VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“Jnana Sangama”, Belagavi -590018, Karnataka

A Mini Project Report On

“MEDICINE RECOMMENDATION SYSTEM”

Submitted in the partial fulfillment for award of the degree of

BACHELOR OF ENGINEERING

IN

ARTIFICIAL INTELLIGENCE AND DATA SCIENCE

Submitted by

KUSUMA S	**4MN21AD018**
PRAKRUTHI K P	**4MN21AD030**
RANJITHA N	**4MN21AD034**
SINDHU K	**4MN21AD039**

Under The Guidance Of

Prof . MADHAN KUMAR G S

Assistant professor , Dept. Of AI & DS

MIT Thandavapura

Maharaja Institute Of Technology Thandavapura

Department of Artificial Intelligence And Data Science

NH 766, Nanjangud taluk, Mysore district-571302

2023-2024

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

NH 766, Nanjangud Taluk, Mysuru- 571 302

(Approved by AICTE & Affiliated to Visveswaraya Technological University, Belagavi)

(Certified by ISO 91001:2015 & ISO 21001:2018)

DEPARTMENT OFARTIFICIAL INTELLIGENCE AND DATA SCIENCE

CERTIFICATE

Certified that the mini project work entitled **"MEDICINE RECOMMENDATION SYSTEM"** is a bonafide work carried out by **KUSUMA S (4MN21AD018), RANJITHA N (4MN21AD034), PRAKRUTHI K P (4MN21AD030), SINDHU K (4MN21AD039)** for the **MINI PROJECT** with course code **21ADMP67** of Sixth Semester in Artificial Intelligence and Data Science under Visvesvaraya Technological University, Belagavi during academic year 2023-24. It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

----------	----------
Signature of the Guide	Signature of the HOD
Mr. Madhan Kumar G S	**Dr. Swarnalatha K**
Assistant professor	Assistant professor &
Dept. Of AI & DS	HOD Dept. Of AI & DS

External viva

Name of the Examiners	**Signature with date**
1)……	……
2)……	....……

ACKNOWLEDGEMENT

It is the time to acknowledgement all those who have extended their guidance, inspiration and their whole hearted co-operation all along our project work.

We would like to extend our thanks to **Dr. Y T Krishne Gowda,** Principal, Maharaja Institute of Technology Thandavapura,for his co-operation throughout the academics. That has helped us in satisfactory completion of project.

We express our sincere thanks to **Dr. H K Chethan,** Joint Secretory of MET Mysuru**.** For his continuous support and appreciation during this program.

We are grateful to **Dr. Swarnalatha K,** Associate Professor and HOD, Department of Artificial intelligence and data Science, for providing us a valuable support throughout the period of project.

We express our deepest sense of gratitude to our guide **Mr. Mohammed Salamath ,** Assistant Professor, Department of Artificial intelligence and Data science, for his valuable guidance, suggestions and cheerful encouragement during the entire period of project.

We also take an opportunity to thank all the teaching and non-teaching staff members of AI&DS Department.

Finally, we thank almighty god and we are grateful to our parents for their faith, love and kindness in believing us. We are indebted to MITT**,** who has directly or indirectly supported us during the period of mini project.

KUSUMA S
(4MN21AD018)
PRAKRUTHI K P
(4MN21AD030)
RANJITHA N
(4MN21AD034)
SINDHU K
(4MN21AD039)

ABSTRACT

In recent years, personalized medicine has been at the forefront of revolutionizing healthcare, tailoring treatments to individual patient profiles for more effective outcomes. This project delves into the development of a medical recommendation system powered by machine learning, highlighting its pivotal role in personalizing medical care. By analysing vast amounts of patient data, machine learning algorithms can provide tailored medical recommendations, enhancing the accuracy and efficiency of t III

This report explores the real-life applications of personalized medical recommendations, demonstrating how artificial intelligence (AI) is reshaping patient-centric healthcare. It underscores the transformative impact of AI and machine learning in delivering customized treatment plans, thereby improving patient outcomes. Furthermore, the report discusses the potential of data-driven insights to significantly enhance patient wellness, offering a glimpse into the future of healthcare where precision medicine becomes the norm.

By understanding these advancements, healthcare professionals can better appreciate the immense value of integrating AI-driven medical recommendation systems into clinical practice, ultimately leading to a more effective and patient-focused healthcare delivery system.

Our machine learning models are trained on extensive healthcare datasets to identify patterns and correlations that inform personalized treatment recommendations. These models are integrated into the Flask framework, enabling real-time analysis and decision-making based on patient data.

By integrating these technologies, our medical recommendation system not only offers cutting-edge personalized medical insights but also ensures a user-friendly and accessible platform for all users. The system provides tailored suggestions for medicines, precautions, workouts, diets and detailed descriptions of the disease predicted based on the symptoms entered by the user. This holistic approach facilitates the adoption of personalized medicine in everyday clinical practice, paving the way for a new era of patient-centric healthcare

CONTENT

Chapter 1

INTRODUCTION

This project focuses on the development of a medical recommendation system powered by machine learning, highlighting its crucial role in personalizing medical care. By analysing vast amounts of patient data, machine learning algorithms provide tailored medical recommendations, enhancing the accuracy and efficiency of treatments.

1.1 Overview

Our Personalized Medical Recommendation System is designed to help users manage their health with ease. By inputting symptoms into our user-friendly interface, users can quickly receive predictions of potential diseases through advanced machine learning models. These models provide accurate and reliable results, ensuring users get the most relevant information.

