Step 14: Final Prediction and recommendation

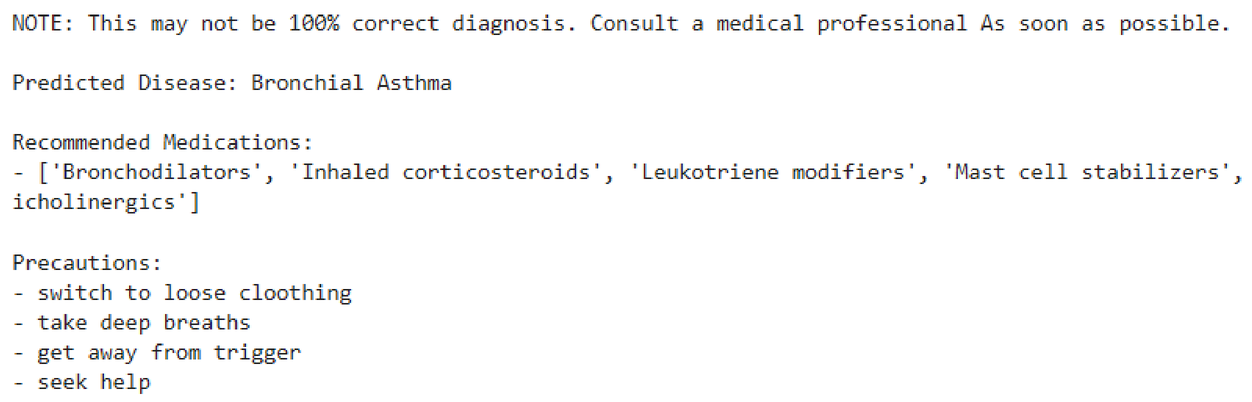
Here the user's processed symptom input is used to train Random Forest model to predict the most likely disease. After showing a medical disclaimer and the predicted disease, it looks up that disease in our datasets to recommend medicines and precautions steps.

It the disease is not found on the datasets, a message saying no medication is found will be shown.



```
# PREDICTION + RECOMMENDATION

# Predict the disease with Random Forest
pred_encoded = rf_model.predict(user_data_scaled)[0]
predicted_disease = le.inverse_transform([pred_encoded])[0]

# Disclaimer
print("\nNOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as
print(f"\nPredicted Disease: {predicted_disease}")

#  Medication recommendation
meds = med_df[med_df["Disease"].str.lower() == predicted_disease.lower()]["Medication"].values

#Precaution recommendation
precautions = prec_df[prec_df["Disease"].str.lower() == predicted_disease.lower()]

# Printing recommended medications
print("\nRecommended Medications:")
if len(meds) > 0:
    for m in meds:
        print("-", m)
else:
    print("No medication data found for this disease.")

# Printing precautions
print("\nPrecautions:")
if not precautions.empty:
    for i in range(1, 5):
        col = f"Precaution_{i}"
        if col in precautions.columns:
            val = precautions.iloc[0][col]
            if isinstance(val, str) and val.strip():
                print("-", val)
else:
    print("No precaution data found for this disease.")
```

Figure 26: Code to predict disease, recommend medicines and precautions.

