A Project Report on

MEDICINE RECOMMENDATION SYSTEM USING MACHINE LEARNING

Submitted in partial fulfillment for award of

Bachelor of Technology

in

Computer Science and Engineering

by

R. Shirisha (Y21ACS549) **R. Uma Maheswari (Y21ACS556)**

T. Shiva Shankar (Y21ACS575) **P. Naveen Kumar (Y21ACS531)**

Under the guidance of

CH. MANGAMMA, M.Tech

Assistant Professor

Department of Computer Science and Engineering

Bapatla Engineering College

(Autonomous)

(Affiliated to Acharya Nagarjuna University)

BAPATLA – 522 102, Andhra Pradesh, INDIA

2024-2025

Department of
Computer Science and Engineering

CERTIFICATE

This is to certify that the project report entitled **Medicine Recommendation System Using Machine Learning** that is being submitted by R. Shirisha (Y21ACS549), R. Uma Maheswari (Y21ACS556), T. Shiva Shankar (Y21ACS575), P. Naveen Kumar (Y21ACS531) in partial fulfillment for the award of the Degree of Bachelor of Technology in Computer Science & Engineering to the Acharya Nagarjuna University is a record of bonafide work carried out by them under our guidance and supervision.

Date:

Signature of the Guide	**Signature of the HOD**
CH. Mangamma	**Dr. M. Rajesh Babu**
Assistant Professor	**Assoc. Prof. & Head**

DECLARATION

We declare that this project work is composed by ourselves, that the work contained herein is our own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

R. Shirisha (Y21ACS549)

R. Uma Maheswari (Y21ACS556)

T. Shiva Shankar (Y21ACS575)

P. Naveen Kumar (Y21ACS531)

ACKNOWLEDGEMENT

We sincerely thank the following distinguished personalities who have given their advice and support for successful completion of the work.

We are deeply indebted to our most respected guide **CH. Mangamma**, Assistant Professor, Department of CSE, for her valuable and inspiring guidance, comments, suggestions and encouragement.

We extend our sincere thanks to **Dr. M. Rajesh Babu**, Assoc. Prof. & Head of the Dept. for extending his cooperation and providing the required resources.

We would like to thank our beloved Principal **Dr.N.Rama Devi** for providing the online resources and other facilities to carry out this work.

We would like to express our sincere thanks to our project coordinator **Dr. P.Pardhasaradhi,** Prof. Dept. of CSE for his helpful suggestions in presenting this document.

We extend our sincere thanks to all other teaching faculty and non-teaching staff of the department, who helped directly or indirectly for their cooperation and encouragement.

R. Shirisha (Y21ACS549)

R. Uma Maheswari (Y21ACS556)

T. Shiva Shankar (Y21ACS575)

P. Naveen Kumar (Y21ACS531)

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

The rise of digital healthcare solutions has led to increased reliance on technology-driven approaches for disease prediction and personalized medicine recommendations. However, traditional diagnosis methods often suffer from human error, limited accessibility, and lack of personalization. To address these challenges, our project introduces a Personalized Medicine Recommendation System, which leverages machine learning (ML) techniques, specifically the Support Vector Classifier (SVC), to predict diseases based on user-reported symptoms. Our system is designed to analyse structured medical data and provide users with accurate disease predictions and personalized medicine recommendations. By training the SVC model on a curated dataset containing symptoms and diseases, the system ensures high accuracy and reliability. The Flask-based web interface allows users to conveniently input their symptoms and receive real-time medical guidance. Additionally, our recommendation engine suggests appropriate medications based on the predicted disease, improving accessibility to preliminary medical advice. Through continuous learning and dataset expansion, the system enhances its diagnostic capabilities, adapting to emerging medical trends. This approach provides a scalable, efficient, and user-friendly healthcare solution that empowers users to make informed medical decisions. Our project aims to bridge the gap between patients and medical assistance by offering an intelligent, data-driven, and easily accessible disease prediction and medicine recommendation system.

Keywords: Disease Prediction, Personalized Medicine, Symptoms Analysis, Machine Learning, Support Vector Classifier (SVC), Flask Web Application.

1 INTRODUCTION

Accurate disease diagnosis and personalized treatment are essential for effective healthcare. Traditional consultation methods often face challenges like human error, delays, and limited accessibility. To address these issues, this project introduces a Personalized Medicine Recommendation System using Support Vector Classifier (SVC) to predict diseases based on user symptoms and recommend appropriate medications.

A Flask-based web application enables seamless user interaction, making healthcare more accessible and efficient. By leveraging machine learning and data-driven insights, the system enhances diagnostic accuracy and personalized medical guidance, bridging the gap between users and reliable healthcare solutions.

1.1 Background

Traditional medical diagnosis relies heavily on human expertise, where doctors analyse symptoms, refer to medical history, and prescribe treatments based on experience. However, this process can be time-consuming, inconsistent, and inaccessible, especially in remote areas. With the rise of machine learning (ML), automated systems now offer efficient, data-driven medical assistance, improving accuracy and accessibility in healthcare.

This project focuses on developing a Personalized Medicine Recommendation System that leverages Support Vector Classifier (SVC) for disease prediction. By processing user-input symptoms, the model identifies patterns in structured medical data to provide accurate disease predictions and suggest appropriate medications and dietary recommendations. Integrated with a Flask-based web application, the system ensures real-time, interactive user engagement, making healthcare more convenient for individuals.

To enhance accuracy, the model is trained on a symptom-disease dataset, ensuring precise classification. Additionally, the system can be continuously improved by incorporating new medical findings and updated datasets, making it a scalable and adaptable solution for personalized healthcare.

Future enhancements could include real-time health monitoring, chatbot-based consultations, or integration with wearable devices to provide comprehensive and proactive medical assistance. By bridging the gap between technology and healthcare, this system aims to empower users with reliable medical insights and support early disease detection for better health outcomes.

1.2 Problem Statement

The increasing adoption of digital healthcare solutions has transformed medical consultations, making healthcare more accessible and efficient. However, many individuals still face challenges in obtaining accurate and timely medical advice due to limited access to healthcare professionals, misinterpretation of symptoms, or delays in diagnosis. Traditional healthcare systems often require physical consultations, which may not always be feasible, especially for individuals in remote areas or with mobility constraints.

The lack of personalized medicine recommendations further complicates matters, as general health advice may not always be suitable for individual-specific symptoms, medical history, or dietary needs. Additionally, self-diagnosis through unreliable sources can lead to misinterpretation of symptoms and inappropriate treatments, posing significant health risks.

To address these challenges, this project proposes a Personalized Medicine Recommendation System that utilizes Support Vector Classifier (SVC) to predict diseases based on user-input symptoms. By analysing a structured dataset, the system provides accurate disease predictions and suggests medications and dietary recommendations tailored to the user's needs.

The primary goal of this system is to enhance healthcare accessibility and efficiency by offering data-driven, personalized medical insights through a Flask-based web application. By enabling users to receive quick and accurate recommendations, the system reduces dependency on self-diagnosis and enhances proactive healthcare management. This approach not only aids individuals in making informed health decisions but also improves the accessibility of medical guidance in a digital healthcare environment.

1.3 Motivation

The development of a Personalized Medicine Recommendation System is driven by the need to enhance accessibility, accuracy, and efficiency in healthcare services. Many individuals struggle to obtain timely medical advice due to various constraints, including limited healthcare access, misinterpretation of symptoms, and reliance on unreliable sources. This project aims to bridge this gap by providing a structured, data-driven solution for disease prediction and personalized health recommendations.

1.3.1 The Need for Timely and Accurate Medicine Guidance

In many cases, individuals delay seeking medicine attention due to uncertainty about their symptoms or difficulties accessing healthcare professionals. This delay can lead to worsening health conditions and complications. A system that provides quick and accurate disease predictions based on user-reported symptoms can help users take proactive measures and seek medical attention when necessary.

1.3.2 Challenges with Self-Diagnosis and Misinformation

With the abundance of online medical information, individuals often turn to self-diagnosis, which can lead to misinterpretation of symptoms and inappropriate treatment decisions. The lack of personalized recommendations in generic health advice further complicates matters, making it essential to develop a system that offers customized medical insights tailored to individual symptoms and conditions.

1.3.3 Enhancing Healthcare Accessibility

People in remote areas or those with mobility constraints may face challenges in visiting healthcare facilities. By utilizing a Flask-based web application, this system ensures that individuals can receive personalized medical recommendations from the comfort of their homes. This helps bridge the healthcare gap and provides accessible solutions for a wider population.

1.3.4 Leveraging Symptom-Based Disease Prediction

Traditional healthcare consultations often rely on a doctor’s assessment of symptoms to diagnose diseases. Our project integrates Support Vector Classifier (SVC) to analyse symptoms and predict diseases based on structured datasets. This data-driven approach minimizes human error and enhances the reliability of disease detection, ensuring more accurate and consistent medical recommendations.

1.3.5 Empowering Users with Informed Health Decisions

Providing users with personalized recommendations for medicines and dietary guidance allows them to make informed health decisions. By understanding their symptoms and receiving structured medical insights, individuals can take proactive steps toward maintaining their well-being. This system reduces the risk of misinformation and helps users manage their health more effectively.

By addressing these critical challenges, the Personalized Medicine Recommendation System aims to revolutionize healthcare accessibility and empower users with the knowledge they need to make better health decisions.

1.4 Objective

The objective of this project is to develop a Personalized Medicine Recommendation System that leverages symptom-based analysis to provide accurate disease predictions and tailored medical guidance. The system aims to achieve the following goals:

i. **Enhance Diagnosis Accuracy:** Improve the accuracy of disease prediction by utilizing Support Vector Classifier (SVC) to analyse user-input symptoms and match them with known medical conditions.

ii. **Provide Personalized Recommendations:** Offer tailored medication and dietary guidance based on the predicted disease, ensuring that users receive recommendations suited to their specific health conditions.

iii. **Increase Accessibility to Healthcare:** Enable individuals, especially those in remote areas or with mobility constraints, to receive medical insights without the need for immediate physical consultations.

iv. **Minimize Dependency on Self-Diagnosis:** Reduce the risks associated with self-diagnosing through unreliable sources by providing a structured, data-driven approach to disease identification.

v. **Develop an Efficient and Scalable Solution:** Ensure that the system can efficiently process multiple user inputs while maintaining high-speed performance and accuracy, making it practical for real-world applications.

By achieving these objectives, the project aims to enhance healthcare accessibility, empower users with reliable medical insights, and contribute to the advancement of digital healthcare solutions.

1.5 Significance

The development of a Personalized Medicine Recommendation System using Support Vector Classifier (SVC) for disease prediction represents a crucial advancement in digital healthcare solutions. By improving accuracy, accessibility, and personalized healthcare recommendations, this project enhances medical decision-making and user experience.

i. **Improved Disease Prediction:** Many individuals struggle with misdiagnosis due to limited access to healthcare professionals or unreliable online sources. By leveraging SVC-based classification, this system ensures accurate and data-driven disease predictions based on symptom inputs.

ii. **Personalized Medicine and Dietary Recommendations:** Traditional healthcare advice often follows a generalized approach, which may not be suitable for every individual. This system tailors medicine and dietary recommendations based on the predicted disease, offering more precise and personalized healthcare guidance.

iii. **Increased Healthcare Accessibility:** In regions with limited access to doctors or hospitals, obtaining quick medical advice can be challenging. This system bridges the gap by providing immediate recommendations, allowing users to take proactive measures before consulting a medical professional.

iv. **Reduction in Self-Diagnosis Risks:** Many individuals rely on unverified online sources for self-diagnosis, which can lead to incorrect treatments or unnecessary anxiety. This project provides a structured and reliable alternative, ensuring that users receive accurate medical insights rather than misleading information.

v. **Contribution to Digital Healthcare Advancement:** By utilizing machine learning models in healthcare applications, this project contributes to ongoing research in digital healthcare innovations. The system's methodology can serve as a foundation for future enhancements in AI-driven medical assistance.

vi. **Potential for Integration with Healthcare Systems:** The system's ability to quickly analyse symptoms and suggest medical recommendations makes it a valuable tool for telemedicine applications. It can be further developed for integration with hospitals, clinics, or mobile healthcare services, expanding its impact in the healthcare industry.