Beyond predictions, our system offers tailored recommendations for the top 5 medicines, prescription details, and workout routines based on the identified condition. Powered by a Flask web application, users can access these healthcare recommendations conveniently from any location. We also prioritize user privacy and security, handling all health information with strict confidentiality. As our system gathers more data, it continuously improves, enhancing the accuracy and relevance of the recommendations provided.

1.2 Problem Statement

The core problem addressed by this Personalized Medical Recommendation System is the challenge of accurately diagnosing potential health issues based on user-input symptoms. Traditional methods of seeking medical advice often involve lengthy consultations or generalized information, which may not be tailored to individual needs. This system leverages advanced machine learning models to analyse symptoms input by users, providing precise predictions of potential diseases and tailored health recommendations.

Additionally, the system addresses the issue of accessing personalized medical advice efficiently. By integrating with a user-friendly Flask web application, it ensures that users can obtain reliable health recommendations and prescription details quickly and conveniently from anywhere. This approach helps users manage their health proactively and make informed decisions, all while ensuring their data privacy and security.

1.3 Objective

- **Develop a Personalized Medical Recommendation System:** Create a robust system that leverages machine learning algorithms to deliver tailored medical recommendations based on individual patient data.
- **Enhance Treatment Accuracy and Efficiency:** Utilize machine learning to analyse extensive healthcare datasets and identify patterns, thereby providing more precise and effective treatment plans.
- **Integrate AI into Clinical Practice:** Implement the recommendation system within the Flask framework to enable real-time analysis and decision-making, making it accessible for healthcare professionals in clinical settings.
- **Provide Comprehensive Health Insights:** Offer personalized suggestions for medicines, precautions, workouts, diets, and detailed disease descriptions based on symptoms entered by the user.
- **Improve Patient Outcomes and Wellness:** Use data-driven insights to enhance patient wellness and outcomes, promoting the adoption of personalized medicine in everyday healthcare.
- **Facilitate User-Friendly Access:** Ensure the system is user-friendly and accessible, allowing both healthcare professionals and patients to easily interact with and benefit from the recommendations.
- **Pave the Way for Future Healthcare Advancements:** Contribute to the evolution of patient-centric healthcare by showcasing the transformative impact of AI and machine learning in personalized medicine.

1.4 System Development

1.4.1 Collecting Dataset: The first step in developing the proposed system is to collect a comprehensive dataset that includes patient history, genetics, lifestyle, and other health metrics. This dataset will serve as the foundation for training the machine learning models. Ensuring the dataset is diverse and extensive will be crucial for the model's ability to provide accurate and personalized medicine recommendations.

1.4.2 Model Training and Testing: Once the dataset is collected, the next step is to train the machine learning models. This involves feeding the data into the models and using various algorithms to identify patterns and relationships. The models will be trained to predict the most effective treatment plans based on individual patient data. Rigorous testing will follow to validate the model's accuracy and ensure it performs well on unseen data. This step is critical to refine the models and improve their predictive capabilities.

1.4.3 Medicine Recommendation: The final step is deploying the trained and tested models to make real-time medicine recommendations. The system will analyse incoming patient data and provide tailored treatment suggestions based on the insights gained during the training phase. Continuous monitoring and updating of the model will be necessary to maintain its accuracy and relevance, ensuring that the recommendations adapt to new data and evolving medical knowledge.

Chapter-2

REQUIREMENT ANALYSIS

This chapter discusses the limitations of the current rule-based medicine recommendation system and outlines the proposed system's improvements, along with the necessary software and hardware requirements.

2.1 Existing System

In the existing system, a rule-based approach is employed where predefined rules, created by medical experts, are used to recommend medicines based on symptoms or conditions. This process is largely manual, requiring doctors or pharmacists to check patient history and symptoms to suggest appropriate medications. The recommendations are static, limited to a fixed set of medicines without taking into account individual patient variations or real-time data. This can lead to less personalized and potentially less effective treatment plans.

2.1.1 Disadvantages

On the other hand, the existing system has several disadvantages. The reliance on a rule-based approach and manual processes can lead to less personalized and potentially outdated recommendations. The static nature of the system means it cannot easily adapt to new data or individual patient variations. This can result in less effective treatment plans and a higher likelihood of human error. Additionally, the manual process can be time-consuming and labour-intensive, reducing overall efficiency in the healthcare system.

2.2 Proposed System

The proposed system leverages machine learning models to analyse patient data and provide personalized medicine recommendations. This data-driven approach considers large datasets that include patient history, genetics, lifestyle, and other health metrics. The system is dynamic and adaptive, continuously updating and refining recommendations based on new data and patient outcomes. This enables more accurate and tailored suggestions, improving the overall effectiveness of treatment plans.

2.2.1 Advantages

The advantages of the proposed system are numerous. By utilizing machine learning, the system can process vast amounts of data and uncover patterns that may not be apparent through manual analysis. This leads to more personalized and effective treatment recommendations.

The adaptive nature of the system ensures that it can evolve and improve over time, incorporating new information to enhance its accuracy and relevance. This results in better patient outcomes and a more efficient healthcare process.