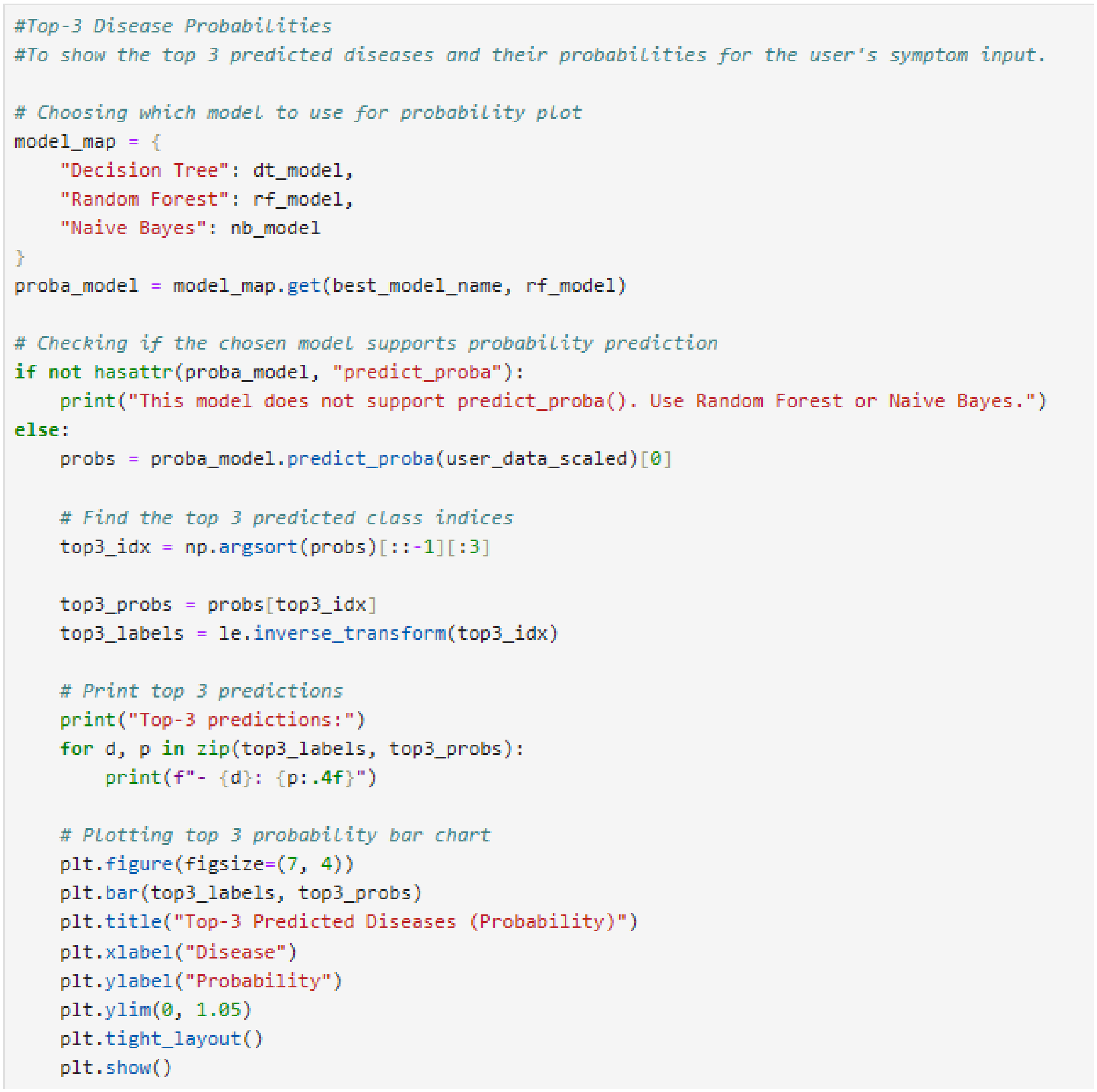


```
NOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as possible.

Predicted Disease: Bronchial Asthma

Recommended Medications:
- ['Bronchodilators', 'Inhaled corticosteroids', 'Leukotriene modifiers', 'Mast cell stabilizers',
icholinergics']

Precautions:
- switch to loose cloothing
- take deep breaths
- get away from trigger
- seek help
```

Figure 27: System predicting disease, recommending meds and precautions.

