K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Mini Project ENSI-152

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

TEAM LEADER

- JAGRITI UPADHYAY - 2401730070

TEAM CO-MEMBERS

1. PRIYA BORA - 2401730119
2. RIDHIMA SINGH - 2401730133
3. VISHU SANGWAN – 2401730090

PROJECT INTRODUCTION AND METHODOLOGY

Project Overview

- A **Medicine Recommendation System (MRS)** is a tool that assists healthcare providers or patients in identifying the most suitable medications based on symptoms, diagnosis, patient history, and other factors. It leverages **data science, artificial intelligence (AI), and machine learning (ML)** techniques to improve treatment accuracy. our goal is to build such a recommendation model which will help the patients for the use of **right drugs high accuracy and efficiency** is very critical for such a recommender system
-

Working Of Project

1. Our system recommend the medicine for **first care.**
2. A place where hospitals are **not available easily.**
3. For the **travelers.**
4. Use for normal or **daily based symptoms.**
5. Automating Prescription: **Reduce time and effort** in choosing the best medication.

Methodology

1. **Data Collection and Preprocessing**
   The initial phase involved collecting a relevant medical dataset and cleaning it using Python libraries such as **NumPy** and **Pandas**. This preprocessing included handling missing values, normalizing data formats, and preparing the data for analysis.
2. **Exploratory Data Analysis (EDA)**
   Using **Jupyter Notebook**, various insights were drawn from the dataset to understand trends, correlations, and user needs. This helped in identifying the right features to be used for generating recommendations.
3. **Backend Development (Python + Flask)**
   A Flask-based backend was developed to manage routing and handle user interactions. Specific Flask routes were created to process user inputs, generate medicine recommendations, and return results dynamically.
4. **Frontend Design (HTML + Bootstrap)**
   A user-friendly web interface was designed using **HTML** and **Bootstrap** to ensure responsiveness and accessibility. The UI allows users to input symptoms or conditions and view recommended medicines.

Project Code

1.Home page

templates > index.html > html > body

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Medicine Recommendation System</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4Q6Gf2aSP4eDXB8Miphtr37CMZZQ5oXLH2yaXMJ2w8e2ZtHTl7G
  </head>
  <body>
    <!-- Navbar -->
    <nav class="navbar navbar-expand-lg navbar-dark bg-dark">···
      </nav>

    <h1 class="text-center mt-3" style="transition: transform 0.3s, opacity 0.3s; opacity: 0;" id="animated-title">Health Care Center</h1>
    <script>
      document.addEventListener("DOMContentLoaded", function() {
      const title = document.getElementById("animated-title");
      setTimeout(() => {
        title.style.opacity = "1";
        title.style.transform = "translateY(10px)";
      }, 300);
      });
    </script>

    <!-- main part -->
    <div class="container mt-5 my-4 p-4" style="background-color: #343a40; border-radius: 10px; color: white;">
      <form action="/predict" method="post">···
      </form>
    </div>

    <!-- Highlights section -->
    <div class="container mt-4">
      {% if predicted_disease %}
        <h1 class="text-center my-4">Our AI Docs Results</h1>
        <div class="result-container text-center d-flex justify-content-center flex-wrap gap-3">
          <!-- Disease Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#diseaseModal">Disease</button>
          <!-- Description Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#descriptionModal">Description</button>
          <!-- Precaution Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#precautionModal">Precautions</button>
          <!-- Medication Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#medicationModal">Medications</button>
          <!-- Workout Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#workoutModal">Workouts</button>
          <!-- Diet Button -->
          <button class="toggle-button btn btn-dark mx-2" data-bs-toggle="modal" data-bs-target="#dietsModal">Diets</button>
        </div>

<!-- Modal for Disease -->
<div class="modal fade" id="diseaseModal" tabindex="-1" aria-labelledby="diseaseModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="diseaseModalLabel">Predicted Disease</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p>{{ predicted_disease }}</p>
      </div>
    </div>
  </div>
</div>

<!-- Modal for Description -->
<div class="modal fade" id="descriptionModal" tabindex="-1" aria-labelledby="descriptionModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="descriptionModalLabel">Disease Description</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <p>Description of {{ predicted_disease }} will go here.</p>
      </div>
    </div>
  </div>
</div>

    <!-- Modal for Precautions -->
    <div class="modal fade" id="precautionModal" tabindex="-1" aria-labelledby="precautionModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="precautionModalLabel">Precautions</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
          <div class="modal-body">
            <p>Precautions for {{ predicted_disease }} will go here.</p>
          </div>
        </div>
      </div>
    </div>
```

```
    <!-- Modal for Medications -->
    <div class="modal fade" id="medicationModal" tabindex="-1" aria-labelledby="medicationModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="medicationModalLabel">Medications</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
          <div class="modal-body">
            <p>Recommended medications for {{ predicted_disease }} will go here.</p>
          </div>
        </div>
      </div>
    </div>
```