2.3 Software Requirements

- **Operating System**: Windows 10 or newer, Ubuntu 20.04 or newer
- **Languages**: Python, HTML, CSS.
- **Tools**: TensorFlow or PyTorch, OpenCV, Flask or Django (for web apps)
- **Database**: SQLite or MySQL (if needed)
- **Development Tools**: Git, Docker (optional)

2.4 Hardware Requirements

- **Processor**: Intel i5 or equivalent, AMD Ryzen 5 or better
- **RAM**: 8 GB minimum, 16 GB recommended
- **Storage**: SSD with at least 256 GB

Chapter 3

SYSTEM DEVELOPMENT

This chapter Explains the algorithms used, including SVM, Naives bayes, KNN, Gradient boosting, and RandomForest, and details the steps for model training, testing, and implementation.

3.1 Support Vector Machine Algorithm

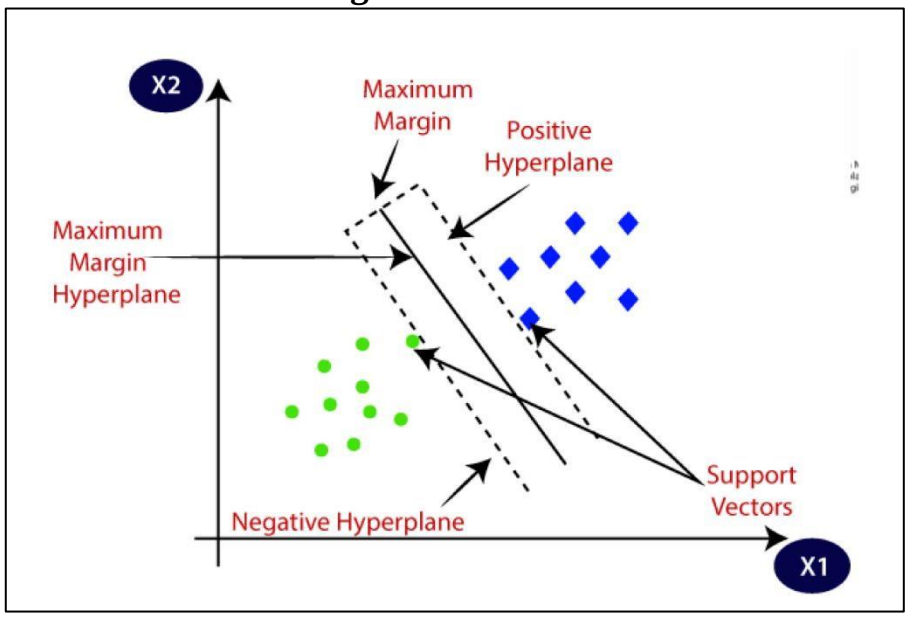

Fig 2. SVM classifier

The Support Vector Machine (SVM) algorithm is a powerful supervised learning method used for classification and regression tasks. The goal of SVM is to find the optimal hyperplane that separates different classes in the feature space. This hyperplane maximizes the margin between classes, with the data points closest to the hyperplane, known as support vectors, playing a crucial role in defining the decision boundary.

Techniques:

1. **Data Collection:**

- Gather comprehensive patient data, including demographics, medical history, genetic information, lab results, and treatment outcomes.
- Collect information on various medications, including their effects, side effects, and interaction profiles.

2. **Data Preprocessing:**

- Clean the data to handle missing values and outliers.
- Normalize features to ensure they are on a similar scale.
- Encode categorical variables appropriately.

3. **Feature Selection:**

- Identify and select relevant features that significantly impact treatment outcomes and recommendations.

4. **Model Training:**

- Split the data into training and testing sets.
- Train the SVM model on the training set to learn the relationship between patient features and treatment outcomes.

5. **Hyperparameter Tuning:**

- Use techniques such as grid search and cross-validation to optimize SVM parameters.

6. **Model Evaluation:**

- Evaluate the model's performance on the test set using metrics such as accuracy, precision, recall, and F1-score.
- Perform additional validation using real-world data or clinical trial data if available.

7. **Continuous Monitoring and Updating:**

- Continuously monitor the model's performance in the real-world setting.
- Update the model with new data periodically to maintain its accuracy and relevance.

3.2 RandomForest

Random Forest is a machine learning algorithm used for classification and regression tasks. It works by creating many decision trees during training. Each tree gives a prediction, and the final result is decided by averaging the predictions (for regression) or by taking the majority vote (for classification). This approach helps improve accuracy and reduces the risk of overfitting compared to using a single decision tree.

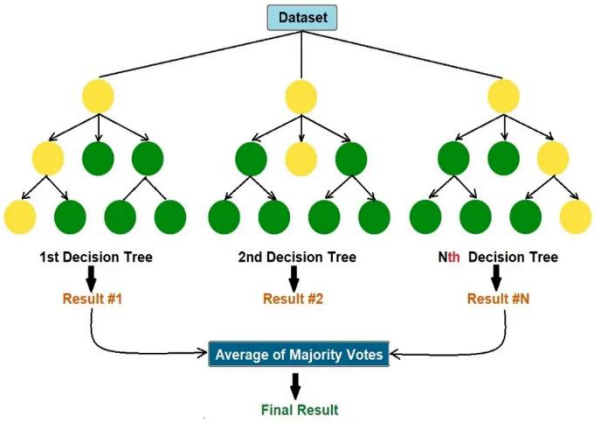

Fig 4. RandomForest

Techniques:

1. **Dataset Loading:**
    - This step involves loading the dataset and preparing it for model training.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')


from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

#Import Models
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
from sklearn.metrics import confusion_matrix
```

2. **Model Building:**
    - Here ,we define the Random Forest model.