Step 15: Disease Probability

At the end, a top 3 disease probability calculations were carried out just to add some more visualizations in our project. Then from those calculation data a bar graph was visualized.



```python
#Top-3 Disease Probabilities
#To show the top 3 predicted diseases and their probabilities for the user's symptom input.

# Choosing which model to use for probability plot
model_map = {
    "Decision Tree": dt_model,
    "Random Forest": rf_model,
    "Naive Bayes": nb_model
}
proba_model = model_map.get(best_model_name, rf_model)

# Checking if the chosen model supports probability prediction
if not hasattr(proba_model, "predict_proba"):
    print("This model does not support predict_proba(). Use Random Forest or Naive Bayes.")
else:
    probs = proba_model.predict_proba(user_data_scaled)[0]

    # Find the top 3 predicted class indices
    top3_idx = np.argsort(probs)[::-1][:3]

    top3_probs = probs[top3_idx]
    top3_labels = le.inverse_transform(top3_idx)

    # Print top 3 predictions
    print("Top-3 predictions:")
    for d, p in zip(top3_labels, top3_probs):
        print(f"- {d}: {p:.4f}")

    # Plotting top 3 probability bar chart
    plt.figure(figsize=(7, 4))
    plt.bar(top3_labels, top3_probs)
    plt.title("Top-3 Predicted Diseases (Probability)")
    plt.xlabel("Disease")
    plt.ylabel("Probability")
    plt.ylim(0, 1.05)
    plt.tight_layout()
    plt.show()
```

Figure 28: Code to predict disease probability.

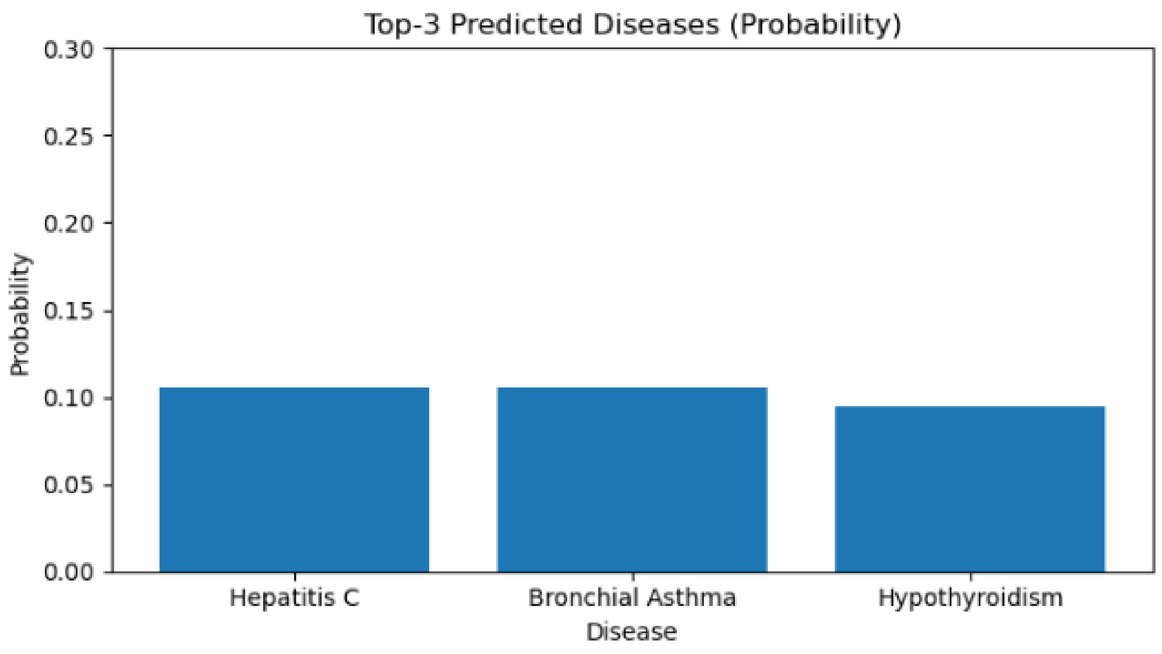


Figure 29: Bar graph of probability for top 3 predicted diseases.

3.6 Development Platforms and Tools:

Platform: Anaconda Navigator was the platform where I have developed and implemented the project.

Figure 30: Anaconda Navigator logo.

Development tool: Jupyter Notebook was used as a code editor. It provided me interactive interfaces for visualizing the results.

Figure 31: Jupyter Notebook logo.

Language: Python was the primary language for my project development. It contains large library ecosystem which was helpful while developing my project.

Figure 32: Python's logo.

These platforms and tools helped me complete my project perfectly and according to my academic course.