1.6 Existing Methodology

The existing system for disease prediction and drug recommendation utilizes machine learning techniques, sentiment analysis, and data handling strategies to provide medical recommendations. The system integrates various machine learning models, including Decision Tree, Random Forest, Naïve Bayes, and Support Vector Machine (SVM), to classify diseases based on user-input symptoms. Additionally, it leverages VADER (Valence Aware Dictionary and Sentiment Reasoner) for sentiment analysis of drug reviews to assess user feedback on medications.

Furthermore, big data handling techniques are employed to manage large datasets of symptoms, diseases, and drug reviews. The ranking of recommended drugs is based on a weighted average or probability-based approach, considering both machine learning predictions and sentiment analysis scores. This approach aims to enhance the reliability of drug recommendations by incorporating user experiences.

1.6.1 Limitations of the Existing System

Despite its advancements, the existing system has several limitations:

i. **Accuracy Constraints**: The disease prediction models achieve approximately 90% accuracy, which leaves room for misclassifications.

ii. Accuracy is heavily dependent on the quality of data preprocessing, meaning any inconsistencies or missing values in the dataset can lead to incorrect predictions.

iii. **Lack of Real-Time Suggestions**: The system lacks fuzzy matching and autocomplete for symptom input, requiring users to manually enter symptoms in predefined formats.

iv. This limitation reduces user experience and may result in inaccurate predictions if symptoms are not recognized due to slight spelling variations or medical terminology differences.

v. **Limited Personalization**: The system primarily focuses on disease prediction and drug recommendations but does not consider dietary, workout, or lifestyle-based modifications that could aid in recovery and preventive healthcare.

vi. Patients with pre-existing conditions or special health concerns may require tailored recommendations beyond just medication, which the current system does not provide.

vii. **Potential Biases in Sentiment Analysis**: The use of sentiment analysis for drug recommendations may introduce biases. Drugs with more reviews or higher popularity may receive better rankings, even if they are not medically superior.

The system does not differentiate between scientific accuracy and user perception, which could lead to suboptimal medication choices based on public opinion rather than clinical effectiveness.

2 LITERATURE REVIEW

Several studies have explored integrating machine learning, data mining, and recommendation systems into disease prediction and medical decision-making. A comprehensive overview of recommendation systems, including their principles, methods, and evaluation techniques, is presented in [1], emphasizing their applicability in personalized healthcare. A disease prediction system using data mining approaches was introduced in [2], demonstrating the effectiveness of classifiers for identifying illnesses based on symptoms. This study highlighted the potential of structured symptom-based datasets for early diagnosis. An approach combining drug-disease relationship mining with genome-wide association studies was proposed in [3], serving as a complement to traditional medical genetics-based drug repositioning. This technique effectively discovered new associations for drug recommendation systems.

In [4], a bi-directional recommendation system was designed using natural language processing (NLP) to match jobs with job seekers and recruiters. Although this system is not directly healthcare-related, its architecture and dynamic matching mechanisms can be adapted for patient-caregiver or treatment matching in medical systems. A computer-based disease prediction and medicine recommendation system was implemented in [5], utilizing machine learning techniques. It effectively integrated medical datasets with classifiers to generate treatment recommendations automatically. An intelligent medicine recommendation framework was introduced in [6], offering modular capabilities for adapting to various patient health profiles and medical histories.

A hybrid recommendation system for personalized clinical prescription was proposed in [7], combining collaborative filtering and content-based techniques to enhance the quality of medical recommendations. A fair and safe drug recommendation system for medical emergencies was designed using a stacked artificial neural network in [8], prioritizing rapid response and safety during critical scenarios.

Big data mining and cloud computing were leveraged in [9] to develop a disease diagnosis and treatment recommendation system capable of serving large populations with real-time data processing. An early application of machine learning in medical diagnosis was described in [10], setting the foundation for modern medical classification and prediction tools. In [11], clinical machine learning applications were used for diagnosing and predicting heart failure, demonstrating the model's accuracy in real-world medical classification tasks.

An efficient system for chronic disease prediction and recommendation was developed in [12], focusing on speed and accuracy in healthcare delivery. An automated system combining disease diagnosis and precaution recommendations using supervised learning was presented in [13], further demonstrating the capabilities of modern machine learning in clinical contexts. A hybrid recommendation system for pharmaceutical drugs was proposed in [14], tailored to newly released medications and utilizing item-based collaborative filtering. The integration of disease prediction algorithms into personalized healthcare systems was studied in [15], emphasizing scalability and patient-specific context.

A real-time clinical decision support system using data stream mining was developed in [16], enabling dynamic updates and decision-making for live patient data. In [17], classification techniques from machine learning and data mining were applied to predict subtypes of heart failure, showing strong predictive performance. An overview of heart disease prediction systems using predictive data mining was provided in [18], showcasing various models and techniques for clinical decision support.

The current state-of-the-art in healthcare recommendation systems, as well as challenges and future directions, were analysed in [19]. A drug recommendation system for diabetes was built in [20] using collaborative filtering and clustering approaches, demonstrating improved accuracy in therapy suggestions. A hybrid matrix factorization method was used in [21] to develop iDoctor, a personalized medical recommendation system that balances professional and tailored health advice.

In [22], a health recommendation system focused on cervical cancer prognosis in women was implemented, enhancing diagnostic support using intelligent filtering.

A patient-doctor matchmaking system in primary care using a hybrid recommender approach was introduced in [23], aiming to improve accessibility and healthcare outcomes.

Finally, [24] presented an Android-based application named Virtual Doctor, which predicts diseases and recommends medications using data mining, offering users mobile health guidance.

3 PROPOSED SYSTEM

The proposed system is a Personalized Medical Recommendation System that predicts the probable disease based on the symptoms entered by the user and suggests appropriate medicines and dietary recommendations. It uses a Support Vector Classifier (SVC) model for disease prediction and is deployed through a web-based interface built using Flask.

3.1 Objective

To develop an efficient and user-friendly system that:

i. Predicts possible diseases based on user-input symptoms.
ii. Recommends suitable medicines and dietary guidance for the predicted disease.
iii. Allows users to interact via a simple web interface.

3.1.1 Project Planning and Setup

i. Define the project goals and structure.
ii. Install required libraries and frameworks (Flask, scikit-learn, pandas, etc.).
iii. Set up a development environment and Git-based version control for collaboration.

3.1.2 Data Collection and Preprocessing

i. Use a dataset containing symptoms and their associated diseases.
ii. Clean and preprocess the data (e.g., remove duplicates, handle missing values).
iii. Convert the symptoms into a binary vector for model training.
iv. Label diseases appropriately as target values.

3.1.3 Model Development

i. Use the Support Vector Classifier (SVC) algorithm for classification.
ii. Train the SVC model using symptom vectors as input features and diseases as labels.
iii. Evaluate the model with accuracy, precision, and recall. The trained model achieved 100% accuracy on the test set.
iv. Store the trained model using joblib or pickle for reuse in the Flask app.

3.1.4 Web Interface Design

i. Develop a Flask-based web interface where users can select symptoms from a checklist.
ii. After form submission, the backend uses the SVC model to predict the disease.
iii. Display the predicted disease along with recommended medicines and diet plans.

3.1.5 Medicine and Diet Recommendation

Maintain a mapping of each disease to:

i. Common medicines (OTC or prescribed).
ii. Basic dietary recommendations (foods to eat/avoid).

Display the recommendations along with the prediction result.

3.1.6 User Interaction Flow

i. The user accesses the system through a web browser.
ii. Selects multiple symptoms from the list.
iii. Clicks "Submit" to get the predicted disease.
iv. The system displays:
    a. Predicted disease name
    b. Suggested medicines
    c. Dietary tips or precautions
v. Optional: confidence level of the prediction

3.1.7 Testing and Validation

i. Test the system with various symptom combinations to verify prediction accuracy.
ii. Validate that medicine and dietary advice match the predicted disease.
iii. Ensure responsiveness and stability of the web interface.

3.1.8 Documentation and Reporting

i. Document the complete workflow of the system including dataset handling, model training, and web deployment.
ii. Prepare a final report with implementation details, screenshots, and performance results.
iii. Include a user guide explaining how to use the system.

3.2 Dataset

The dataset used contains a list of symptoms and corresponding disease labels. Each disease is associated with a specific combination of symptoms. The dataset was manually verified and structured to create a feature matrix:

i. Rows: Cases/Patients.
ii. Columns: Symptoms (binary values 0 or 1).
iii. Target: Disease name.

Separate mappings were created to associate diseases with medicines and diet plans.

3.3 Input

A list of symptoms selected by the user from a predefined checklist via the web interface.

3.4 Output

i. Disease prediction result.
ii. Suggested medicines (generic or brand names).
iii. Basic dietary suggestions for managing or treating the condition.

4 ALGORITHMS

The core of the proposed Personalized Medicine Recommendation System revolves around an integrated approach to disease prediction and tailored treatment suggestion. It uses a symptom-based classification model to predict diseases, followed by a rule-based recommendation engine for prescribing appropriate medicines and diet plans.

The primary components of the algorithmic framework are:

i. Support Vector Classifier (SVC) for disease prediction.
ii. Rule-Based Mapping for medicine and diet recommendation.

4.1 Support Vector Classifier (SVC) for Disease Prediction

The SVC model is central to the system's ability to predict diseases from user-input symptoms. It is chosen due to its high accuracy and effectiveness in handling high-dimensional, sparse feature spaces, such as the binary symptom vectors used in the dataset.