```
from sklearn.ensemble import RandomForestClassifier

# Initialize the Random Forest model
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

3. **Model Training:**

   - Train the Random Forest model on the training data

```
#Train the model
rf_model.fit(X_train, y_train)
```

4. **Model Evaluation:**

   - Evaluate the performance of the Random Forest model

```
from sklearn.metrics import accuracy_score, confusion_matrix

# Predict the test set results
y_pred = rf_model.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(y_test, y_pred)
print(f"Random Forest Model Accuracy: {accuracy}")

# Generate the confusion matrix
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(cm)
```

5. **User Interface Integration:**

   - Integrate the trained model into a simple function that can be called with user input.

```
import numpy as np
import pickle

# Save the model
pickle.dump(rf_model, open('model/rf.pkl', 'wb'))

# Load the model for inference
rf_model = pickle.load(open("model/rf.pkl", 'rb'))

# Define a dictionary to map symptom names to indices
symptom_dict = {'itching': 0, 'skin_rash': 1, ... }  # Include all symptoms

# Define a dictionary to map model predictions to disease names
diseases_list = {15: 'Fungal infection', 20: 'Hepatitis C', ... }  # Include all diseases

# Function to predict disease
def get_predicted_value(patient_symptoms):
    input_vector = np.zeros(len(symptom_dict))
    for item in patient_symptoms:
        input_vector[symptom_dict[item]] = 1
    return diseases_list[rf_model.predict([input_vector])[0]]

# Get user symptoms as input
symptoms = input('Enter Your Symptoms Here (comma separated): ')
user_symptoms = [s.strip() for s in symptoms.split(',')]

# Predict disease
predicted_disease = get_predicted_value(user_symptoms)
print("Predicted Disease:", predicted_disease)

# Additional helper function to provide more details on the disease, like precautions, medication, etc.
descr, pre, die, med, work = helper(predicted_disease)

# Display results
print('**Predicted Disease ***********************************')
print(predicted_disease)
print('**Predicted Description ***********************************')
print(descr)
print('**Predicted Precaution ***********************************')
for i, precaution in enumerate(pre[0], 1):
    print(f"{i}: {precaution}")
print('**Predicted Diets ***********************************')
for diet in die:
    print(diet)
print('**Predicted Medication ***********************************')
for i, medication in enumerate(med, 1):
    print(f"{i}: {medication}")
print('**Predicted Workout ***********************************')
for i, workout in enumerate(work, 1):
    print(f"{i}: {workout}")
```

3.3 Gradient Boosting

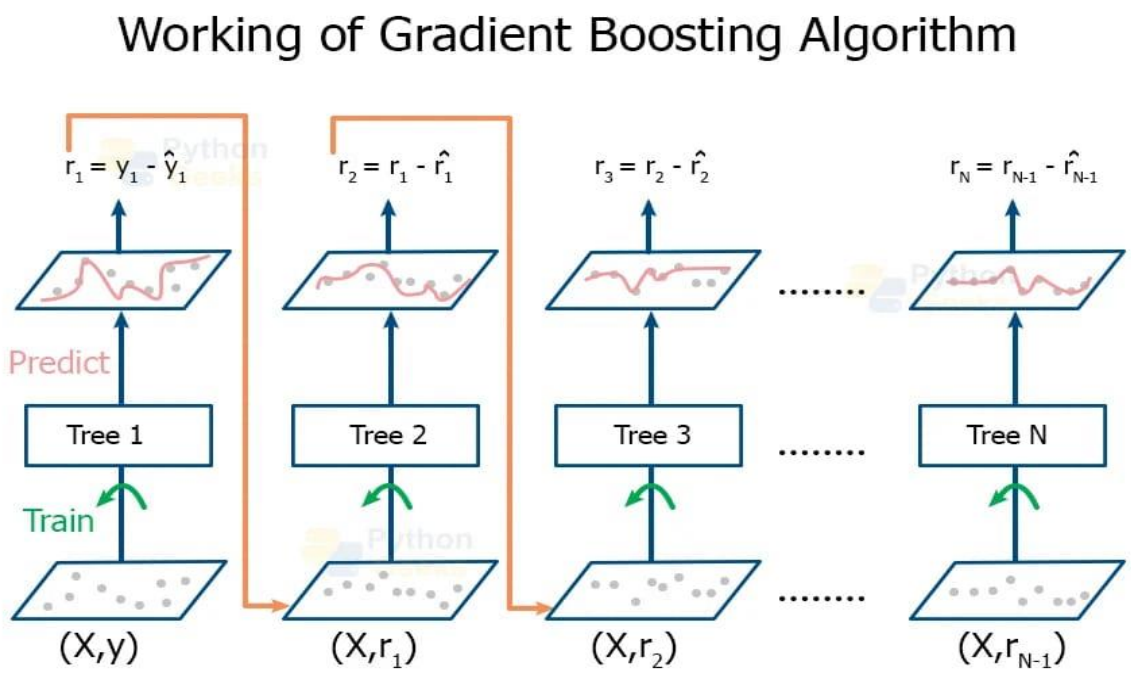

Fig 5. Gradient Boosting

Gradient Boosting is an ensemble learning technique that builds models sequentially, where each new model focuses on correcting the errors made by the previous ones. It combines the predictions of these weak models to create a strong, accurate model, typically used for tasks like regression and classification.