Libraries used:

- **Pandas**: It was mainly used to load and clean my datasets. They helped me organize data in a tabular form.
- **NumPy**: Mainly NumPy was used for creating symptom input vectors which are 0 and 1s.
- **Scikit-learn:** This library provided me machine learning tools to scale data, convert labels to numbers, train models, encoding, evaluate the model performances.
- **Matplotlib:** It was used to draw graphs, pie charts and other visualizations

These libraries made sure to fulfil each and every requirement of my coursework throughout the development phase. At the end. they were helpful to build a robust, effective and efficient system to predict disease and recommend medicines and precautions.

3.7 Achieved results

Model performance:

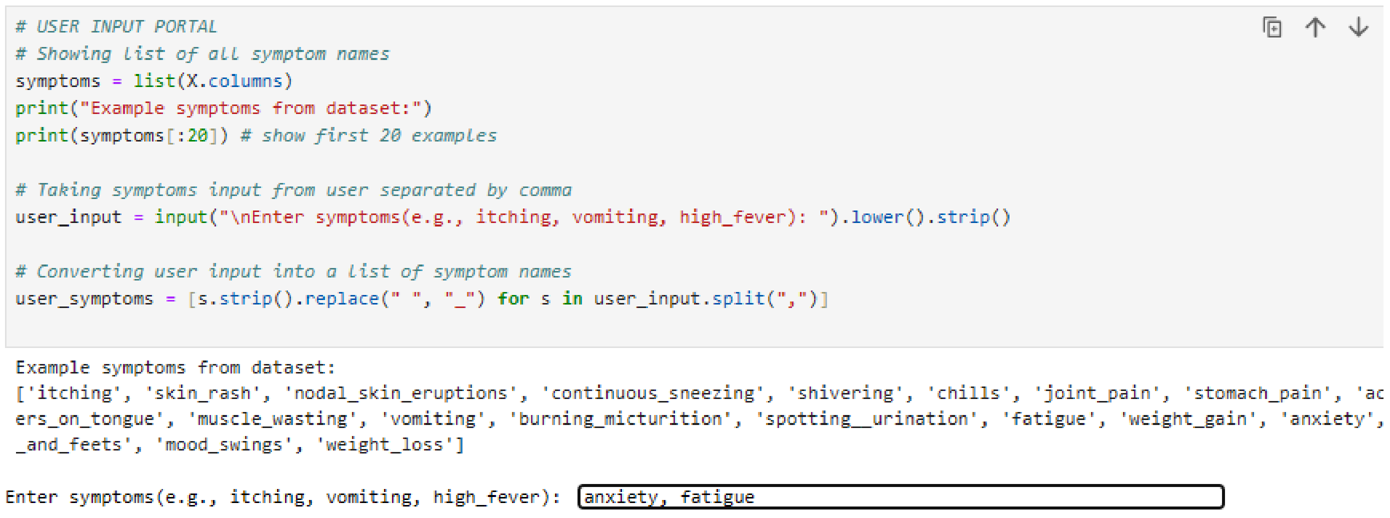
- **Random Forest** = 1.0000 (best)
- **Naive Bayes** = 0.9672
- **Decision Tree** = 0.7049

So, Random Forest was selected as the best and final model.

Final System Output:

The developed system is capable of taking symptoms form the users, then predict the possible disease. Then again the system recommends the medicines and precaution measures for the predicted disease.

System taking symptoms as input from user:

