HealthTrack'D

A Disease Predicting Program Based on Symptoms

Using Python

A PROJECT REPORT

Submitted by

Roshan

in partial fulfilment for the award of the

the AI Winter Internship Program

in

Brainy n Bright

UAE ,Dubai: Office 206, Platinum Business Center, Al Nahda

December 2024 – January 2025

ABSTRACT

This project, called **HealthTrack'D**, is a simple program that helps predict diseases based on symptoms. It uses a tool called a **Decision Tree Classifier**, which is a type of machine learning, to learn from a list of diseases and their symptoms. The data comes from a **CSV** file that has all the diseases, symptoms, and descriptions.

The program works like this:

1. It reads the data and organizes it so that every symptom is unique and easy to use.
2. It trains three different machine learning models to predict diseases based on the symptoms you give it.
3. It has a user-friendly interface made with **Tkinter**, where you can type in two symptoms, and it gives back up to three diseases.
4. If you type something that isn't in the list of symptoms, it shows an error message so you can fix it.

HealthTrack'D makes it easy for people to guess what might be wrong with them based on their symptoms. It's an easy way to use technology and machine learning to help people.

Table of Contents

Chapter 1

Introduction

HealthTrack'D is a symptom-based disease prediction application built using Python. It leverages machine learning to assist users in diagnosing potential diseases based on the symptoms they input. By integrating a graphical user interface (GUI) designed with Tkinter, the application provides an accessible and user-friendly experience for health diagnostics.

1.1 Objective

The primary objective of this project is to design and implement a symptom-based disease prediction tool that serves as a preliminary diagnostic aid. The specific objectives include:

1. Enhance Accessibility to Health Diagnostics
    - Provide a user-friendly platform where doctors can input symptoms and receive potential diagnoses, reducing the need for immediate medical consultations for minor health concerns.
2. Utilize Machine Learning for Prediction
    - Employ machine learning algorithms, specifically decision tree classifiers, to analyze and predict diseases based on symptoms entered by users.
3. Simplify the Symptom to Diagnosis Process
    - Transform raw symptom data into a structured format that allows for efficient and accurate disease prediction, ensuring timely results.
4. Provide Variability in Predictions
    - Use multiple classifiers to generate a broader range of predictions, allowing users to consider different possibilities for their symptoms.

The overarching goal is to bridge the gap between symptom identification and professional medical advice, empowering doctors to take informed steps toward their health management while understanding the limitations of such predictive tools.

1.2 Problem statement

Doctors often encounter challenges in quickly correlating a patient's symptoms with potential diseases, especially when dealing with a large volume of cases or rare symptom combinations. While medical expertise remains paramount, there is a need for supplemental tools that assist healthcare professionals in making faster, data-driven preliminary diagnoses.

The specific challenges include:

1. **Time-Consuming Symptom Analysis**: Analyzing and correlating multiple symptoms to potential diseases can be time-intensive, especially for less common conditions.
2. **Need for Enhanced Diagnostic Support**: Doctors could benefit from decision-support tools that provide data-backed predictions, allowing them to focus on confirming diagnoses and determining treatment plans.
3. **Managing Information Overload**: With a vast amount of medical knowledge and evolving disease data, it becomes challenging to recall all possible disease-symptom associations accurately.
4. **Improving Diagnostic Efficiency**: In busy healthcare settings, having a quick reference tool can help prioritize cases that require immediate attention.

This software aims to empower doctors by providing an intelligent, machine learning-based tool that predicts potential diseases from symptoms entered. It is designed as a diagnostic aid, enhancing the efficiency, accuracy, and speed of preliminary diagnoses while allowing healthcare professionals to focus on their expertise and patient care.