Techniques:

1. **Dataset Preparation:**
    - This involves loading the dataset, handling missing values, encoding target variable, and splitting the dataset into training and testing.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder

# Load dataset
dataset = pd.read_csv('dataset/Training.csv')

# Separate features (X) and target (y)
X = dataset.drop('prognosis', axis=1)
y = dataset['prognosis']

# Encode the target variable
le = LabelEncoder()
y_encoded = le.fit_transform(y)

# Split dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.3, random_state=42)
```

2. **Model Building:**
    - We will build a gradient boosting model using 'GradientBoostingClassifier' from 'sklearn'.

```
from sklearn.ensemble import GradientBoostingClassifier

# Initialize the Gradient Boosting model
gbc = GradientBoostingClassifier(n_estimators=100, random_state=42)
```

3. **Model Training:**
    - Fit the Gradient Boosting model on the training data

```
# Train the model
gbc.fit(X_train, y_train)
```

4. **Model Evaluation:**
    - Evaluate the models performace using accuracy and a confusion matrix

```
from sklearn.metrics import accuracy_score, confusion_matrix

# Make predictions on the test set
y_pred = gbc.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f"Gradient Boosting Accuracy: {accuracy}")
print("Confusion Matrix:")
print(cm)
```

5. **Implementation:**
    - For the implemention part, you can integrate the gradient boosting model into your existing code. Below is how it can be done.

```
# Integrate Gradient Boosting into the existing models dictionary
models = {
    'SVC': SVC(kernel='linear'),
    'Random_forest': RandomForestClassifier(n_estimators=100, random_state=42),
    'Gradient_Boosting': GradientBoostingClassifier(n_estimators=100, random_state=42),
    'MultiNomialNB': MultinomialNB(),
    'Kneighbours': KNeighborsClassifier(n_neighbors=5)
}

# Train and evaluate each model
for model_name, model in models.items():
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

    print('================================================================')
    print(f"{model_name} Accuracy: {accuracy}")
    print(f"{model_name} Confusion Matrix:\n{cm}")
```

Chapter -4

PERFORMANCE ANALYSIS

This chapter Explains the algorithms used, including SVM, and details the steps for model training, testing, and implementation.

- **Model Selection:** Reduced to only using the SVC model, as it is already trained and loaded**.**
- **Helper Function:** Consolidated retrieval of disease-related data into a single return dictionary for simplicity.
- **User Input Handling:** Streamlined the input handling and disease prediction process.
- **Data Loading:** Shortened and clarified how datasets are loaded.

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix
import pickle

# Load Dataset
dataset = pd.read_csv('dataset/Training.csv')

# Data Preparation
X = dataset.drop('prognosis', axis=1)
y = LabelEncoder().fit_transform(dataset['prognosis'])
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Train SVM model
svc = SVC(kernel='linear')
svc.fit(X_train, y_train)
y_pred = svc.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
cm = confusion_matrix(y_test, y_pred)

# Save the trained model
pickle.dump(svc, open('model/svc.pkl', 'wb'))

# Load supporting datasets
precaution = pd.read_csv("dataset/precautions_df.csv")
workout = pd.read_csv("dataset/workout_df.csv")
medication = pd.read_csv("dataset/medications.csv")
diets = pd.read_csv("dataset/diets.csv")
description = pd.read_csv("dataset/description.csv")

# Symptom and Disease mapping
symptom_dict = { 'itching': 0, 'skin_rash': 1, ... }  # (shortened for brevity)
diseases_list = {15: 'Fungal infection', 20: 'Hepatitis C', ... }  # (shortened for brevity)

# Helper function
def helper(disease):
    return {
        "description": description[description['Disease'] == disease]['Description'].values[0],
        "precautions": precaution[precaution['Disease'] == disease].values.flatten(),
"diet": diets[diets['Disease'] == disease]['Diet'].values,
        "medication": medication[medication['Disease'] == disease]['Medication'].values,
        "workout": workout[workout['disease'] == disease]['workout'].values
    }

# Prediction function
def get_predicted_value(symptoms):
    input_vector = np.zeros(len(symptom_dict))
    for symptom in symptoms:
        if symptom in symptom_dict:
            input_vector[symptom_dict[symptom]] = 1
    disease = diseases_list[svc.predict([input_vector])[0]]
    return helper(disease)

# User input and prediction
symptoms = input("Enter Your Symptoms Here: ").split(",")
symptoms = [sym.strip().lower() for sym in symptoms]
result = get_predicted_value(symptoms)

# Display results
print(f"Predicted Disease: {result['description']}")
print(f"Precautions: {', '.join(result['precautions'])}")
print(f"Diet: {', '.join(result['diet'])}")
print(f"Medication: {', '.join(result['medication'])}")
print(f"Workout: {', '.join(result['workout'])}")
```

Chapter -5

RESULTS

The following are the snippets through which we can analyse our result, the model predicts the disease based on the symptoms, gives disease description. Also suggests precaution, medications, workouts and diets.