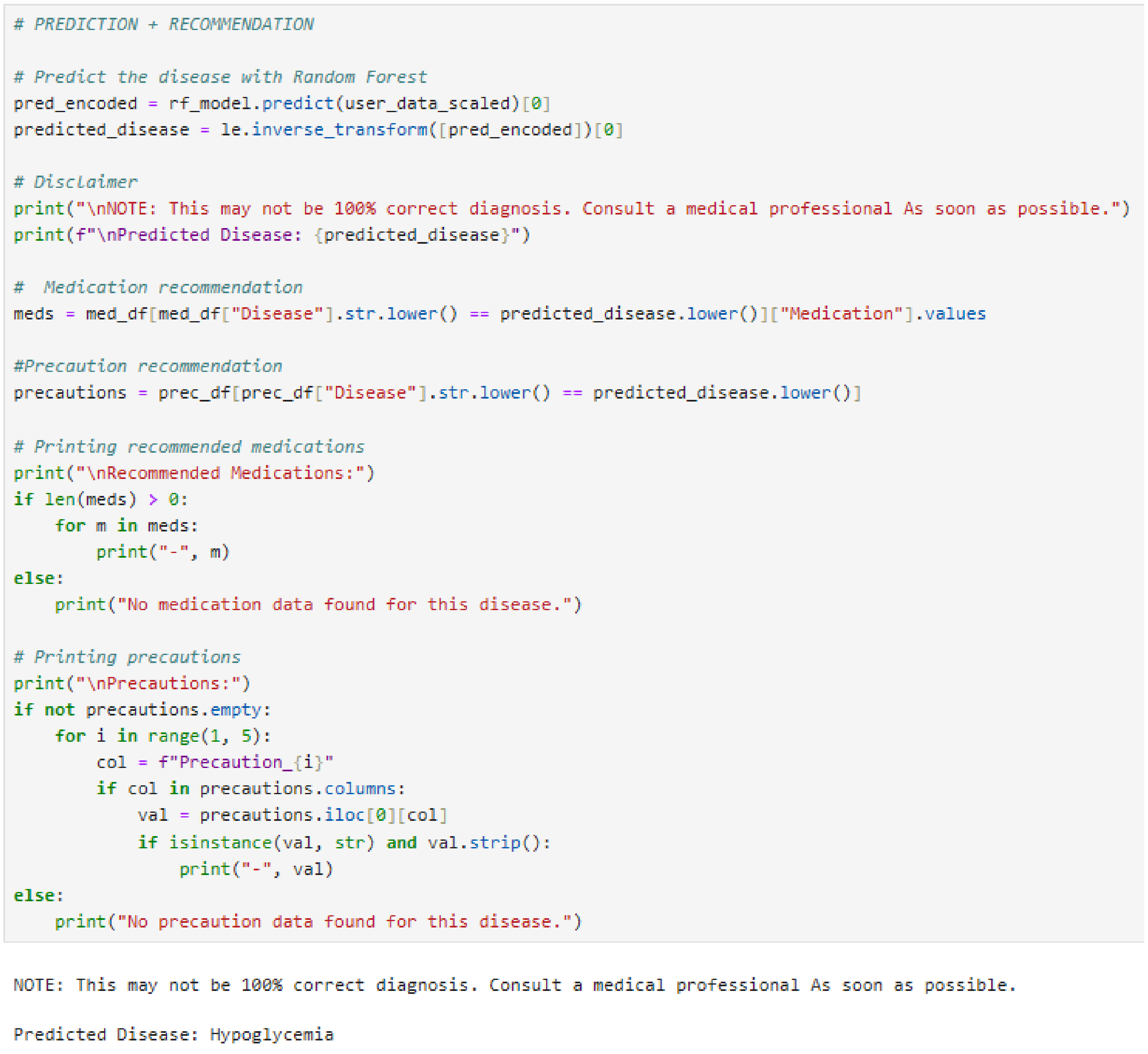


```
# USER INPUT PORTAL
# Showing list of all symptom names
symptoms = list(X.columns)
print("Example symptoms from dataset:")
print(symptoms[:20]) # show first 20 examples

# Taking symptoms input from user separated by comma
user_input = input("\nEnter symptoms(e.g., itching, vomiting, high_fever): ").lower().strip()

# Converting user input into a list of symptom names
user_symptoms = [s.strip().replace(" ", "_") for s in user_input.split(",")]
```

```
Example symptoms from dataset:
['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing', 'shivering', 'chills', 'joint_pain', 'stomach_pain', 'ac
ers_on_tongue', 'muscle_wasting', 'vomiting', 'burning_micturition', 'spotting__urination', 'fatigue', 'weight_gain', 'anxiety',
_and_feets', 'mood_swings', 'weight_loss']

Enter symptoms(e.g., itching, vomiting, high_fever): anxiety, fatigue
```

Figure 33: System successfully taking users symptoms as inputs.

