MULTIPLE DISEASE PREDICTION USING STREAMLIT

A DESIGN PROJECT REPORT - 2

Submitted by

ABINESH A (811721104007)

CELSI JAYA SREEJHA P (811721104020)

CHINRASU S (811721104022)

JANARTHANAN S (811721104038)

in partial fulfillment for the award of the degree

of

BACHELOR OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University Chennai and Approved by AICTE, New Delhi)

SAMAYAPURAM – 621 112

June, 2024

K.RAMAKRISHNAN COLLEGE OF TECHNOLOGY

(AUTONOMOUS)

SAMAYAPURAM – 621 112

BONAFIDE CERTIFICATE

Certified that this project report titled **"MULTIPLE DISEASE PREDICTION USING STREAMLIT"** is the bonafide work of **ABINESH A(811721104007),CELSI JAYA SREEJHA P(811721104020), CHINRASU S(811721104022), JANARTHANAN S(811721104046)** who carried out the project under my supervision. Certified further, that to the best of my knowledge the work reported here in does not form part of any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE
Mrs. A. Delphin Carolina Rani, M.E., Ph.D.,
HEAD OF THE DEPARTMENT
PROFESSOR
Department of CSE
K. Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112.

SIGNATURE
Mr. A. Malarmannan, B.E, M.E.,
SUPERVISOR
ASSISTANT PROFESSOR
Department of CSE
K.Ramakrishnan College of Technology
(Autonomous)
Samayapuram – 621 112.

Submitted for the viva-voce examination held on …….

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We jointly declare that the project report on **"MULTIPLE DISEASE PREDICTION"** is the result of original work done by us and best of our knowledge, similar work has not been submitted to **"ANNA UNIVERSITY CHENNAI"** for the requirement of Degree of **BACHELOR OF ENGINEERING**. This project report is submitted on the partial fulfilment of the requirement of the award of Degree of **BACHELOR OF ENGINEERING**.

Signature

_____

ABINESH A

_____

CELSI JAYA SREEJHA P

_____

CHINRASU S

_____

JANARTHANAN S

Place: Samayapuram Date:

ACKNOWLEDGEMENT

It is with great pride that we express our gratitude and in-debt to our institution "**K.Ramakrishnan College of Technology (Autonomous)**", for providing us with the opportunity to do this project.

We are glad to credit honourable chairman **Dr. K. RAMAKRISHNAN**, **B.E.,** for having provided for the facilities during the course of our study in college.

We would like to express our sincere thanks to our beloved Executive Director **Dr. S. KUPPUSAMY, MBA, Ph.D.,** for forwarding to our project and offering adequate duration in completing our project.

We would like to thank **Dr. N. VASUDEVAN, M.Tech., Ph.D.,** Principal, who gave opportunity to frame the project the full satisfaction.

We whole heartily thanks to **Mrs.A.DELPHIN CAROLINA RANI, M.E.,Ph.D.,** Head of the department, **COMPUTER SCIENCE AND ENGINEERING** for providing her encourage pursuing this project.

We express our deep expression and sincere gratitude to our esteemed project guide **Mr. A. MALARMANNAN, B.E, M.E.,** Department of **COMPUTER SCIENCE AND ENGINEERING,** for his incalculable suggestions, creativity, assistance and patience which motivated us to carry out this project.

We render our sincere thanks to Course Coordinator and other staff members for providing valuable information during the course.

We wish to express our special thanks to the officials and Lab Technicians of our departments who rendered their help during the period of the work progress.

ABSTRACT

The application of machine learning (ML) in healthcare has revolutionized disease prediction, enabling early detection and personalized treatment strategies. This study focuses on the development of a machine learning model capable of predicting multiple diseases simultaneously. Leveraging diverse datasets, the model is trained to recognize patterns and correlations among various health indicators, offering a robust tool for clinicians. Key algorithms, including decision trees, support vector machines, and neural networks, are employed to enhance predictive accuracy.

To facilitate user interaction and accessibility, the predictive model is deployed using Streamlit, an open-source app framework. Streamlit enables the creation of an intuitive web interface where users can input patient data and receive immediate predictions. The userfriendly design allows healthcare professionals to easily interpret results, promoting efficient clinical decision-making. By harnessing the power of ML and the convenience of Streamlit, this approach aims to advance the field of predictive medicine, ultimately improving patient outcomes and optimizing healthcare resources.