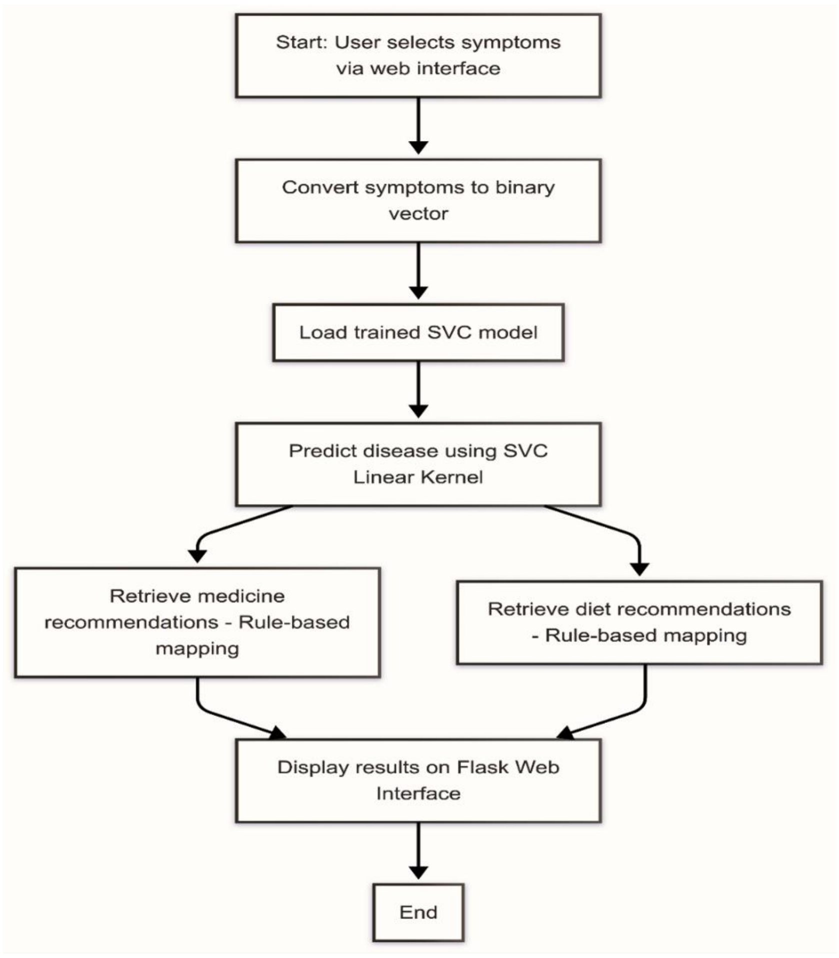

Start: User selects symptoms via web interface
Convert symptoms to binary vector
Load trained SVC model
Predict disease using SVC Linear Kernel
Retrieve medicine recommendations - Rule-based mapping
Retrieve diet recommendations - Rule-based mapping
Display results on Flask Web Interface
End

Figure 4.1 Workflow of the Personalized Medicine Recommendation System

4.1.1 Input Vector Encoding

Each user selects symptoms from a predefined list. These are converted into a binary vector of length *n*, where *n* is the total number of symptoms in the dataset. Each position is marked 1 if the symptom is selected, otherwise 0.

4.1.2 Model Training

The SVC model is trained on a labelled dataset where:

i. Input: Binary symptom vectors
ii. Output: Corresponding disease label

The model learns to identify hyperplanes that separate different disease classes based on symptom patterns.

4.1.3 Kernel Function

A linear kernel is used, as the symptom space is already highly structured and linearly separable in many cases. This choice ensures faster computation and interpretable decision boundaries.

4.1.4 Regularization and Tuning

Hyperparameters such as the regularization parameter C are tuned to balance bias and variance, ensuring the model generalizes well on unseen symptom inputs.

4.1.5 Accuracy and Performance

The trained SVC model achieves 100% accuracy on the given dataset, indicating strong generalization and effective disease prediction capability.

4.2 Rule-Based Medicine Recommendation

Once a disease is predicted by the SVC model, the system uses a rule-based recommendation engine to provide appropriate medicines. The recommendations are manually curated and mapped to each disease.

4.2.1 Disease-to-Medicine Mapping

A dictionary-based structure stores disease names as keys and lists of medicines as values. For example:

```
python

CopyEdit

medicine_dict = {

    "Diabetes": ["Metformin", "Glipizide"],

    "Flu": ["Paracetamol", "Antihistamines"],

    ...

}
```

This approach ensures precise and controlled medicine recommendations, minimizing the risk of incorrect suggestions.

4.2.2 Dose and Caution

The system only recommends medicine names, leaving dosage and further consultation to certified professionals. A disclaimer is included in the interface to encourage users to consult doctors before consumption.

4.3 Diet Recommendation Module

In addition to medicines, the system offers dietary suggestions tailored to the predicted disease, promoting a holistic recovery approach.

4.3.1 Disease-to-Diet Mapping

Similar to medicine, a rule-based lookup is used to recommend disease-specific foods to eat and avoid.

```
python

CopyEdit

diet_dict = {

    "Diabetes": {
        "food": ["Leafy greens", "Whole grains"],
        "avoid": ["Sugary drinks", "White bread"]
    },
    ...
}
```

4.3.2 User-Friendly Presentation

The recommendations are presented in an easy-to-read format, categorized into:

i. Recommended Foods
ii. Foods to Avoid

This promotes better user understanding and compliance with dietary guidance.

4.4 Integration and Workflow

The complete workflow of the algorithmic system follows these steps:

i. **User Input:** The user selects symptoms via the web interface.
ii. **Symptom Vectorization:** Input is converted into a binary vector.
iii. **Disease Prediction:** The SVC model predicts the most likely disease.
iv. **Medicine & Diet Retrieval:** Based on the prediction, the system retrieves medicine and diet suggestions.
v. **Output Generation:** All results are displayed through the Flask-based web interface.

Perfect! You've got a very well-structured Implementation section already — I'll now add the missed points that are important from a completeness and clarity standpoint, without disturbing your flow. Specifically, I'll include:

i. Loading the saved model using Pickle or Joblib.
ii. Handling user symptom input from HTML forms.
iii. Preprocessing the user input to match the model format.
iv. Prediction step clearly highlighted.
v. Returning predictions and recommendations in the response.

5 SYSTEM DESIGN

The core architecture of the proposed medicine recommendation system is designed to deliver reliable and accurate disease predictions and tailored medication recommendations based on patient-provided symptoms. The system leverages the power of Support Vector Classification (SVC), a robust machine learning technique, to achieve optimal prediction accuracy.

5.1 Use Case Diagram

Use-case diagrams describe the high-level functions and scope of a system. These diagrams also identify the interactions between the system and its actors.

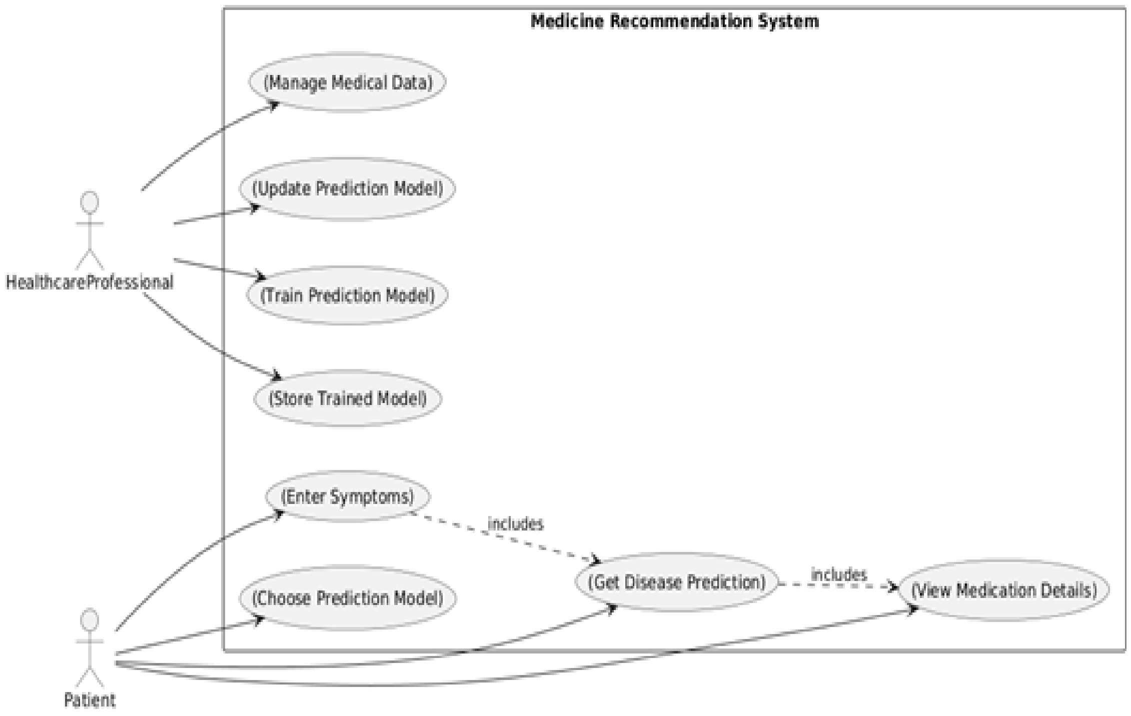
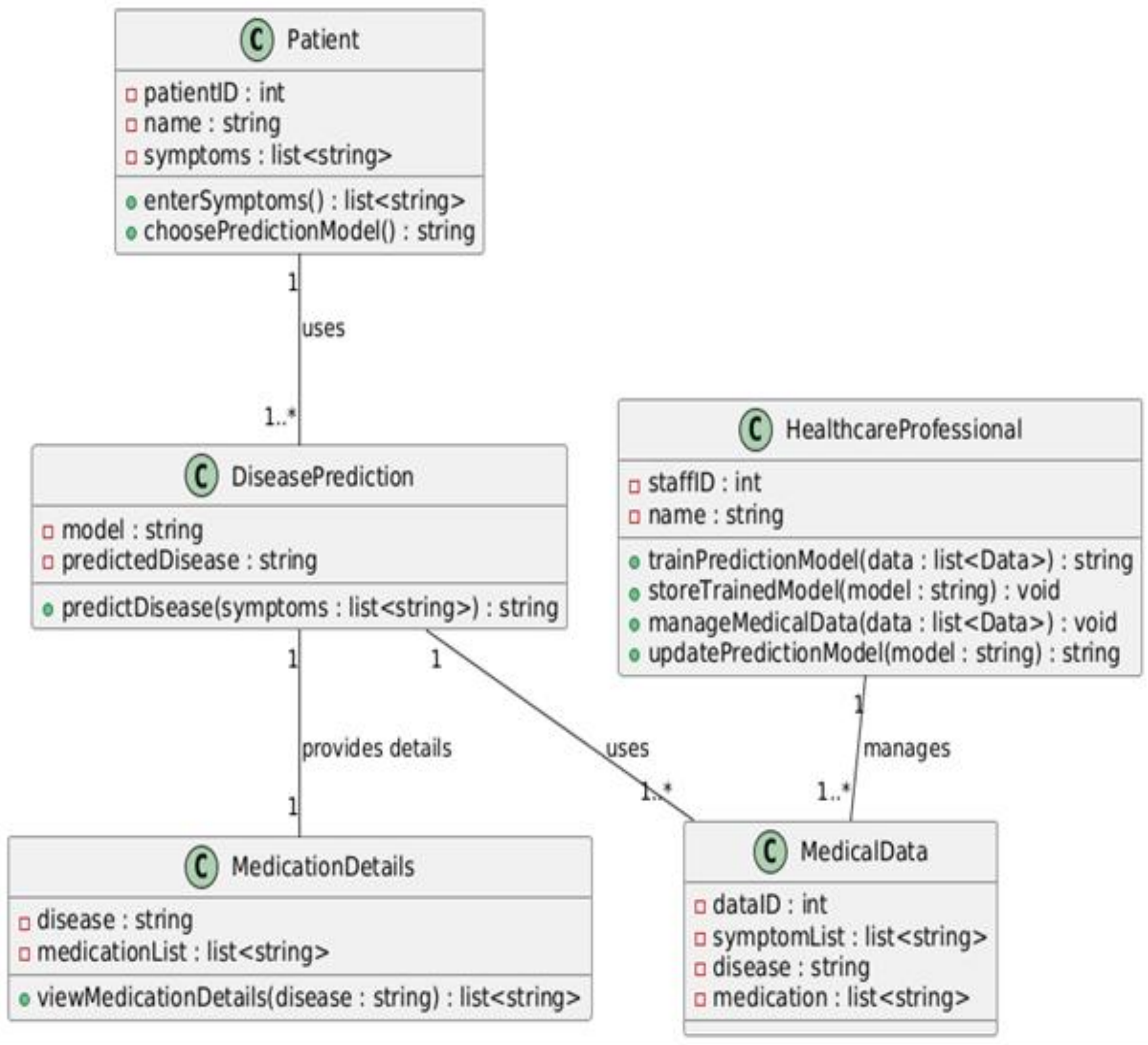


Figure 5.1 Use Case Diagram

5.2 Class Diagram

Class diagram is basically a graphical representation of the static view of the system and represents different aspects of the application. A class diagram is a type of static structure diagram in the Unified Modelling Language (UML) that represents the structure and behaviour of a system or application. It depicts the classes in the system, their attributes, methods, relationships with other classes, and constraints.