1.3 Project Overview

HealthTrack'D is a machine learning-based tool designed to assist doctors in preliminary diagnosis by analyzing patient symptoms and predicting potential diseases. Utilizing Decision Tree Classifiers and a user-friendly interface, it enables healthcare professionals to input symptoms and receive real-time, data-driven predictions. This tool enhances diagnostic efficiency, supports rare case identification, and serves as a supplemental aid in clinical decision-making.

1.4 Introduction to Machine Learning

Machine learning enables software applications to improve their accuracy in predicting outcomes without explicit programming. It is broadly categorized into three types: **Supervised Learning**, **Unsupervised Learning**, and **Reinforcement Learning**.

1. **Supervised Learning** involves training a model using labeled input and output data, allowing it to learn the relationship between the two.
2. **Unsupervised Learning** is a self-directed approach where the system identifies patterns and structures in unlabeled data without predefined categories.
3. **Reinforcement Learning** focuses on training models to make a sequence of decisions by learning from feedback based on actions taken within an environment.

Applications of machine learning span various fields, including medical diagnosis, image recognition, traffic prediction, product recommendations, self-driving cars, and speech recognition.

Chapter 2

System Description

The language used to code the algorithm is python. Machine learning models are trained using datasets. In HealthTrack'D it uses a Dataset of 400 Diseases, multiple symptoms and many treatments. The user enters two symptoms as input. The detected diseases displayed on the screen.

2.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms and can be freely distributed.

2.2 NumPy

NumPy (Numerical Python) is a Python library designed for efficient numerical computations. It provides support for multi-dimensional arrays (ndarray) and a wide range of mathematical operations, including element-wise computations and linear algebra. Known for its speed and memory efficiency compared to Python lists, NumPy is widely used in data science, machine learning, and scientific research, and integrates seamlessly with libraries like Pandas and Matplotlib.

2.3 Pandas

Pandas is a Python library ideal for working with structured data, particularly CSV files. It simplifies reading, writing, and manipulating CSV data through its powerful DataFrame structure. With Pandas, tasks like filtering rows, selecting columns, and applying transformations become efficient and intuitive, making it a go-to tool for handling CSV data in Python.

2.4 Tkinter

Tkinter is a Python library used to create graphical user interfaces (GUIs). It provides various widgets, like buttons, labels, and text boxes, to build interactive applications. As a wrapper around the Tcl/Tk GUI toolkit, Tkinter is cross-platform and comes pre-installed with Python, making it easy to create simple desktop applications without needing extra installations.

2.5 Decision Tree Classifier

The Decision Tree Classifier in Scikit-learn is a tool for sorting data into categories. It works like a flowchart, asking questions about the data to split it into groups. Each question helps narrow down the options until it reaches a final decision or class. You can adjust settings like how deep the tree grows to make it work better and avoid mistakes. It's easy to use and understand, making it great for tasks like predicting categories or labels.

Chapter 3

System Design

3.1 Collection of Data

The Diseases and Symptoms dataset, sourced from Kaggle, contains information about various diseases and their associated symptoms. It includes columns such as Disease Name, Symptoms, and Diagnosis. The dataset is used to train machine learning models, specifically Decision Tree Classifier models, to predict potential diseases based on user-input symptoms. The symptoms are preprocessed and encoded into a binary matrix, with each row representing a disease and each column corresponding to a symptom. This dataset enables the creation of a diagnostic tool that can assist in predicting diseases based on the symptoms provided by the user. The dataset can be accessed from Kaggle here.

3.2 Model Development

For the development of the diagnostic tool, three Decision Tree Classifier models were trained using the preprocessed Diseases and Symptoms dataset. The dataset, consisting of disease names, symptoms, and diagnoses, was encoded into a binary matrix representing the presence or absence of each symptom for every disease. The classifiers were trained to predict the disease based on user-input symptoms, with each model being initialized with different random states to ensure variance in each model's results. The models were evaluated based on their ability to predict the disease accurately, and their predictions are displayed to the user in the graphical user interface (GUI). The final model output includes three potential diagnoses, providing a range of predictions based on the input symptoms.