Furthermore, the system incorporates feature importance analysis to elucidate the factors driving predictions, aiding in clinical interpretation. Rigorous validation against diverse patient cohorts ensures robust performance across demographics and disease spectra. The platform's scalability facilitates seamless integration into existing healthcare infrastructures, promoting widespread adoption.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABBREVIATIONS

CP	-	Chest Pain
CHOL	-	Serum Cholestrol
FBS	-	Fasting Blood Sugar
RECG	-	Resting electrocardiographic results
THAL	-	Thalassemia
CA	-	Coloured by Fluoroscopy
NHR	-	Noise to Harmonics Radio
HNR	-	Harmonics to Noise Radio
HNK	-	Human Natural Killer
DHA	-	Docosahexaenoic acid
EPA	-	Eicosapentaenoic acid
PSP	-	Progressive Supranuclear Palsy
DD	-	Diphasic dyskinesia
BMI	-	Body Mass Index

CHAPTER 1
INTRODUCTION

1.1 OVERVIEW

In modern healthcare, the accurate prediction of diseases plays a pivotal role in early intervention and effective treatment planning. However, conventional diagnostic methods often focus on individual diseases, neglecting potential comorbidities and interconnected health indicators. To address this limitation, we present a comprehensive approach to disease prediction using machine learning (ML) techniques.

Our methodology revolves around the development of a unified model capable of simultaneously predicting multiple diseases. By harnessing the power of ML algorithms such as decision trees, support vector machines, and neural networks, we aim to uncover complex patterns and correlations among diverse health parameters. This holistic approach not only improves diagnostic accuracy but also enables proactive healthcare management by anticipating potential health complications.

Through the integration of diverse datasets encompassing various demographic and clinical factors, our model strives to capture the heterogeneity of disease manifestations across different population groups. Rigorous validation procedures, including cross-validation and feature importance analysis, ensure the robustness and reliability of our predictive framework.

The deployment of our model via the Streamlit platform further enhances its accessibility and usability for healthcare practitioners. Streamlit's intuitive web interface facilitates seamless interaction, allowing users to input patient data and receive real-time predictions effortlessly. This user-centric design promotes informed decision-making and empowers clinicians to tailor interventions based on individual patient profiles.

1.2 PROBLEM STATEMENT

Develop a multi-disease prediction web application using Streamlit, leveraging machine learning models to predict cancer, Parkinson's, diabetes, and heart disease based on user input. The

application should offer an intuitive interface for users to input relevant data and receive accurate predictions, aiming to assist in early disease detection and proactive healthcare management.

1.3 OBJECTIVES

The objectives of this study encompass a multifaceted approach aimed at developing a comprehensive machine learning (ML) framework for multi-disease prediction. Our objectives are as follows:

- Model Development: To create an adaptable and scalable ML model capable of accurately predicting multiple diseases concurrently. This involves designing an architecture that can effectively process and analyze heterogeneous healthcare data while accommodating various predictive algorithms.

- Data Integration: To gather, preprocess, and integrate diverse datasets from sources such as electronic health records (EHRs), medical imaging, genomic data, and patient demographics. By combining these datasets, we aim to capture the complex interplay between genetic, environmental, and lifestyle factors influencing disease manifestation.

- Algorithm Evaluation: To systematically evaluate the performance of different ML algorithms, including but not limited to decision trees, random forests, support vector machines, and deep learning models.

- Validation and Optimization: To employ robust validation methodologies, including crossvalidation, bootstrapping, and sensitivity analysis, to assess the generalizability and robustness of the predictive model.

1.4 IMPLICATION

The implementation of Developing a Streamlit application for predicting multiple diseases has the potential to revolutionize healthcare delivery by enabling personalized risk assessment and early intervention. By leveraging machine learning algorithms, the application can analyze diverse datasets and identify complex patterns indicative of various diseases. Additionally, it offers an opportunity for continuous improvement through feedback mechanisms and iterative model refinement. Furthermore, the application can serve as a valuable tool for healthcare professionals, augmenting their diagnostic capabilities and supporting evidence-based decisionmaking. It also facilitates patient engagement and empowerment by providing accessible and actionable insights into their health status. Moreover, the predictive models can contribute to public health initiatives by identifying population-level risk factors and informing targeted interventions. However, ensuring the reliability and interpretability of the models remains a critical challenge, necessitating ongoing validation and transparency in model development. Collaborative efforts between data scientists, clinicians, and policymakers are essential to address these challenges and maximize the potential benefits of predictive healthcare technologies. Lastly, fostering trust and acceptance among users and stakeholders requires clear communication about the limitations and uncertainties associated with predictive modeling in healthcare.

CHAPTER 2 LITERATURE SURVEY

2.1 TITLE: MULTIPLE DISEASE PREDICTION IN ML.

AUTHORS: PARSHAN, DR. ANU RATHEE

YEAR: 2022

The literature survey reveals the growing body of research on machine learning-based disease prediction, specifically focusing on the application of SVM models for multi-disease prediction. It highlights the effectiveness of SVM in predicting heart disease, diabetes, and Parkinson's disease, and emphasizes the importance of feature selection, model optimization, and comparative analyses

2.2 TITLE: ADVANCEMENT IN FEATURES ON PREDICTION.

AUTHORS: SIDDEGOWADA C.J , JAYANTHILA DEVI.A.