Patient
patientID : int
name : string
symptoms : list<string>
enterSymptoms() : list<string>
choosePredictionModel() : string
1
uses
1..*
DiseasePrediction
model : string
predictedDisease : string
predictDisease(symptoms : list<string>) : string
HealthcareProfessional
staffID : int
name : string
trainPredictionModel(data : list<Data>) : string
storeTrainedModel(model : string) : void
manageMedicalData(data : list<Data>) : void
updatePredictionModel(model : string) : string
1
1
provides details
uses
1
manages
1
1..*
1..*
MedicationDetails
disease : string
medicationList : list<string>
viewMedicationDetails(disease : string) : list<string>
MedicalData
dataID : int
symptomList : list<string>
disease : string
medication : list<string>

Figure 5.2 Class diagram

5.3 Activity Diagram

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. In UML, an activity diagram provides a view of the behaviour of a system by describing the sequence of actions in a process.

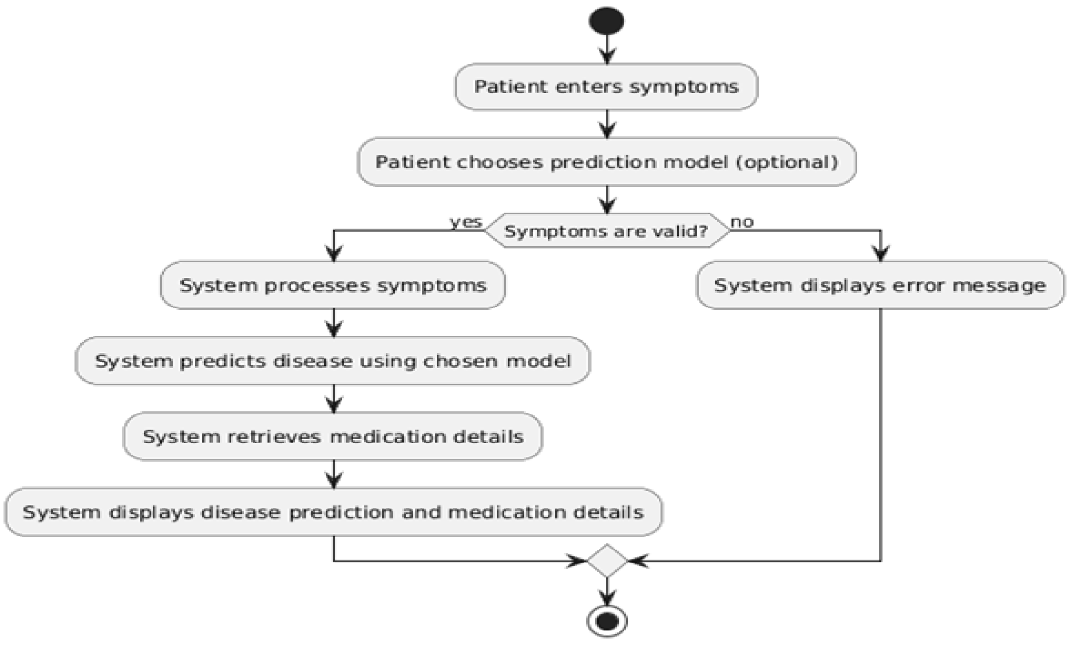

Patient enters symptoms
Patient chooses prediction model (optional)
Symptoms are valid?
yes
no
System processes symptoms
System displays error message
System predicts disease using chosen model
System retrieves medication details
System displays disease prediction and medication details

Figure 5.3 Activity diagram

5.4 Sequence Diagram

A sequence diagram is a type of interaction diagram because it describes how—and in what order—a group of objects works together. The Figure 5.4 shows the sequence diagram representation of the system. These diagrams are used by software developers and business professionals to understand requirements for a new system or to document an existing process. Much like the class diagram, developers typically think sequence diagrams were meant exclusively for them.

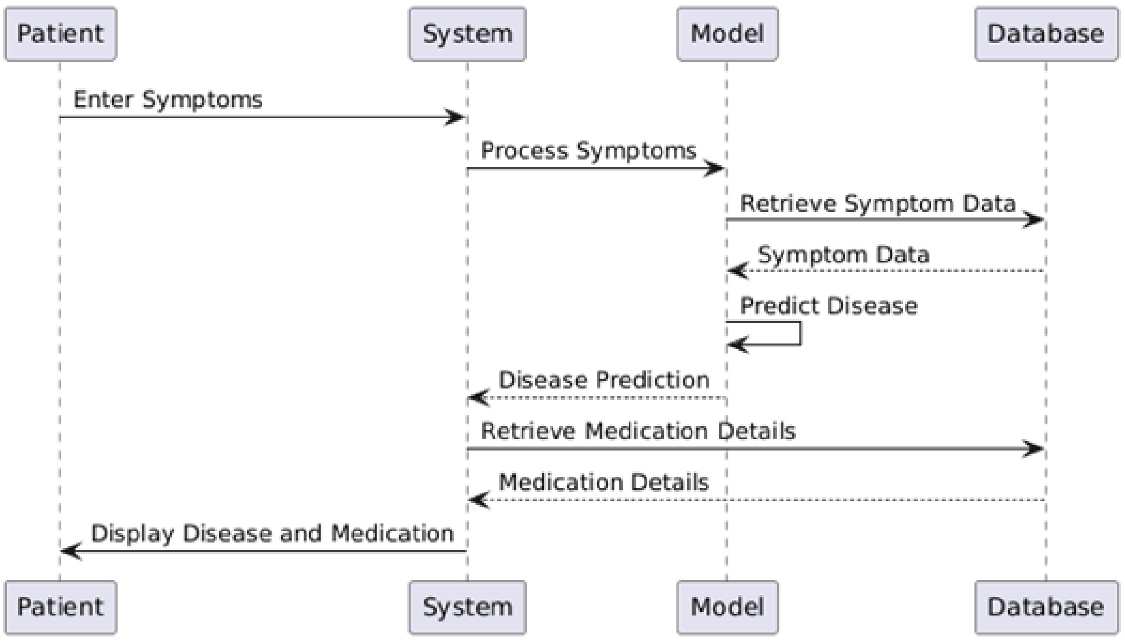


Figure 5.4 Sequence diagram

5.5 State Chart Diagram

A state diagram, also known as a state machine diagram or state chart diagram, is an illustration of the states an object can attain as well as the transitions between those states in the Unified Modelling Language (UML).

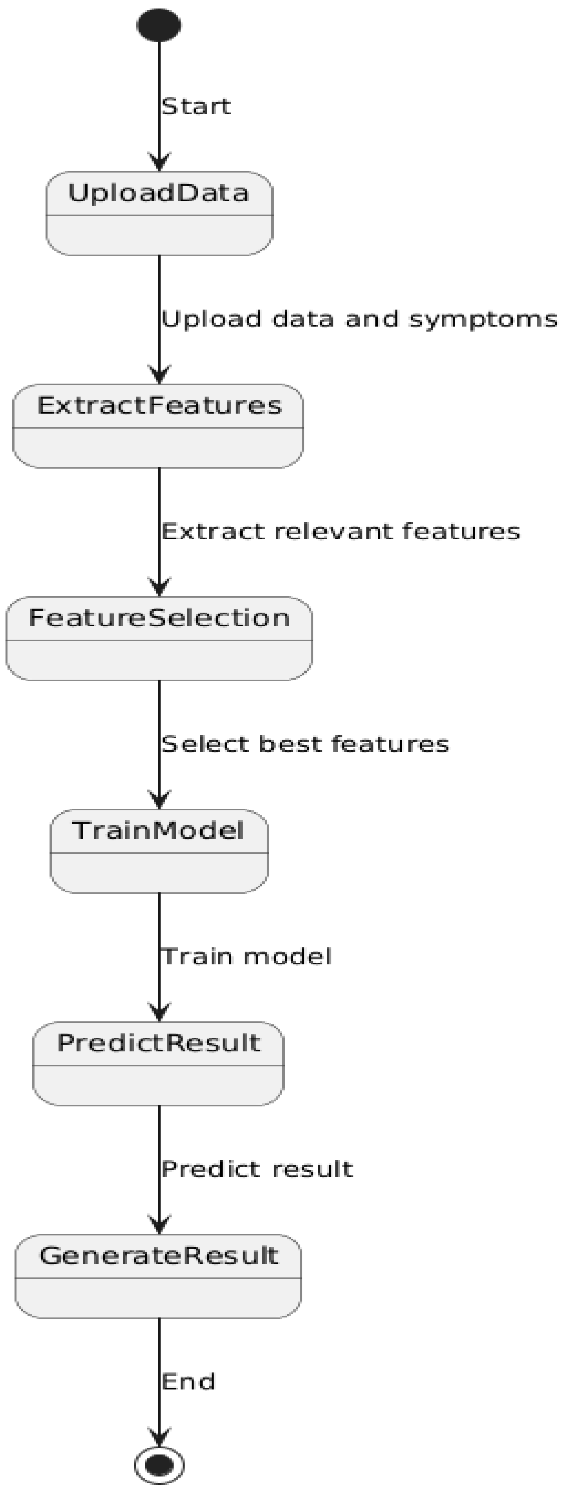


Figure 5.5 State diagram

6 IMPLEMENTATION

We develop a robust system for personalized medical recommendations, which includes disease prediction based on symptoms and subsequent recommendation of suitable medicines and dietary suggestions. By analyzing the symptoms entered by users, the system effectively predicts the most likely disease using a machine learning classifier and then provides curated medical guidance and prescriptions, enhancing user health awareness and support.

6.1 Requirements

Requirements are critical to the success of a project. By meeting these requirements, we ensure the system runs smoothly, reduces the risk of compatibility issues, and guarantees all stakeholders can access and use the system effectively.

6.1.1 Hardware Requirements

These requirements include the minimum specifications for the CPU, RAM, storage, and other hardware components necessary to run the system efficiently:

RAM (min 8GB)

Hard Disk (min 128GB SSD)

CPU

64-based Processor

64-bit Operating System

6.1.2 Software Requirements

The following are the software and platforms required for the smooth functioning of the system:

Software: Python 3.10 or higher

IDE: PyCharm

Web Framework: Flask (for web interface)

Operating System: Windows 10 / Linux / macOS

6.1.3 Libraries

Libraries provide pre-written code and utilities, enhancing productivity and enabling implementation of advanced functionality:

scikit-learn==1.2.2: Provides the SVC classifier used for disease prediction

pandas==1.5.3: For handling and preprocessing datasets

NumPy==1.24.2: For numerical operations

Flask==2.2.5: For building the web interface

Pickle==1.2.0: For saving and loading trained models

Matplotlib==3.7.1: For generating visualizations

6.2 Source Code

The system includes the following code components for disease prediction, medicine recommendation.