3.3 Decision Tree Algorithm

The Decision Tree algorithm is a supervised machine learning technique used for both classification and regression tasks. In this project, it was employed for classification to predict diseases based on user-provided symptoms. The algorithm splits the data into subsets based on feature values, creating a tree-like structure where each node represents a decision rule, and each leaf node corresponds to a predicted class label (in this case, a disease diagnosis). The tree is constructed by recursively selecting the feature that provides the best split at each node, using criteria like Gini impurity or information gain. Decision trees are interpretable, making them suitable for medical applications where understanding the decision-making process is essential. In this project, multiple decision trees were trained on the symptom-disease data, providing predictions that are easy to interpret and explain to users.

3.4 Model Algorithm

The model uses the **Decision Tree** algorithm, which is a classification method that predicts outcomes based on input features. The algorithm works by splitting the data into smaller groups using the best feature at each step. Each split creates a decision node, and the process continues until the data is classified into a final group or "leaf" node. In this project, three different decision tree models were trained using different random states to improve prediction accuracy. The models predict diseases based on user-input symptoms, and the simplicity of decision trees makes them easy to understand and interpret, which is important for medical applications.

Chapter 4

Program Functionality and Workflow

The system is designed to predict diseases based on user-provided symptoms. The design consists of two main components: the **Backend Model** and the **Frontend User Interface (UI)**.

4.1 Backend Model

The backend model is based on the **Decision Tree** algorithm, a classification technique used to predict diseases from input symptoms. The system uses three different **Decision Tree Classifier** models, each trained on a dataset containing diseases and their corresponding symptoms. The model processes the input symptoms by converting them into a binary matrix, where each symptom is either present (1) or absent (0) for a particular disease. This allows the system to predict potential diseases based on user input.

The system is designed to manage multiple classifiers, providing a range of predictions to increase the likelihood of an accurate diagnosis. Each classifier works by recursively splitting the symptom data into subsets, classifying the disease based on the provided symptoms.

4.2 Frontend User Interface (UI)

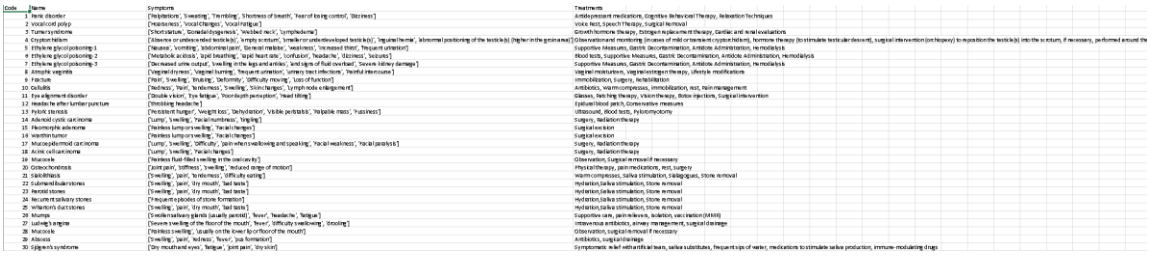
The **User Interface** (UI) is built using **Tkinter**, a Python library that allows for the creation of simple graphical applications. The UI is designed to be user-friendly, where users can input their symptoms and receive disease predictions. The interface consists of the following key elements:

- **Symptom Input Fields**: Users can enter up to two symptoms.
- **Prediction Results**: After entering the symptoms, the system displays three predicted diagnoses based on the trained models.
- **Buttons**: A "Predict Diagnosis" button to trigger the prediction and a "Quit" button to exit the application.