```
    <!-- Modal for Workouts -->
    <div class="modal fade" id="workoutModal" tabindex="-1" aria-labelledby="workoutModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="workoutModalLabel">Workouts</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
          <div class="modal-body">
            <p>Suggested workouts for {{ predicted_disease }} will go here.</p>
          </div>
        </div>
      </div>
    </div>
```

```
    <!-- Modal for Diets -->
    <div class="modal fade" id="dietsModal" tabindex="-1" aria-labelledby="dietsModalLabel" aria-hidden="true">
      <div class="modal-dialog">
        <div class="modal-content">
          <div class="modal-header">
            <h5 class="modal-title" id="dietsModalLabel">Diets</h5>
            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
          </div>
          <div class="modal-body">
            <p>Recommended diets for {{ predicted_disease }} will go here.</p>
          </div>
        </div>
      </div>

    {% endif %}
  </div>

  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.6/dist/js/bootstrap.bundle.min.js" integrity="sha384-j1CDi7MgGQ12Z7Qab0qlWQ/Qqz
</body>
</html>
```

2. Main Python File

main.py > predict

```python
from flask import Flask, request, render_template, jsonify  # Import jsonify
import numpy as np
import pandas as pd
import pickle

# flask app
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'))

```

```python
# Helper function to get disease info
def helper(dis):
    # Description
    desc = description[description['Disease'] == dis]['Description']
    desc = " ".join([w for w in desc]) if not desc.empty else "No description available."

    # Precautions
    pre = precautions[precautions['Disease'] == dis][['Precaution_1', 'Precaution_2', 'Precaution_3', 'Precaution_4']]
    pre = [", ".join([str(val) for val in row if pd.notnull(val)]) for row in pre.values]  # Join non-null precautions

    # Medications
    med = medications[medications['Disease'] == dis]['Medication']
    med = ", ".join([str(m) for m in med]) if not med.empty else "No medications available."

    # Diets
    die = diets[diets['Disease'] == dis]['Diet']
    die = ", ".join([str(d) for d in die]) if not die.empty else "No specific diet available."

    # Workouts
    wrkout = workout[workout['disease'] == dis]['workout']
    wrkout = ", ".join([str(w) for w in wrkout]) if not wrkout.empty else "No specific workout available."

    return desc, pre, med, die, wrkout
```

```python


# Dictionary for symptoms and diseases
symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills': 5, 'joint_pai
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic ulcer diseae'

# 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]]

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

```python
@app.route('/predict', methods=['GET', 'POST'])
def predict():
    if request.method == 'POST':
        symptoms = request.form.getlist('symptoms')
        # Split the symptoms string by commas if it comes as a single string
        user_symptoms = [s.strip() for s in symptoms[0].split(',')]  # Assuming symptoms are coming as a comma-separated string

        predicted_disease = get_predicted_value(user_symptoms)

        desc, pre, med, die, wrkout = helper(predicted_disease)

        return render_template('index.html', predicted_disease=predicted_disease, desc=desc, pre=pre, med=med, die=die, wrkout=wrkout)
```

```python
@app.route('/about')
def about():
    return render_template("about.html")

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

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

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

if __name__ == '__main__':
    app.run(debug=True)
```

3. DATASETS

Title: Personalized Medical Recommendation System with Machine Learning

Description:

Welcome to our cutting-edge Personalized Medical Recommendation System, a powerful platform designed to assist users in understanding and managing their health. Leveraging the capabilities of machine learning, our system analyzes user-input symptoms to predict potential diseases accurately.

load dataset & tools

```python
import pandas as pd
```

```python
dataset = pd.read_csv('Datasets/Training.csv')
```

```python
dataset
```

	itching	skin_rash	nodal_skin_eruptions	continuous_sneezing	shivering	chills	joint_pain	stomach_pain	acidity	ulcers_on_tongue	...	blackheads	scurring	skin_peeling	silver_like_dusting	small_dents_in_nails	inflammatory_nails	blister	red_sore_around_nose	yellow_crust_ooze	progi
0	1	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Fu infe
1	0	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Fu infe
2	1	0	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Fu infe
3	1	1	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Fu infe
4	1	1	1	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Fu infe
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4915	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	(ver Paroy Posit Ve
4916	0	1	0	0	0	0	0	0	0	0	...	1	1	0	0	0	0	0	0	0	
4917	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	Ur

4920 rows × 133 columns

```python
# vals = dataset.values.flatten()
```

```python
dataset.shape
```

```
(4920, 133)
```

train test split

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

```python
X = dataset.drop('prognosis', axis=1)
y = dataset['prognosis']

# ecoding prognonsis
le = LabelEncoder()
le.fit(y)
Y = le.transform(y)

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=20)
```

Training top models

```
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
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
import numpy as np


# Create a dictionary to store models
models = {
    '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()
}

# Loop through the models, train, test, and print results
for model_name, model in models.items():
    # Train the model
    model.fit(X_train, y_train)

    # Test the model
    predictions = model.predict(X_test)

    # Calculate accuracy
    accuracy = accuracy_score(y_test, predictions)
    print(f"{model_name} Accuracy: {accuracy}")

    # Calculate confusion matrix
    cm = confusion_matrix(y_test, predictions)
    print(f"{model_name} Confusion Matrix:")
    print(np.array2string(cm, separator=', '))

    print("\n" + "="*40 + "\n")
```