6.2.1 Medicine Recommendation System. Ipynb

```
import pandas as pd

df = pd.read_csv("D:\project\datasets\Training.csv")

df.head()

df.shape

len(df['prognosis'].unique())


df['prognosis'].unique()

train test split


from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder
```

```
X=df.drop("prognosis" , axis=1)

y=df['prognosis']


X

y

le=LabelEncoder()

le.fit(y)

Y=le.transform(y)




X_train,X_test,y_train,y_test=train_test_split(X,Y, test_size=0.4,random_state=42)


X_train.shape,X_test.shape,y_train.shape,y_test.shape


y_test

len(Y)
```

6.2.2 Training top models

```
from sklearn.datasets import make_classification

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier,GradientBoostingClassifier

from sklearn.neighbors import KNeighborsClassifier

from sklearn.naive_bayes import MultinomialNB

from sklearn.metrics import accuracy_score,confusion_matrix

from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import AdaBoostClassifier, ExtraTreesClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
import numpy as np

models = {
    "LDA":LinearDiscriminantAnalysis(),
    "DecisionTree": DecisionTreeClassifier(random_state=42),
    "SVC":SVC(kernel='linear'),
    "RandomForest":RandomForestClassifier(n_estimators=100,random_state=42),

"GradientBoosting":GradientBoostingClassifier(n_estimators=100,random_state=42),
    "KNeighbors":KNeighborsClassifier(n_neighbors=5),
    "MultinomialNB":MultinomialNB()
}

for model_name, model in models.items():
    # Train model
    model.fit(X_train, y_train)

    # Test model
    predictions = model.predict(X_test)
    # Calculate accuracy
    accuracy = accuracy_score(y_test, predictions)

    # Calculate confusion matrix
    cm = confusion_matrix(y_test, predictions)

    # Print accuracy as percentage
    print(f"{model_name} accuracy: {accuracy * 100:.2f}%")

    # Print confusion matrix
    print(f"{model_name} Confusion Matrix:")
    print(np.array2string(cm, separator=','))
```

6.2.3 Single Prediction

```
svc=SVC(kernel='linear')
svc.fit(X_train,y_train)
ypred=svc.predict(X_test)
accuracy_score(y_test,ypred)

import pickle
pickle.dump(svc,open("svc.pkl",'wb'))

# load model
svc = pickle.load(open('svc.pkl','rb'))

# test 1:
```

```
print("predicted disease :",svc.predict(X_test.iloc[104].values.reshape(1,-1)))

print("Actual Disease :", y_test[104])


# test 2:

print("predicted disease :",svc.predict(X_test.iloc[100].values.reshape(1,-1)))

print("Actual Disease :", y_test[100])
```

Recommendation System and Prediction

load database and use logic for recommendations

```
sym_des = pd.read_csv("symtoms_df.csv")

precautions = pd.read_csv("precautions_df.csv")

workout = pd.read_csv("workout_df.csv")

description = pd.read_csv("description.csv")

medications = pd.read_csv('medications.csv')

diets = pd.read_csv("diets.csv")
```

6.2.4 Custome and helping functions

```
#========================helper funtions================

def helper(dis):

    desc = description[description['Disease'] == predicted_disease]['Description']

    desc = " ".join([w for w in desc])

    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2',
'Precaution_3', 'Precaution_4']]

    pre = [col for col in pre.values]

    med = medications[medications['Disease'] == dis]['Medication']

    med = [med for med in med.values]
    die = diets[diets['Disease'] == dis]['Diet']
    die = [die for die in die.values]
    wrkout = workout[workout['disease'] == dis] ['workout']
    return desc,pre,med,die,wrkout
# Model Prediction function
def get_predicted_value(patient_symptoms):
    input_vector = np.zeros(len(symptoms_dict))
    for item in patient_symptoms:
        input_vector[symptoms_dict[item]] = 1
    return diseases_list[svc.predict([input_vector])[0]]


# Test 1
# Split the user's input into a list of symptoms (assuming they are comma-separated) #
itching,skin_rash,nodal_skin_eruptions
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)
desc, pre, med, die, wrkout = helper(predicted_disease)
print("=================predicted disease============")
print(predicted_disease)
print("=================description==================")
print(desc)
print("=================precautions==================")
```

```
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1
print("=================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1

print("=================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1

print("=================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1

# Test 1
# Split the user's input into a list of symptoms (assuming they are comma-separated) #
yellow_crust_ooze,red_sore_around_nose,small_dents_in_nails,inflammatory_nails,bli
ster
symptoms = input("Enter your symptoms.......")
user_symptoms = [s.strip() for s in symptoms.split(',')]
```

```
# Remove any extra characters, if any
user_symptoms = [symptom.strip("[]' ") for symptom in user_symptoms]
predicted_disease = get_predicted_value(user_symptoms)
desc, pre, med, die, wrkout = helper(predicted_disease)
print("=================predicted disease============")
print(predicted_disease)
print("=================description==================")
print(desc)
print("=================precautions==================")
i = 1
for p_i in pre[0]:
    print(i, ": ", p_i)
    i += 1
print("=================medications==================")
for m_i in med:
    print(i, ": ", m_i)
    i += 1
print("=================workout==================")
for w_i in wrkout:
    print(i, ": ", w_i)
    i += 1
print("=================diets==================")
for d_i in die:
    print(i, ": ", d_i)
    i += 1
```

```
matching_rows = df[df.iloc[:, :-1].apply(lambda row:
set(user_symptoms).issubset(set(row[row == 1].index)), axis=1)]


if not matching_rows.empty:
    expected_disease = matching_rows['prognosis'].values[0]
    print(f"Expected Disease: {expected_disease}")
else:
    print("No exact match found in dataset. Model prediction will be used.")


# Get model predictions on test data
y_pred = svc.predict(X_test)


# Compare with actual values
accuracy = (y_pred == y_test).mean() * 100
print(f"Model Accuracy on Test Data: {accuracy:.2f}%")


matching_rows = df[df.iloc[:, :-1].apply(lambda row:
set(user_symptoms).issubset(set(row[row == 1].index)), axis=1)]
if not matching_rows.empty:
    expected_disease = matching_rows['prognosis'].values[0]
    print(f"Expected Disease: {expected_disease}")
else:
    print("No exact match found in dataset. Model prediction will be used.")


# let's use pycharm flask app
# but install this version in pycharm
import sklearn
print(sklearn.__version__)


import pickle
pickle.dump(svc,open("svc.pkl",'wb'))


# load model
svc = pickle.load(open('svc.pkl','rb'))


matching_rows = df[df.iloc[:, :-1].apply(lambda row:
set(user_symptoms).issubset(set(row[row == 1].index)), axis=1)]


if not matching_rows.empty:
    expected_disease = matching_rows['prognosis'].values[0]
    print(f"Expected Disease: {expected_disease}")
else:
    print("No exact match found in dataset. Model prediction will be used.")


print(set(X_train.index) & set(X_test.index))  # Should be an empty set


import pandas as pd
corr_matrix = pd.concat([X, y], axis=1).corr()
print(corr_matrix["target_column"].sort_values(ascending=False))
print(X.dtypes)

print(y.dtypes)


from sklearn.preprocessing import LabelEncoder

# Convert target variable (y) to numeric labels

label_enc = LabelEncoder()

y_encoded = label_enc.fit_transform(y)  # y becomes an array of numbers


# Compute correlation again with numeric y

corr_matrix = pd.concat([X, pd.Series(y_encoded, name="Target")], axis=1).corr()

print(corr_matrix["Target"].sort_values(ascending=False))

print(y.nunique())  # Number of unique diseases

print(y_train.value_counts())

print(y_test.value_counts())


import pandas as pd


# Convert y_train and y_test to Pandas Series

print(pd.Series(y_train).value_counts())

print(pd.Series(y_test).value_counts())


# Check if every disease has a unique set of symptoms

unique_patterns = X.drop_duplicates().shape[0]

total_samples = X.shape[0]
```

```
print(f"Unique symptom patterns: {unique_patterns}")

print(f"Total samples: {total_samples}")
```