Figure 4.1 Program Screenshot

4.3 Dataset

The dataset used in the system is the **Diseases and Symptoms** dataset, sourced from Kaggle. This dataset includes a list of diseases along with their symptoms, which is used to train the decision tree models. Each row in the dataset represents a disease, with associated symptoms and a diagnosis. Here's a sample of the dataset:



Code	Name	Symptoms	Treatments
1	Panic disorder	['Hyperventilation', 'Sweating', 'Trembling', 'Shortness of breath', 'Fear of losing control', 'Dizziness']	Antidepressant medications, Cognitive Behavioral Therapy, Relaxation techniques
2	Vocal cord polyp	['Hoarseness', 'Vocal changes', 'Vocal fatigue']	Voice rest, Speech therapy, Surgical removal
3	Turner syndrome	['Short stature', 'Gonadal dysgenesis', 'Webbed neck', 'Lymphedema']	Growth hormone therapy, Estrogen replacement therapy, Cardiac and renal evaluations
4	Cryptorchidism	['Absence of undescended testicle(s)', 'Empty scrotum', 'Smaller or underdeveloped testicle(s)', 'Inguinal hernia', 'Abnormal positioning of the testicle(s) (higher in the groin area)']	Close maintenance monitoring (masses of mild or transient cryptorchidism), hormone therapy (to stimulate testicular descent), surgical intervention (orchiopexy) to reposition the testicle(s) into the scrotum, if necessary, performed around the
5	Ethylene glycol poisoning-1	['Nausea', 'vomiting', 'abdominal pain', 'General malaise', 'weakness', 'increased thirst', 'frequent urination']	Supportive measures, Gastric Decontamination, Antidote Administration, Hemodialysis
6	Ethylene glycol poisoning-2	['Metabolic acidosis', 'rapid breathing', 'rapid heart rate', 'confusion', 'headache', 'dizziness', 'seizures']	Blood tests, Supportive measures, Gastric Decontamination, Antidote Administration, Hemodialysis
7	Ethylene glycol poisoning-3	['Decreased urine output', 'swelling in the legs and ankles', 'and signs of fluid overload', 'Severe kidney damage']	Supportive measures, Gastric Decontamination, Antidote Administration, Hemodialysis
8	Atrophic vaginitis	['Vaginal dryness', 'Vaginal burning', 'frequent urination', 'urinary tract infections', 'Painful intercourse']	Vaginal moisturizers, Vaginal estrogen therapy, Lifestyle modifications
9	Fracture	['Pain', 'Swelling', 'Bruising', 'Deformity', 'Difficulty moving', 'Loss of function']	Immobilization, Surgery, Rehabilitation
10	Cellulitis	['Redness', 'Pain', 'Tenderness', 'Swelling', 'Skin changes', 'Lymph node enlargement']	Antibiotics, Warm compresses, Immobilization, Rest, Pain management
11	Eye alignment disorder	['Double vision', 'Eye fatigue', 'Poor depth perception', 'Head tilting']	Glasses, Patching therapy, Vision therapy, Botox injections, Surgical intervention
12	Headache after lumbar puncture	['Throbbing headache']	Epidural blood patch, Conservative measures
13	Pyloric stenosis	['Persistent hunger', 'Weight loss', 'Dehydration', 'Visible peristalsis', 'Palpable mass', 'Fussiness']	Ultrasound, Blood tests, Pyloromyotomy
14	Adenoid cystic carcinoma	['Lump', 'swelling', 'Facial numbness', 'Tingling']	Surgery, Radiation therapy
15	Fibromyxolipoma	['Painless lump or swelling', 'Facial changes']	Surgical excision
16	Warthin tumor	['Painless lump or swelling', 'Facial changes']	Surgical excision
17	Mucoepidermoid carcinoma	['Lump', 'swelling', 'Difficulty', 'pain when swallowing and speaking', 'Facial weakness', 'Facial paralysis']	Surgery, Radiation therapy
18	Acinic cellular carcinoma	['Lump', 'swelling', 'Facial changes']	Surgery, Radiation therapy
19	Mucocele	['Painless fluid-filled swelling in the oral cavity']	Observation, Surgical removal if necessary
20	Osteochondritis	['Joint pain', 'stiffness', 'swelling', 'reduced range of motion']	Physical therapy, pain medications, rest, surgery
21	Sialolithiasis	['Swelling', 'pain', 'tenderness', 'difficulty eating']	Warm compresses, Saliva stimulation, Sialogogues, Stone removal
22	Submandibular stones	['Swelling', 'pain', 'dry mouth', 'bad taste']	Hydration, Saliva stimulation, Stone removal
23	Parotid stones	['Swelling', 'pain', 'dry mouth', 'bad taste']	Hydration, Saliva stimulation, Stone removal
24	Recurrent salivary stones	['Frequent episodes of stone formation']	Hydration, Saliva stimulation, Stone removal
25	Wharton's duct stones	['Swelling', 'pain', 'dry mouth', 'bad taste']	Hydration, Saliva stimulation, Stone removal
26	Mumps	['Swollen salivary glands (usually parotid)', 'fever', 'headache', 'fatigue']	Supportive care, pain relievers, isolation, vaccination (MMR)
27	Ludwig's angina	['Severe swelling of the floor of the mouth', 'fever', 'difficulty swallowing', 'drooling']	Intravenous antibiotics, airway management, surgical drainage
28	Mucocele	['Painless swelling', 'usually on the lower lip or floor of the mouth']	Observation, surgical removal if necessary
29	Abscess	['Swelling', 'pain', 'redness', 'fever', 'pus formation']	Antibiotics, surgical drainage
30	Sjögren's syndrome	['Dry mouth and eyes', 'fatigue', 'joint pain', 'dry skin']	Symptomatic relief with artificial tears, saliva substitutes, frequent sips of water, medications to stimulate saliva production, immune-modulating drugs