System predicting the possible disease:

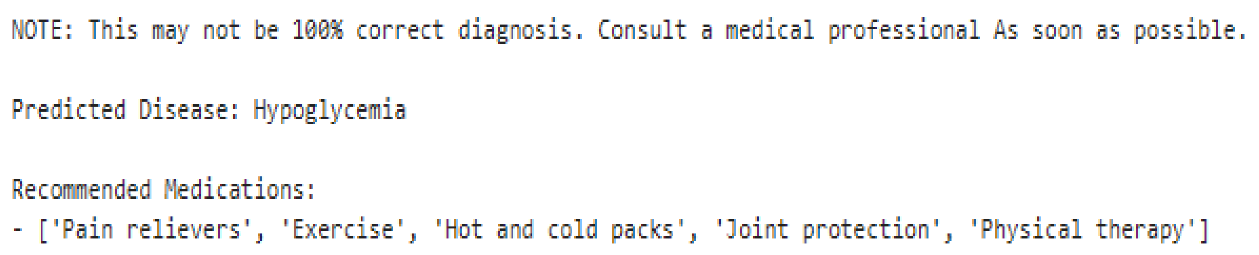


```
# PREDICTION + RECOMMENDATION

# Predict the disease with Random Forest
pred_encoded = rf_model.predict(user_data_scaled)[0]
predicted_disease = le.inverse_transform([pred_encoded])[0]

# Disclaimer
print("\nNOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as possible.")
print(f"\nPredicted Disease: {predicted_disease}")

#  Medication recommendation
meds = med_df[med_df["Disease"].str.lower() == predicted_disease.lower()]["Medication"].values

#Precaution recommendation
precautions = prec_df[prec_df["Disease"].str.lower() == predicted_disease.lower()]

# Printing recommended medications
print("\nRecommended Medications:")
if len(meds) > 0:
    for m in meds:
        print("-", m)
else:
    print("No medication data found for this disease.")

# Printing precautions
print("\nPrecautions:")
if not precautions.empty:
    for i in range(1, 5):
        col = f"Precaution_{i}"
        if col in precautions.columns:
            val = precautions.iloc[0][col]
            if isinstance(val, str) and val.strip():
                print("-", val)
else:
    print("No precaution data found for this disease.")
```

```
NOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as possible.

Predicted Disease: Hypoglycemia
```

Figure 34: System successfully predicting possible disease.

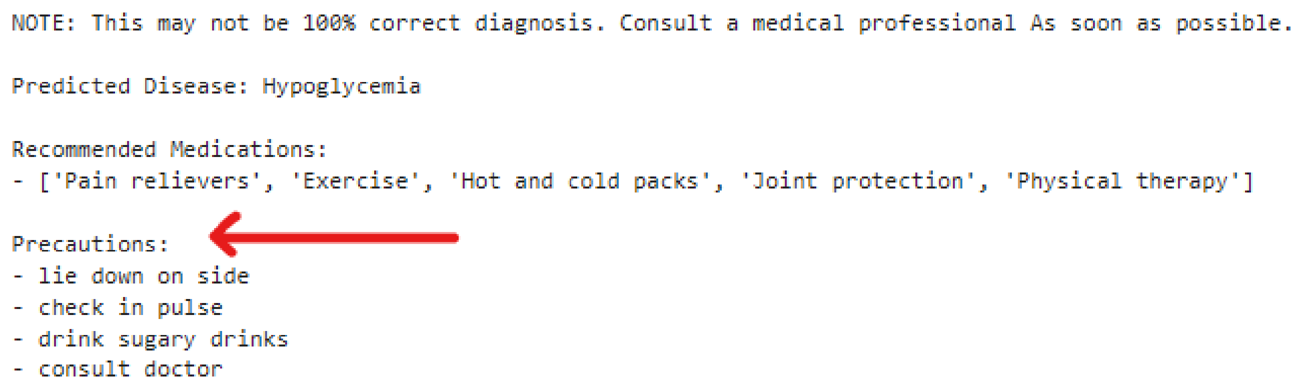
System recommending medicines:



```
NOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as possible.

Predicted Disease: Hypoglycemia

Recommended Medications:
- ['Pain relievers', 'Exercise', 'Hot and cold packs', 'Joint protection', 'Physical therapy']
```

Figure 35: System successfully recommending Meds for predicted disease.