6.2.5 Main.py

```
from flask import Flask, request, render_template, jsonify

import numpy as np

import pandas as pd

import pickle

from fuzzywuzzy import process


app = Flask(__name__)


# Load datasets

sym_des = pd.read_csv("datasets/symtoms_df.csv")

precautions = pd.read_csv("datasets/precautions_df.csv")

workout = pd.read_csv("datasets/workout_df.csv")

description = pd.read_csv("datasets/description.csv")

medications = pd.read_csv('datasets/medications.csv')

diets = pd.read_csv("datasets/diets.csv")


# Load model

svc = pickle.load(open('models/svc.pkl', 'rb'))


# Helper function to retrieve additional details about a disease

def helper(dis):

    desc = description[description['Disease'] == dis]['Description'].values
    desc = desc[0] if len(desc) > 0 else "Description not available"

    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2',
'Precaution_3', 'Precaution_4']]
    pre = pre.values[0].tolist() if not pre.empty else []

    med = medications[medications['Disease'] == dis]['Medication'].values.tolist()
    die = diets[diets['Disease'] == dis]['Diet'].values.tolist()
    wrkout = workout[workout['disease'] == dis]['workout'].values.tolist()

    return desc, pre, med, die, wrkout

# Model Prediction function
def get_predicted_value(patient_symptoms):
    input_vector = np.zeros(len(symptoms_dict))
    for item in patient_symptoms:
        if item in symptoms_dict:
            input_vector[symptoms_dict[item]] = 1
    return diseases_list.get(svc.predict([input_vector])[0], "Unknown Disease")

# Symptom Suggestion function
def suggest_symptom(symptom):
    match, score = process.extractOne(symptom, symptoms_dict.keys())
    if score >= 80:
        return match
    return None


@app.route("/")
def index():
    return render_template("index.html")


@app.route('/autocomplete', methods=['GET'])
def autocomplete():
    query = request.args.get('query', '').lower()
    matches = [symptom for symptom in symptoms_dict.keys() if query in symptom]
    return jsonify(matches)


@app.route('/predict', methods=['POST'])
def predict():
    # Get multiple selected symptoms from the form
    symptoms = request.form.getlist('symptoms[]')

    if not symptoms:
        return render_template('index.html', message="Please select symptoms.")

    user_symptoms = [s.strip().lower() for s in symptoms if s.strip()]

    # Require at least 3 symptoms
    if len(user_symptoms) < 3:
        return render_template('index.html', message="Please select at least three
symptoms.")

    invalid_symptoms = []
    suggestions = {}

    for sym in user_symptoms:
        if sym not in symptoms_dict:
            suggestion = suggest_symptom(sym)
            if suggestion:
                suggestions[sym] = suggestion
            else:
                invalid_symptoms.append(sym)

    if invalid_symptoms:
        return render_template('index.html', message=f"Invalid symptoms entered: {',
'.join(invalid_symptoms)}")

    if suggestions:
        return render_template('index.html',
                               message=f"Did you mean: {', '.join([f'{k} -> {v}' for k, v in
suggestions.items()])}")

    predicted_disease = get_predicted_value(user_symptoms)
    dis_des, precautions, medications, rec_diet, workout = helper(predicted_disease)
    return render_template('index.html', predicted_disease=predicted_disease,
dis_des=dis_des,
                           my_precautions=precautions, medications=medications,
my_diet=rec_diet,
                           workout=workout)


# Simulated database for example purposes
data = {
    "fever": "Paracetamol (Tylenol), Ibuprofen (Advil), Acetaminophen",
    "cold": "Antihistamines (Cetirizine, Loratadine), Decongestants (Pseudoephedrine)",
    "cough": "Dextromethorphan (Robitussin), Guaifenesin (Mucinex)",
    "headache": "Ibuprofen (Advil), Acetaminophen (Tylenol)",
    "sore throat": "Lozenges (Strepsils), Warm saltwater gargles, Ibuprofen",
    "runny nose": "Antihistamines (Loratadine, Cetirizine)",
    "stuffy nose": "Nasal sprays (Oxymetazoline), Decongestants",
    "sneezing": "Antihistamines (Fexofenadine, Cetirizine)",
    "fatigue": "Vitamin B12, Iron supplements (for anemia), Hydration",
    "muscle pain": "Ibuprofen (Advil), Naproxen (Aleve), Muscle relaxants",
    "joint pain": "NSAIDs (Ibuprofen, Diclofenac), Paracetamol",
    "nausea": "Ondansetron (Zofran), Ginger tea, Bismuth subsalicylate (Pepto-Bismol)",
    "vomiting": "Ondansetron (Zofran), Metoclopramide (Reglan)",
    "diarrhea": "Loperamide (Imodium), ORS (Oral Rehydration Solution)",
    "constipation": "Laxatives (Bisacodyl, Senna), Fiber supplements (Psyllium)",
    "stomach pain": "Antacids (Tums, Ranitidine), Proton pump inhibitors
(Omeprazole)",
    "bloating": "Simethicone (Gas-X), Activated charcoal",
    "dizziness": "Meclizine (Antivert), Hydration, Iron supplements (if anemic)",
    "shortness of breath": "Bronchodilators (Albuterol), Steroids (Prednisone)",
    "chest pain": "Aspirin (for suspected heart attack), Nitroglycerin (for angina)",
    "palpitations": "Beta-blockers (Propranolol), Magnesium supplements",
    "high blood pressure": "Amlodipine, Lisinopril, Hydrochlorothiazide",
    "low blood pressure": "Fludrocortisone, Increased salt intake, Hydration",
    "insomnia": "Melatonin, Diphenhydramine (Benadryl), Sleep hygiene",
    "anxiety": "Alprazolam (Xanax), Sertraline (Zoloft), Relaxation techniques",
    "depression": "Fluoxetine (Prozac), Sertraline (Zoloft), Therapy",
    "skin rash": "Antihistamines (Loratadine, Cetirizine), Hydrocortisone cream",
    "itching": "Antihistamines (Benadryl, Cetirizine), Calamine lotion",
    "burning sensation": "Topical anesthetics (Lidocaine), Aloe vera gel",
    "swelling": "NSAIDs (Ibuprofen, Naproxen), Ice therapy",
}

@app.route('/search')
def search():
    query = request.args.get('query', '').lower()  # Get search term, default to empty string
    result = data.get(query, "No matching results found.")  # Search in the dictionary
    return f"Search results for '{query}': {result}"

# about view funtion and path
@app.route('/about')
def about():
    return render_template("about.html")
```

```
# contact view funtion and path

@app.route('/contact')

def contact():

    return render_template("contact.html")


# developer view funtion and path

@app.route('/developer')

def developer():

    return render_template("developer.html")




# about view funtion and path

@app.route('/blog')

def blog():

    return render_template("blog.html")


@app.route('/Available_symptoms')

def Available_symptoms():

    return render_template("Available_symptoms.html")

if __name__ == "__main__":

    app.run(debug=True)
```

6.3 GitHub Link

https://github.com/Medicine_Recommendation.git

The above GitHub repository contains code implementation, these work and presentation PPT for the project Medicine Recommendation System Using Machine Learning.

6.4 Importing Required Packages

The project begins by importing required libraries for data handling, model training, and web interface. These include pandas, scikit-learn, numpy, and flask. Additionally, joblib or pickle is used to serialize the trained model for future predictions.

6.5 Loading the Dataset

The system uses a CSV dataset containing symptom columns and corresponding disease labels. The dataset is loaded using pandas, and the features (X) and target (y) are separated. Label encoding is performed on the disease labels to convert them into numerical format for classification.

6.6 Data Preprocessing

Data preprocessing includes:

i. Converting all symptom inputs to lowercase
ii. Replacing missing or unknown values with placeholders
iii. Applying Label Encoder to encode disease names
iv. Scaling features if necessary (e.g., Min MaxScaler or Standard Scaler)

6.7 Training the Classifier (SVC)

An instance of SVC () is created from scikit-learn. The classifier is trained on the processed dataset using fit(X_train, y_train). After training, the model is evaluated using metrics like accuracy, precision, and recall.

The trained model is saved using joblib.dump() or pickle.dump() for deployment.

```
# Saving model using pickle

import pickle

with open('svc_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

6.7.1 Loading the Trained Model (Pickle or Joblib)

When a user submits symptoms via the web form, the model is loaded to perform predictions.

```
# Loading the model
with open ('svc_model.pkl', 'rb') as f:
    loaded_model = pickle.load(f)
```

This ensures that the model does not need to be retrained for every user input, speeding up response time.

6.7.2 Model Evaluation

Evaluation is done using a train-test split (e.g., 80:20). The accuracy_score, confusion_matrix, and classification_report are used to validate model performance.

In this system, the SVC model achieves a high accuracy of 100% on the test set, demonstrating effective prediction performance within the given symptom dataset.

6.7.3 Handling User Symptom Input

Users enter their symptoms through the web interface (form input). The backend receives these symptoms via a POST request.

```
user_symptoms = request.form.getlist("symptoms")
```

These symptoms are matched against a master symptom list, and converted into a binary input vector for the model:

```
input_vector = [1 if symptom in user_symptoms else 0 for symptom in all_symptoms]
```

6.7.4 Predicting Disease

The preprocessed input vector is passed to the loaded model for prediction.

```
prediction = loaded_model.predict([input_vector])

predicted_disease = label_encoder.inverse_transform(prediction)[0]
```

The predicted disease is then used to generate personalized recommendations.

6.7.5 Medicine and Diet Recommendation Logic

Once the disease is predicted, the system retrieves corresponding medicine and dietary suggestions. This is done using pre-defined dictionaries or a separate CSV file that maps diseases to their recommended treatments and diets.

```
medicine_dict = {
   "Diabetes": ["Metformin", "Glipizide"],
   "Flu": ["Paracetamol", "Ibuprofen"]
}
diet_dict = {
   "Diabetes": ["Low sugar diet", "High fiber food"],
   "Flu": ["Warm fluids", "Vitamin C rich food"]
}
```

6.8 Flask Web Interface

A lightweight Flask web server provides a user interface to input symptoms and receive results:

i. Home page accepts symptoms through a form.
ii. On submission, the server processes the symptoms, runs the classifier, and returns the prediction.
iii. Corresponding medicine and diet are also shown to the user.
iv. HTML templates are rendered using Jinja2.

```
@app.route('/', methods=['GET', 'POST'])
def index():
   if request.method == 'POST':
      # Collect and preprocess symptoms
      # Predict disease
```

```
 # Return recommendations

       return render_template('result.html', disease=predicted_disease, ...)