Figure 4.2 Dataset

The dataset was preprocessed by splitting the symptoms into a list, normalizing them, and encoding them into a binary format for model training.

4.4 System Workflow

1. **Data Input**: The user enters symptoms through the input fields in the UI.
2. **Prediction**: Upon clicking the "Predict Diagnosis" button, the system encodes the entered symptoms and feeds them into the decision tree models.
3. **Result Display**: The system displays the predicted diseases in the UI based on the classifier outputs.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

HealthTrack'D successfully combines machine learning and an intuitive user interface to create an efficient diagnostic tool for predicting diseases based on symptoms. By leveraging the **Decision Tree** algorithm, the system provides accurate predictions through a user-friendly platform built with Tkinter. The inclusion of multiple classifiers enhances the reliability of the predictions, making it a more reliable solution for preliminary medical diagnosis. This project highlights the effectiveness of integrating data-driven approaches with interactive design to address real-world challenges. Future developments could focus on expanding the dataset, improving model accuracy, and adding more features to make **HealthTrack'D** even more comprehensive and dependable.

5.2 Future Work

The **HealthTrack'D** system has significant potential for improvement and expansion. The following enhancements are proposed:

- **More Data**: Increase the dataset size by including more diseases and symptoms to improve prediction accuracy.
- **Mobile App**: Develop a mobile application version to make the system more accessible for users on smartphones.
- **For Patients**: Design the system for use at home, enabling individuals to better understand their symptoms without visiting a doctor.
- **Health Records**: Integrate the system with patient medical records to provide personalized and more accurate suggestions.