[7] Python

```
SVC Accuracy: 1.0
SVC Confusion Matrix:
[[40,  0,  0, ...,  0,  0,  0],
 [ 0, 43,  0, ...,  0,  0,  0],
 [ 0,  0, 28, ...,  0,  0,  0],
 ...,
 [ 0,  0,  0, ..., 34,  0,  0],
 [ 0,  0,  0, ...,  0, 41,  0],
 [ 0,  0,  0, ...,  0,  0, 31]]

========================================

RandomForest Accuracy: 1.0
RandomForest Confusion Matrix:
[[40,  0,  0, ...,  0,  0,  0],
 [ 0, 43,  0, ...,  0,  0,  0],
 [ 0,  0, 28, ...,  0,  0,  0],
 ...,
 [ 0,  0,  0, ..., 34,  0,  0],
 [ 0,  0,  0, ...,  0, 41,  0],
 [ 0,  0,  0, ...,  0,  0, 31]]

========================================

GradientBoosting Accuracy: 1.0
...
 [ 0,  0,  0, ...,  0,  0, 31]]

========================================
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

single prediction

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

[8] Python

```
1.0
```

```
# save svc
import pickle
pickle.dump(svc,open('svc.pkl','wb'))
```

[9] Python

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

[10] Python

```
# test 1:
print("predicted disease :",svc.predict(X_test.iloc[0].values.reshape(1,-1)))
print("Actual Disease :", y_test[0])
```

[11] Python

```
predicted disease : [40]
Actual Disease : 40
c:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

```
# test 2:
print("predicted disease :",svc.predict(X_test.iloc[100].values.reshape(1,-1)))
print("Actual Disease :", y_test[100])
```

[12] Python

```
predicted disease : [39]
Actual Disease : 39
c:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

Recommendation System and Prediction

load database and use logic for recommendations

```
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")
```

[13] Python

```
#-------------------------------------------------------------
# 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

symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills': 5, 'joint_pain': 6, 'stomach_pain': 7, 'acidity': 8, 'ulcers_on_tongue': 9, 'muscle_wasting': 10, 'vomiting': 11, 'bur
diseases_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic ulcer diseae', 1: 'AIDS', 12: 'Diabetes ', 17: 'Gastroenteritis', 6: 'Bronchial Asthma', 23: 'Hypertension ', 30: 'Mig
```

```
# 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]]
```

[14] Python

```
# 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
```

```
==================predicted disease============
Fungal infection
==================description==================
Fungal infection is a common skin condition caused by fungi.
==================precautions==================
1 :  bath twice
2 :  use detol or neem in bathing water
3 :  keep infected area dry
4 :  use clean cloths
==================medications==================
5 :  ['Antifungal Cream', 'Fluconazole', 'Terbinafine', 'Clotrimazole', 'Ketoconazole']
==================workout==================
6 :  Avoid sugary foods
7 :  Consume probiotics
8 :  Increase intake of garlic
9 :  Include yogurt in diet
10 :  Limit processed foods
11 :  Stay hydrated
12 :  Consume green tea
13 :  Eat foods rich in zinc
14 :  Include turmeric in diet
15 :  Eat fruits and vegetables
==================diets==================
16 :  ['Antifungal Diet', 'Probiotics', 'Garlic', 'Coconut oil', 'Turmeric']
c:\Users\HP\anaconda3\Lib\site-packages\sklearn\base.py:493: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

```
# 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,blister
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
```

Python

```
Enter your symptoms.......yellow_crust_ooze,red_sore_around_nose,small_dents_in_nails,inflammatory_nails,blister
==========predicted disease============
Impetigo
==========description==========
Impetigo is a highly contagious skin infection causing red sores that can break open.
==========precautions==========
1 :  soak affected area in warm water
2 :  use antibiotics
3 :  remove scabs with wet compressed cloth
4 :  consult doctor
==========medications==========
5 :  ['Topical antibiotics', 'Oral antibiotics', 'Antiseptics', 'Ointments', 'Warm compresses']
==========workout==========
6 :  Maintain good hygiene
7 :  Stay hydrated
8 :  Consume nutrient-rich foods
9 :  Limit sugary foods and beverages
10 :  Include foods rich in vitamin C
11 :  Consult a healthcare professional
12 :  Follow medical recommendations
13 :  Avoid scratching
14 :  Take prescribed antibiotics
15 :  Practice wound care
==========diets==========
16 :  ['Impetigo Diet', 'Antibiotic treatment', 'Fruits and vegetables', 'Hydration', 'Protein-rich foods']
C:\Users\naimat\anaconda3\lib\site-packages\sklearn\base.py:465: UserWarning: X does not have valid feature names, but SVC was fitted with feature names
  warnings.warn(
```

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

[16] Python

```
1.5.1
```