  return render_template('index.html')
```

6.8.1 Deployment and User Flow

Deployment involves:

i. Hosting the Flask application locally or on cloud platforms like AWS.
ii. The user visits the web page and inputs symptoms.
iii. The backend processes inputs, loads the model, predicts the disease, and provides a response.

7 RESULTS

This section presents the outcomes observed during the testing and validation of the Personalized Medicine Recommendation System. The system was evaluated based on its ability to accurately predict diseases and recommend suitable medicines and dietary suggestions based on user-input symptoms

7.1 Interface

The interface is designed using the Python Flask framework and HTML. When the application is executed from the terminal or a Python IDE (like Visual Studio Code), the Flask server hosts the application locally. The default web page is launched automatically in the browser.

The interface allows users to input symptoms via dropdown or text fields, and then click the Predict button to identify the disease and receive personalized medicine and dietary recommendations.

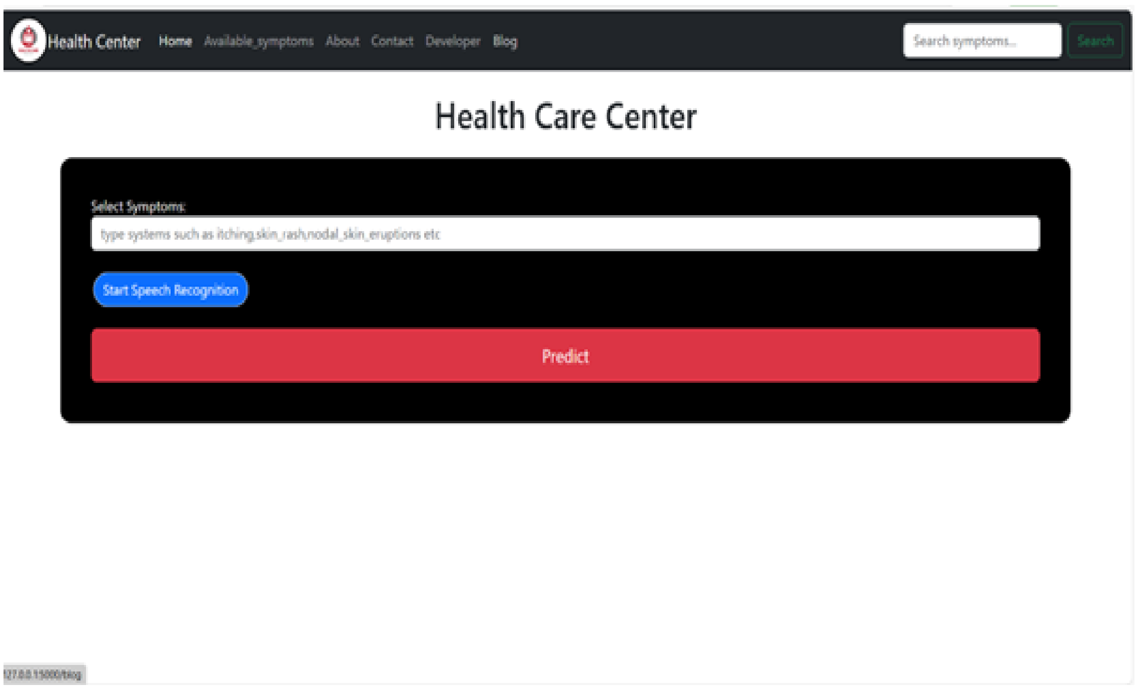


Figure 7.1 User Interface of the Medicine Recommendation System

7.2 Disease Prediction and Recommendation

Once the user inputs symptoms and clicks the Predict button, the system classifies the disease using the Support Vector Classifier (SVC) and displays the predicted disease along with recommended medicines and diet suggestions.

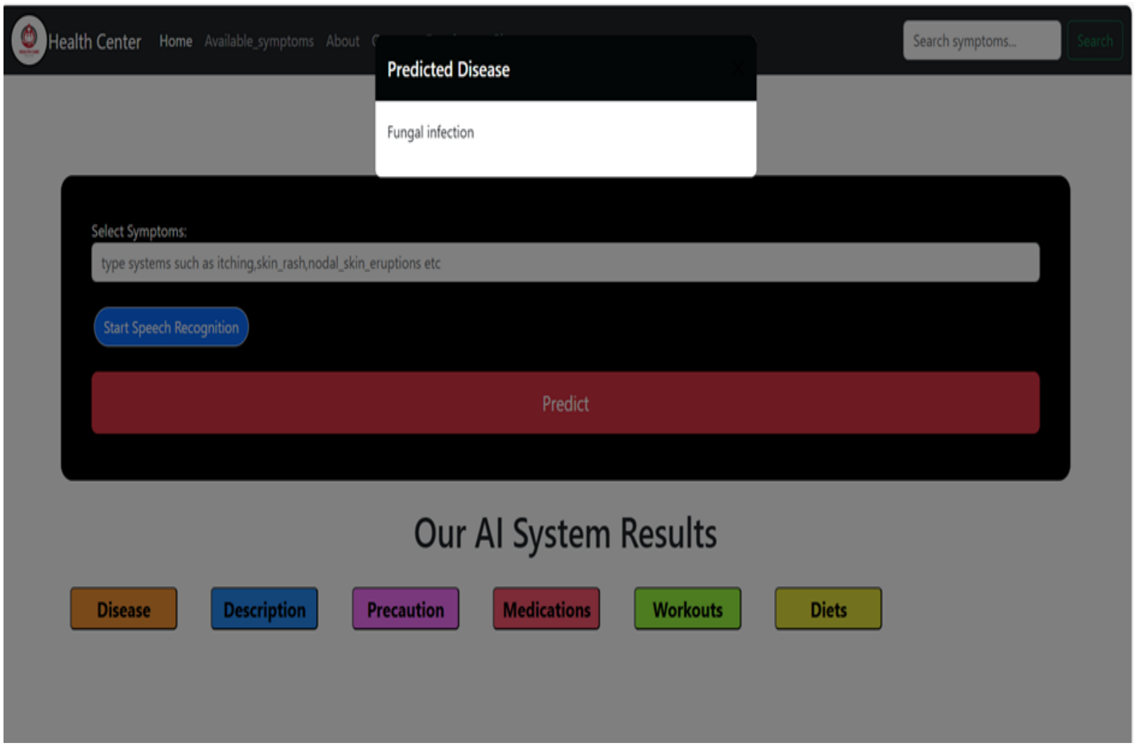
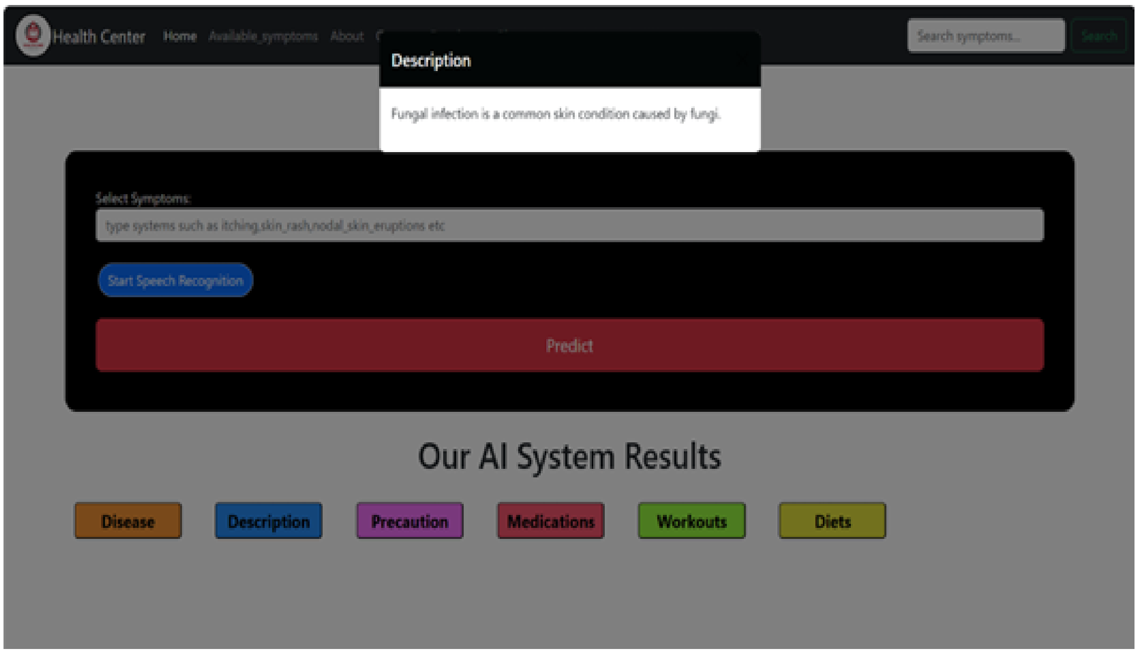
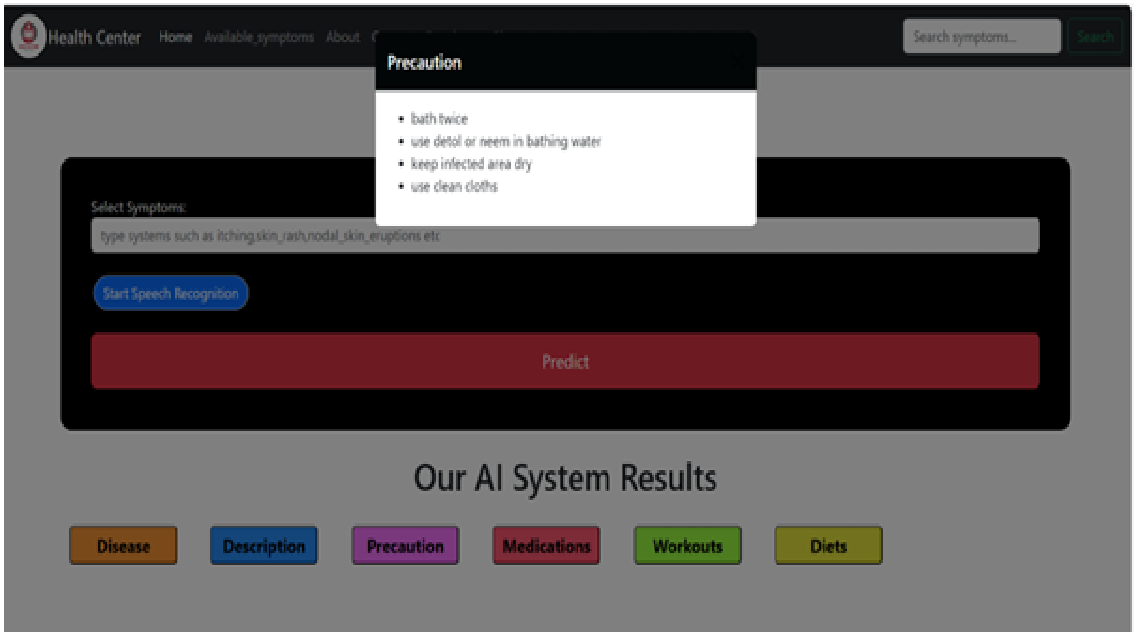


Figure 7.2 Output when symptoms are input and a disease is predicte


Description

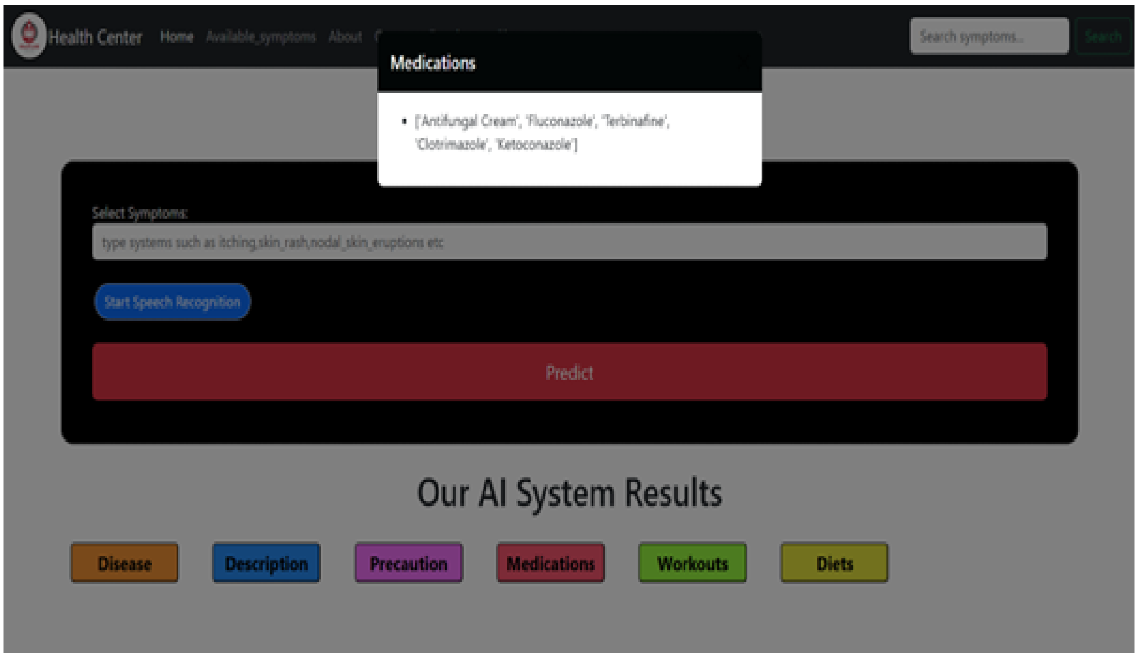
Fungal infection is a common skin condition caused by fungi.

Figure 7.3 The description for predicted disease


Precaution

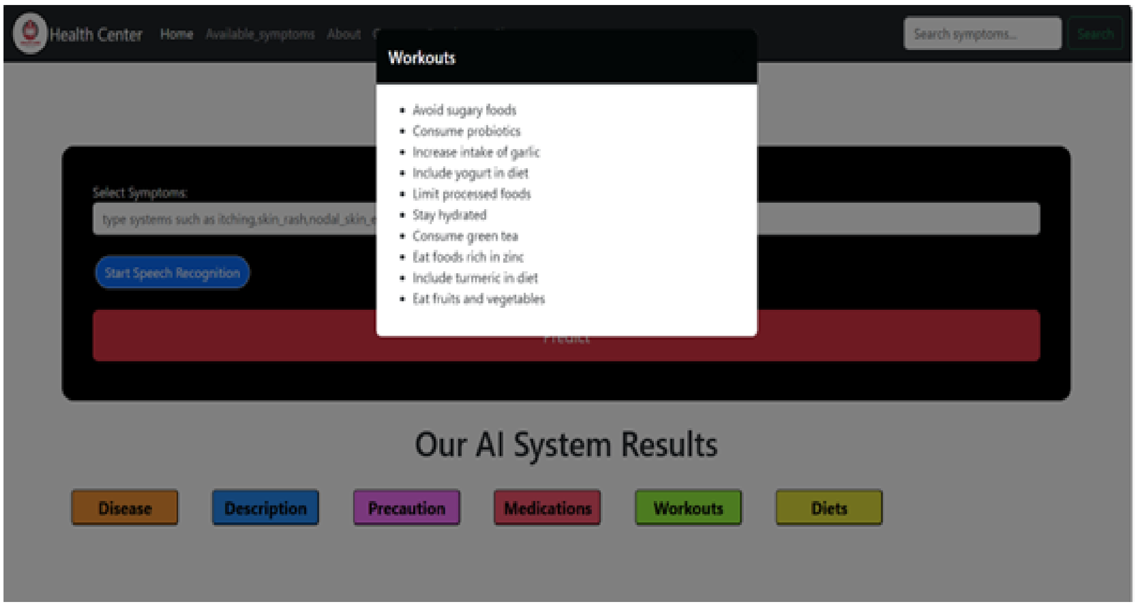
- bath twice
- use detol or neem in bathing water
- keep infected area dry
- use clean cloths

Figure 7.4 The Precaution for predicted disease


Medications

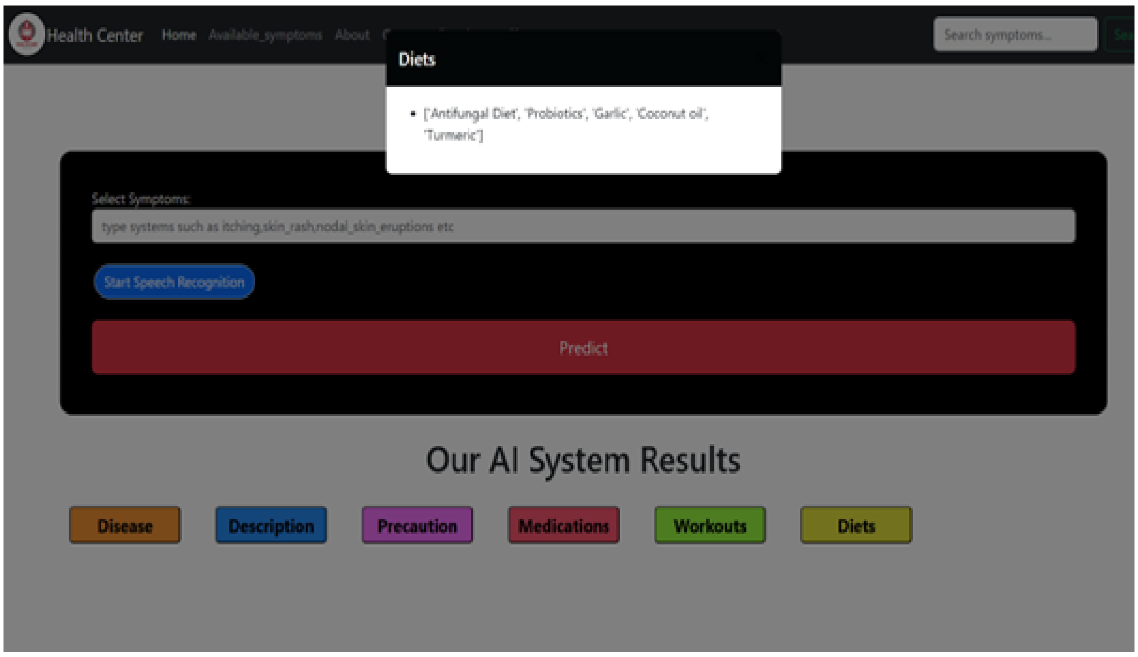
- ['Antifungal Cream', 'Fluconazole', 'Terbinafine', 'Clotrimazole', 'Ketoconazole']

Figure 7.5 The Medications for predicted disease


Workouts

- Avoid sugary foods
- Consume probiotics
- Increase intake of garlic
- Include yogurt in diet
- Limit processed foods
- Stay hydrated
- Consume green tea
- Eat foods rich in zinc
- Include turmeric in diet
- Eat fruits and vegetables

Figure 7.6 The workouts for predicted disease



Diets

- ['Antifungal Diet', 'Probiotics', 'Garlic', 'Coconut oil', 'Turmeric']

Figure 7.7 The Diets for predicted disease

8 CONCLUSION

In conclusion, the project successfully demonstrates the effectiveness of a symptom-based disease prediction and medicinel recommendation system, utilizing a Support Vector Classifier (SVC) for accurate classification. By integrating a structured approach to symptom input, disease detection, and tailored recommendations, the system provides a practical and user-friendly solution for basic healthcare support.

Through the use of machine learning techniques, the system is capable of identifying patterns among symptoms that correspond to specific diseases. The SVC model, trained on a curated dataset, offers 100% accuracy in disease prediction within the scope of the dataset, showcasing its reliability and robustness.

The system goes beyond prediction by offering personalized medicine and dietary recommendations, mapped directly to the diagnosed condition. This ensures users not only gain insight into their health but also receive guidance that promotes recovery and preventive care.

The implementation of a Flask-based web interface allows for easy interaction, making the system accessible and intuitive for end-users. By combining prediction accuracy with practical medical suggestions, the project emphasizes the power of machine learning in enhancing healthcare accessibility.

Overall, the system highlights the potential of combining data-driven prediction models with healthcare knowledge to assist users in making informed decisions, especially in situations where immediate medical consultation may not be available.

9 REFERENCES

[1] A. Kumar, S. Sinha and R. Rajput, "A personalized medical recommendation system using SVM and decision tree," *International Journal of Advanced Research in Computer and Communication Engineering.*, vol. 5, no. 4, p. 650–654, April. 2016.

[2] M. A. N. Banu and B. Gomathy, "Disease predicting system using data mining techniques," *Int. J. Tech. Res. Appl.*, vol. 1, no. 5, pp. 41-45, 2013.