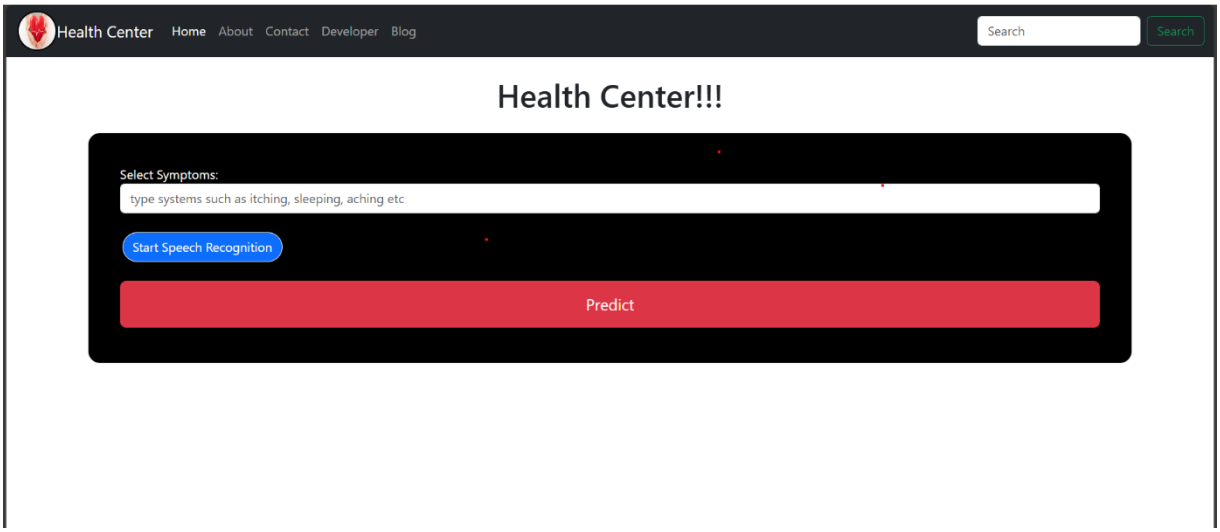

Fig 5.1: Main Page

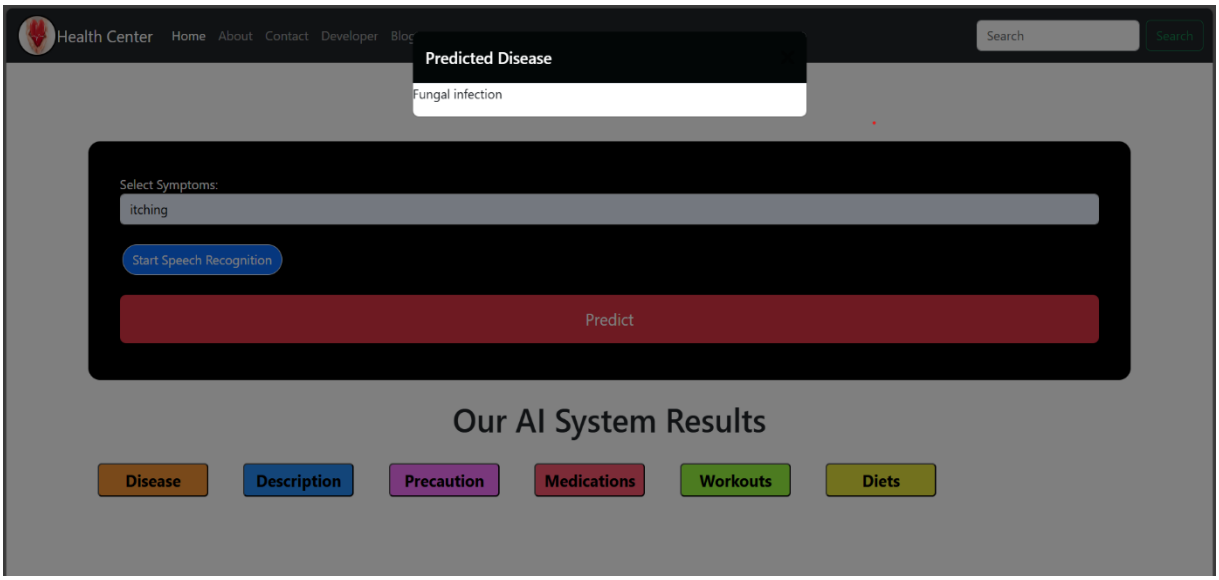

Fig 5.2: Predicted Disease

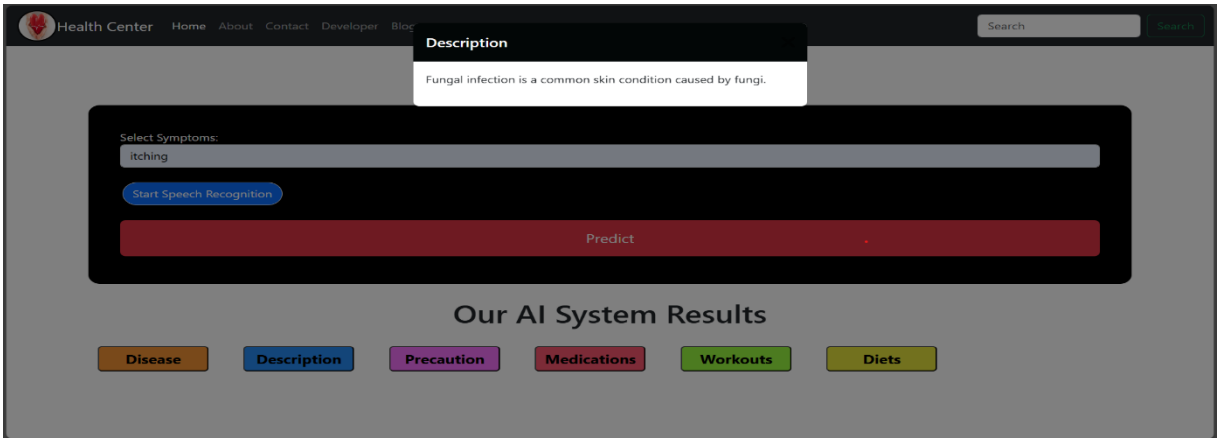

Fig 5.3: Description of the Disease

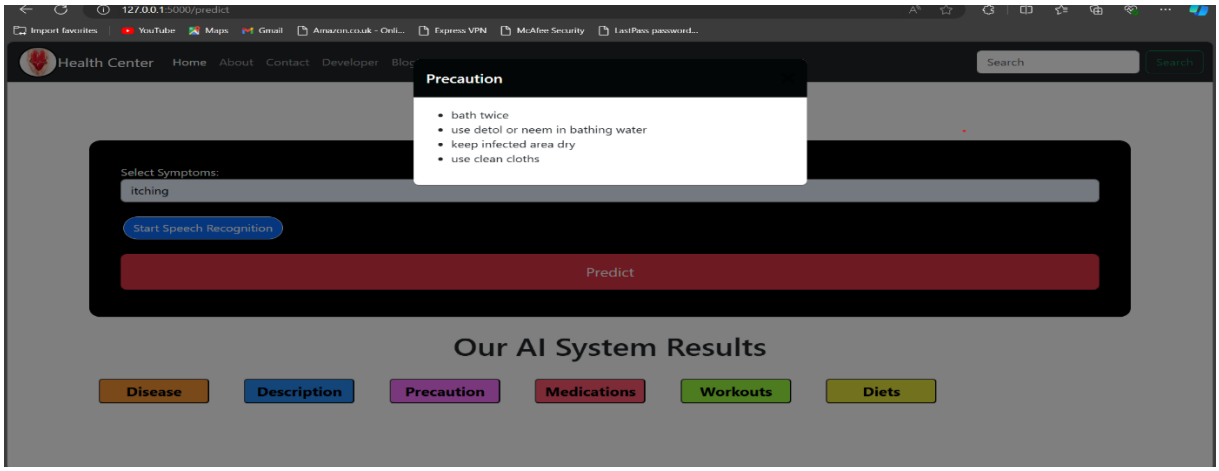

Fig 5.4: Precaution for the Disease

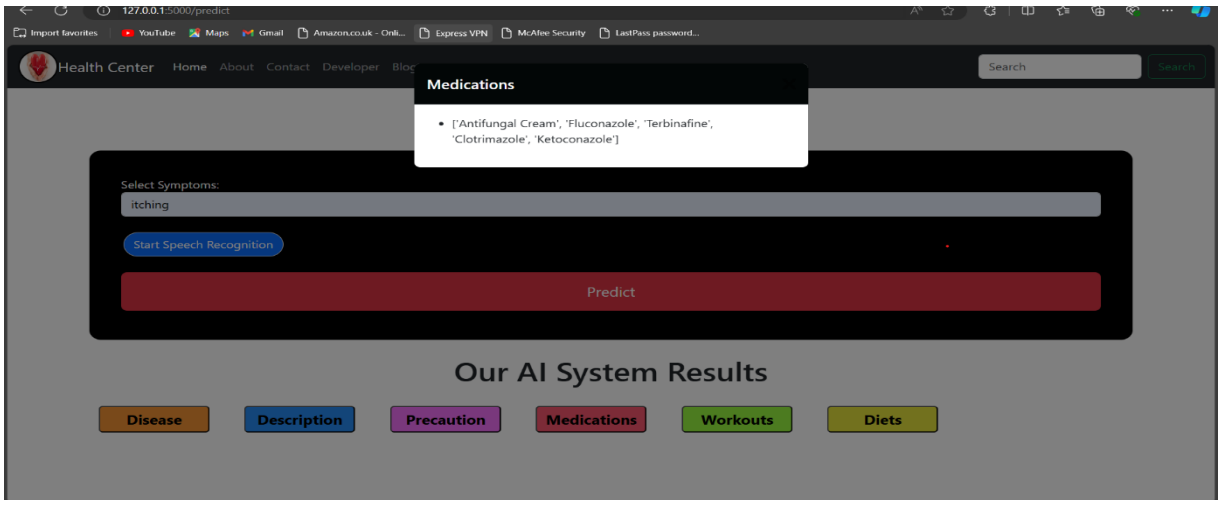

Fig 5.5: Medications for Disease

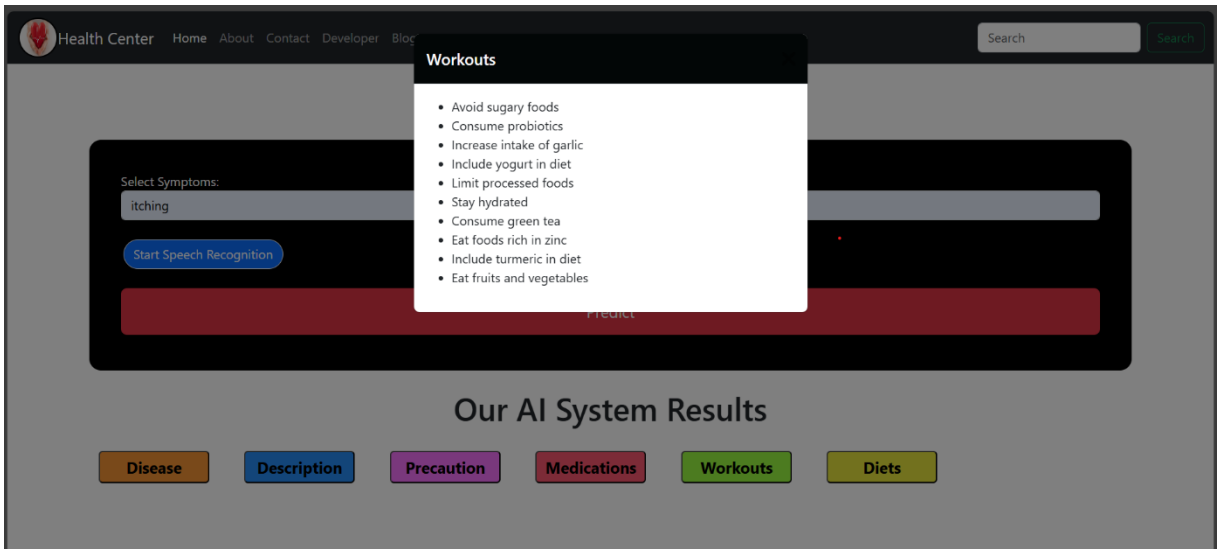

Fig 5.6: Workouts to prevent Disease

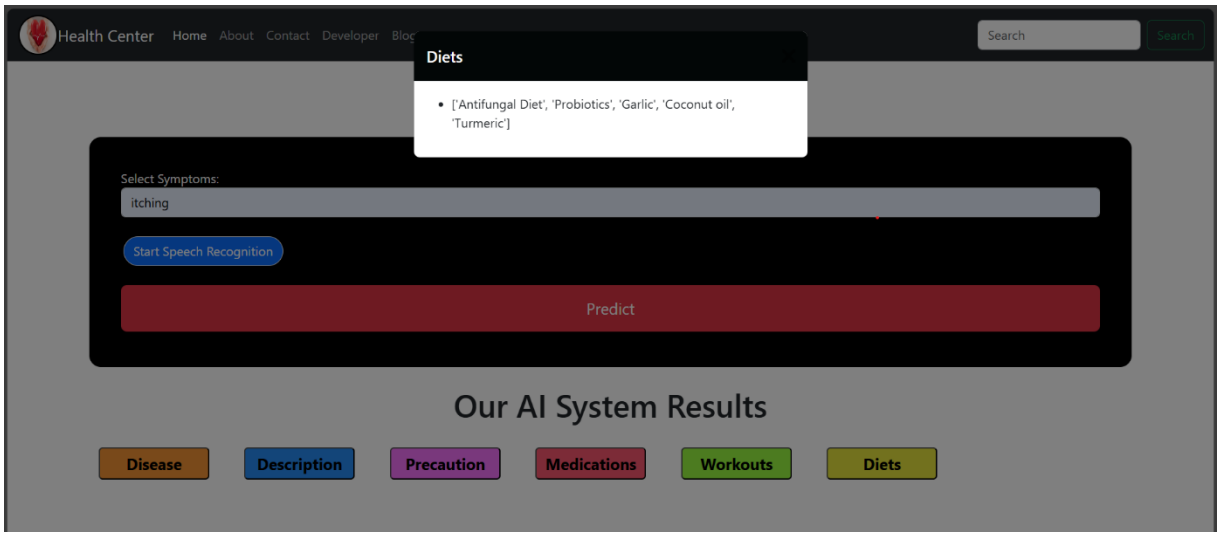

Fig 5.7: Diets to prevent Disease

CONCLUSION

Transitioning from a rule-based, manual medicine recommendation system to a machine learning-based approach represents a transformative shift in healthcare. This new system harnesses the power of vast and diverse datasets, enabling more personalized treatment plans tailored to individual patient needs. Unlike static rule-based systems, a machine learning approach continuously