System giving precautions to users:



```
NOTE: This may not be 100% correct diagnosis. Consult a medical professional As soon as possible.

Predicted Disease: Hypoglycemia

Recommended Medications:
- ['Pain relievers', 'Exercise', 'Hot and cold packs', 'Joint protection', 'Physical therapy']

Precautions:
- lie down on side
- check in pulse
- drink sugary drinks
- consult doctor
```

Figure 36: System successfully recommending 4 precautions for predicted disease.

4. Conclusion:

4.1 Analyzing the work done:

The purposed solution focuses on the development of a ML based system which help in possible disease prediction based on user provided symptoms and then recommending suitable medicines and preventive measures also. The main aim of this system is to provide early guidance and reduce confusion caused by lack of medical professionals and blind faiths in our society but not to give final medical verdict. By using Supervised Learning practices, our system learns and trains from historical medical data where the provided symptoms are already linked to known diseases.

This solution utilizes Machine Learning algorithms like Naïve Bayes, Decision Trees and Random Forests to perform disease classifications. There above algorithms are very capable for these tasks because they can handle structures symptom data and produces clear disease as predictions. It includes important steps such as data preprocessing, feature encoding, model training and evaluation using performance measures as accuracy. And at the end, by selecting the best performing model, our system will ensure a possible disease prediction before recommending medicines and preventive measures.

It is also important to know that this platform is not intended to replace doctors or medical professionals. Even after possible disease prediction and medicine recommendation, consulting a qualified doctor is very important. This system is rather developed also to work along with professionals in medical pharmacies and in hospitals by qualified personalities.

4.2 Solution Addressing the Real World problem:

In countries like our, Nepal, the access to healthcare is limited because of many challenges; like geographical context, shortage of medicines and medical professionals and lack of awareness. Many people rely in delayed treatment, blind faiths, Jhakri rather than taking medical help which increases health related risks. To tackle these type of challenges, our system can play an important role and provide structured way to get initial health guidance based on user's symptoms.

This system can also be useful in multiple real-world cases:

Early Awareness and Guidance:

The system helps users understand what disease their symptoms may indicate at an early stage. This awareness encourages people to seek medical help sooner instead of ignoring symptoms or relying on misinformation.

Accessibility:

In rural areas where getting medical help needs days, our system can predict the possible disease from patient's symptoms at any time at anywhere.

Public Awareness:

Using this system can make people educated about using medicine, consulting doctors etc. rather believing traditional measures and blind faiths.

For medical Professionals:

Our system can be utilized by doctors, pharmacists and other medical professionals for preliminary guidance and diagnosis.

Reducing Healthcare Load:

This system also reduces unnecessary hospital visits for minor or common conditions which allows medical professionals to focus on serious cases.

Consistency and Reliability:

Sometimes, human evaluation can vary on experience or workload but our model provides consistent predictions based on trained medical data which reduces random or biased decisions.

Overall, this system acts as a supportive tool which helps users to make more informed decisions while also encouraging professional medical counselling as important.

4.3 Further Work:

Although this system shows promising results however it is still incomplete and many improvements can be made in future. Some of them are given below:

Validation: Collaborating with medical professionals and Institutions to validate and certify our system.

Real-Time Application: Deploying our system in a mobile app platform will insure to our reach to large individuals for real time use.

Expanded data: We can always increase more data of disease, symptoms and medications.

Language Accessibility: we can modify our system to make them support Nepali and other local language by which the users can understand us better.

To conclude, this proposed medicine recommendation system can serve as a supportive healthcare tool that helps both users and medical professionals make best possible decisions. I can vouch that this system can contribute to better healthcare accessibility while also respecting the roles of doctors and experts.

5. References

Belcic, I., 2025. *What is a flowchart?.* [Online]
Available at: https://www.ibm.com/think/topics/flowchart
[Accessed December 2025].

Mahr, N., 2025. *Pseudocode in Programming* **|** *Definition, Examples & Advantage.* [Online]
Available at: https://study.com/learn/lesson/pseudocode-examples-what-is-pseudocode.html
[Accessed December 2025].