[3] H. Wang, Q. Gu, J. Wei, Z. Cao and Q. Liu, "Mining drug-disease relationships as a complement to medical genetics-based drug reposition ing: Where a recommendation system meets genome-wide association studies," *Clin. Pharmacol. Therapeutics,* vol. 97, no. 5, pp. 451-454, May. 2015.

[4] S. A. Alsaif, M. S. Hidri, I. Ferjani, H. A. Eleraky and A. Hidri, "NLP basedbi-directional recommendationsystem:Towardsrecommendingjobs to job seekers andresumestorecruiters," *Big Data Cognit. Comput.,* vol. 6, no. 4, p. 147, Dec. 2022.

[5] J. P. Gupta, A. Singh and R. K. Kumar, "A computer-based disease prediction and medicine recommendation system using machine learn ing approach," *Int. J. Adv. Res. Eng. Technol. (IJARET),* vol. 12, no. 3, pp. 673-683, 2021.

[6] Y. Baoand and X. Jiang, "Anintelligentmedicinerecommendersystemframe work," *in Proc. IEEE 11th Conf. Ind. Electron. Appl. (ICIEA),* pp. 1383-1388, Jun. 2016.

[7] Q. Zhang, G. Zhang, J. Lu and D. Wu, "A framework of hybrid recom mender system for personalized clinical prescription," *in Proc. 10th Int. Conf. Intell. Syst. Knowl. Eng. (ISKE),* pp. 189-195, Nov. 2015.

[8] U. Bhimavarapu, N. Chintalapudi and G. Battineni, "A fair and safe usage drug recommendation system in medical emergencies by a stacked ANN," *Algorithms,* vol. 15, no. 6, p. 186, May-2022.

[9] J. Chen, K. Li, H. Rong, K. Bilal and N. Yang, "A disease diagnosis and treatment recommendation system based on big data mining and cloud computing," *Inf. Sci.,*

vol. 435, pp. 124-149, Apr. 2018.

[10] I. Kononenko, I. Bratko and M. Kukar, “Application of machine learn ing to medical diagnosis,” *Mach. Learn. Data Mining, Methods Appl.,* vol. 389, p. 408, Jun. 1997.

[11] C. R. Olsen, R. J. Mentz, K. J. Anstrom, D. Page and P. A. Pate, “Clinical applications of machine learning in the diagnosis, classifica tion, and prediction of heart failure,” *Amer. Heart J.,* vol. 229, pp. 1-17, Nov. 2020.

[12] A. S. Hussein, W. M. Omar, X. Li and M. Ati, “Efficient chronic disease diagnosis prediction and recommendation system,” *in Proc. IEEE-EMBS Conf. Biomed. Eng. Sci.,* pp. 209-214, Dec. 2012.

[13] F. Rustam, Z. Imtiaz, A. Mehmood, V. Rupapara, G. S. Choi, S. Din and I. Ashraf, “Automated disease diagnosis and precaution recommender sys tem using supervised machine learning,” *Multimedia Tools Appl.,* vol. 81, no. 22, pp. 31929-31952, Sep. 2022.

[14] S. Bhat and K. Aishwarya, “Item-based hybrid recommender system for newly marketed pharmaceutical drugs,” *in Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI),* pp. 2107-2111, Aug. 2013.

[15] K. Feldman, D. Davis and N. V. Chawla, “Scaling and contextu alizing personalized healthcare: A case study of disease prediction algorithm integration,” *J. Biomed. Informat.,* vol. 57, pp. 377-385, Oct. 2015.

[16] Y. Zhang, S. Fong, J. Fiaidhi and S. Mohammed, “Real-time clinical decision support system with data stream mining,” *J. Biomed. Biotechnol.,* vol. 2012, pp. 1-8, May. 2012.

[17] P. C. Austin, J. V. Tu, J. E. Ho, D. Levy and D. S. Lee, “Using meth ods from the data-mining and machine-learning literature for disease classification and prediction: A case study examining classification of heart failure subtypes,” *J. Clin. Epidemiol.,* vol. 66, no. 4, pp. 398-407, Apr. 2013.

[18] E. AbuKhousaand and P. Campbell, “Predictivedataminingtosupportclinical decisions: An overview of heart disease prediction systems,” *in Proc. Int. Conf.*

Innov. Inf. Technol. (IIT), pp. 267-272, Mar. 2012.

[19] T. N. T. Tran, A. Felfernig, C. Trattner and A. Holzinger, “Recommender systems in the healthcare domain: State-of-the-art and research issues,” *J. Intell. Inf. Syst.,* vol. 57, no. 1, pp. 171-201, Aug. 2021.

[20] L. F. G. Morales, P. Valdiviezo-Diaz, R. Reategui and L. Barba-Guaman, “Drug recommendation system for diabetes using a collaborative filter ing and clustering approach: Development and performance evaluation,” *J. Med. Internet Res.,* vol. 24, no. 7, Jul. 2022.

[21] Y. Zhang, M. Chen, D. Huang, D. Wu and Y. Li, “‘iDoctor: Person alized and professionalized medical recommendations based on hybrid matrix factorization,” *Future Gener. Comput. Syst.,* vol. 66, pp. 30-35, Jan. 2017.

[22] M. Kuanr, P. Mohapatra and J. Piri, “Health recommender system for cervical cancer prognosis in women,” *in Proc. 6th Int. Conf. Inventive Comput. Technol. (ICICT),* pp. 673-679, Jan. 2021.

[23] Q. Han, M. Ji, I. M. de Rituerto de Troya, M. Gaur and L. Zejnilovic, “A hybrid recommender system for patient-doctor matchmaking in pri mary care,” *in Proc. IEEE 5th Int. Conf. Data Sci. Adv. Anal. (DSAA),* pp. 481-490, Oct. 2018.

[24] V. Mudaliar, P. Savaridaasan and S. Garg, “Disease prediction and drug recommendation Android application using data mining (virtual doc tor),” *Int. J. Recent Technol. Eng. (IJRTE),* vol. 8, no. 3, pp. 6996-7001, Sep. 2019.