APPENDIX

HealthTrack'D.py

```
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import pandas as pd
import tkinter as tk
from tkinter import messagebox
from tkinter import *
from PIL import Image, ImageTk

# Load and preprocess data
data = pd.read_csv('Diseases_Symptoms.csv')
data = data.to_numpy()

# Preprocess symptoms
symptoms_rawdata = data[:, 2]
symptoms = []

for symptoms_entry in symptoms_rawdata:
    symptoms.extend([s.strip().lower() for s in symptoms_entry.split(',')])

# Ensure symptoms are unique and sorted
symptoms = sorted(set(symptoms))

# Prepare symptom matrix for each disease
symptom_matrix = []
diagnosis = []
```

```
for i in range(len(data)):
    current_symptoms = data[i, 2].split(',')
    current_symptoms = [s.strip().lower() for s in current_symptoms]

    # Encode the current symptoms using the list of all possible symptoms
    current_symptom_encoded = np.zeros(len(symptoms))
    for s in current_symptoms:
        if s in symptoms:
            idx = symptoms.index(s)
            current_symptom_encoded[idx] = 1

    symptom_matrix.append(current_symptom_encoded)
    diagnosis.append(f"{data[i, 1]}: {data[i, 3]}")

symptom_matrix = np.array(symptom_matrix)

# Train the classifier
classifier = DecisionTreeClassifier(random_state=0)
classifier.fit(symptom_matrix, diagnosis)

classifier1 = DecisionTreeClassifier(random_state=1)
classifier1.fit(symptom_matrix, diagnosis)

classifier2 = DecisionTreeClassifier(random_state=2)
classifier2.fit(symptom_matrix, diagnosis)
# Tkinter User Interface setup
m = tk.Tk()
m.attributes("-fullscreen", True)
```

```
bg_image = Image.open("Bg-Image.png")
bg_image = bg_image.resize((m.winfo_screenwidth(), m.winfo_screenheight()),
Image.Resampling.LANCZOS)
bg_photo = ImageTk.PhotoImage(bg_image)

# Set the background image using a Label widget
bg_label = tk.Label(m, image=bg_photo)
bg_label.place(relwidth=1, relheight=1)

def quit_button():
   m.destroy()

Heading = tk.Label(m, text="HealthTrack'D", font=("Arial", 48))
Heading.pack()



# Create symptom input fields
symptom_label = tk.Label(m, text="Enter Symptom 1:", font=("Arial", 14))
symptom_label.pack(pady=5)
symptom_entry = tk.Entry(m, font=("Arial", 14))
symptom_entry.pack(pady=10)

symptom2_label = tk.Label(m, text="Enter Symptom 2:", font=("Arial", 14))
symptom2_label.pack(pady=5)
symptom2_entry = tk.Entry(m, font=("Arial", 14))
symptom2_entry.pack(pady=10)

predict_1 = tk.Label(m, text= 'Prediction 1')
```

```
predict_1.pack()
predict_2 = tk.Label(m, text= 'Prediction 2')
predict_2.pack()
predict_3 = tk.Label(m, text= 'Prediction 3')
predict_3.pack()



# Function to get user input and predict diagnosis
def user_input():
  symptom1 = "\'" + symptom_entry.get().lower().strip() + "\'"
  symptom2 = "\'" + symptom2_entry.get().lower().strip() + "\'"

  # Check if symptoms are valid
  if symptom1 in symptoms and symptom2 in symptoms:
    # Encode symptoms into the symptom matrix
    user_input_encoded = np.zeros(len(symptoms))
    user_input_encoded[symptoms.index(symptom1)] = 1
    user_input_encoded[symptoms.index(symptom2)] = 1

    # Predict the diagnosis
    prediction = classifier.predict([user_input_encoded])
    prediction1 = classifier1.predict([user_input_encoded])
    prediction2 = classifier2.predict([user_input_encoded])
    predict_1.config(text=prediction[0])
    predict_2.config(text=prediction1[0])
    predict_3.config(text=prediction2[0])
  else:
```

```
    messagebox.showerror("Symptom Error!", "One or both symptoms are not recognized.")

# Button to trigger prediction
predict_button = tk.Button(m, text="Predict Diagnosis", font=("Arial", 14),
command=user_input)
predict_button.pack(pady=20)


button = tk.Button(m, text="Quit", font=("Arial", 14), command=quit_button)
button.pack()


m.mainloop()
quit
```