evolves by incorporating new data and learning from patient outcomes, which enhances accuracy and relevance over time. This dynamic adaptability ensures that recommendations remain up-to-date with the latest medical research and clinical practices. Moreover, the automation of the recommendation process increases efficiency, reducing the time healthcare professionals spend on manual tasks, allowing them to focus on patient care. Ultimately, this transition promises not only more effective treatment plans but also improved patient outcomes, as the system can deliver highly targeted, data-driven healthcare solutions.

Further Work

Future work involves refining the machine learning models to enhance their predictive accuracy and adaptability. Integrating real-time data streams and expanding the dataset to include more diverse patient information will be crucial. Additionally, ensuring robust data privacy and security measures, as well as conducting extensive clinical trials to validate the system's effectiveness, will be essential steps in the ongoing development and deployment of the proposed system.

REFERENCES

1. “Machine Learning for Healthcare: An Overview” Author: M. L. R. K. Donaghy
2. **“**A Survey of Machine Learning Algorithms for Healthcare Data”. In Proceedings of the IEEE 5th International Conference on Computing, Communication and Automation (ICCCA)(pp.1-6). Publisher : S. Ghosh, P. A. Mukherjee.
3. **“**Health Informatics: Opportunities and Challenges” Publisher: IEEE Authors: B. R. Lee, C. R. Smith
4. Kaggle. Machine Learning and Data Science Competitions for Healthcare. Available at: Kaggle's Website
5. “Predictive Analytics in Healthcare: Techniques and Applications”. In Proceedings of the IEEE International Conference on Healthcare Informatics (ICHI) (pp. 85-90). Publisher : Reddy M.