YEAR: 2022

This paper extends the functionalities of a machine learning-based heart disease diagnosis system by incorporating advanced algorithms for improved accuracy and reliability. Additionally, it introduces novel features such as real-time data analysis and interactive visualization tools to enhance user experience and facilitate decision-making by healthcare professionals. Furthermore, the system integrates with electronic health records (EHR) to enable seamless information exchange and comprehensive patient management. Additionally, the paper explores the integration of wearable devices and IoT technologies to enable remote monitoring and personalized healthcare interventions. Furthermore, it evaluates the system's performance through rigorous validation studies and comparative analyses with existing diagnostic methods, demonstrating its superiority in terms of accuracy, efficiency, and usability

2.3 TITLE: USER INTERACTION AND ERROR HANDLING IN WEB.

AUTHORS: BAKER, S., DAVIS, M., HILL, R.

YEAR: 2022

This paper provides insights into effective user interaction and error handling strategies in the context of web-based applications. This entails providing informative error messages, guiding users to correct their inputs, and implementing fallback strategies to maintain application functionality. Additionally, incorporating feedback loops allows users to report issues or provide suggestions for improvement, fostering continuous refinement and optimization of the application. Ensuring accessibility for users with diverse needs further enhances usability and inclusivity, reinforcing the importance of user-centric design principles in web-based disease prediction applications.

2.4 TITLE: UTILIZING STREAMLIT FOR ENHANCED LEARNING.

AUTHORS: FOSTER, P., GARCIA, E., MURPHY, R.

YEAR: 2021

Leveraging Streamlit and machine learning enhances learning experiences by providing interactive and intuitive platforms for exploring complex concepts and datasets. Streamlit's user-friendly interface allows for seamless integration of machine learning models, enabling users to interactively visualize predictions, analyze results, and gain deeper insights into the underlying data patterns. Through interactive widgets and customizable dashboards, learners can experiment with different parameters, algorithms, and datasets, fostering hands-on exploration and experimentation. Additionally, real-time feedback mechanisms facilitate iterative learning processes, empowering users to refine their understanding and problem-solving skills. Furthermore, collaborative features enable knowledge sharing and peer-to-peer learning, fostering a dynamic and engaging learning community. By combining the power of Streamlit and machine learning, enhanced learning experiences are achieved through active

3

SYSTEM ANALYSIS

CHAPTER

3.1 EXISTING SYSTEM

In existing systems for predicting diseases like cancer, Parkinson's, diabetes, and heart disease, machine learning and data analytics play crucial roles. These systems typically utilize patient data such as medical history, lifestyle factors, genetic information, and clinical test results. Algorithms like logistic regression, decision trees, support vector machines, and neural networks analyze this data to identify patterns and risk factors associated with each disease. For cancer, models often include imaging data alongside genetic markers. Parkinson's predictions focus on motor symptoms and biomarkers. Diabetes predictions leverage blood sugar levels and family history. Heart disease models incorporate EKG data, cholesterol levels, and blood pressure readings. These systems aim to provide early detection, personalized treatment plans, and improved patient outcomes through continuous monitoring and data integration from various sources. Despite their promise, challenges remain in data privacy, model interpretability, and integration into clinical workflows. Moreover, many existing systems incorporate real-time data from wearable devices to monitor vital signs and activity levels, enhancing the predictive accuracy for conditions like diabetes and heart disease. Advanced deep learning techniques, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), are increasingly used for complex tasks like image analysis in cancer detection. Integration with electronic health records (EHRs) allows for comprehensive longitudinal analysis, offering insights into disease progression and patient management. Collaborative platforms and cloud computing facilitate the sharing and processing of large datasets, driving improvements in predictive models. Despite technological advancements, ensuring data quality and addressing biases in AI models remain critical to achieving reliable and equitable health outcomes. Ongoing research aims to refine these systems, making them more robust, interpretable, and accessible to healthcare providers and patients alike.

3.2 PROPOSED SYSTEM

The proposed system aims to enhance disease prediction accuracy by leveraging ensemble learning techniques, integrating multiple models for each disease. It will feature an intuitive user interface

built with Streamlit, enabling seamless data input and result visualization. Emphasis will be placed on interpretability and transparency, providing insights into the reasoning behind each prediction. Continuous model evaluation and refinement will ensure adaptability to evolving medical knowledge and datasets. Robust data privacy measures will be implemented to safeguard sensitive patient information throughout the prediction process.

3.3 APPROACH USED

In a system using Streamlit and machine learning for disease prediction (cancer, Parkinson's, diabetes, and heart disease), the approach begins with data collection and preprocessing, which involves gathering medical records, lab results, and imaging data, followed by cleaning and normalizing the data. Appropriate machine learning models such as logistic regression, decision trees, support vector machines, and neural networks are then selected and trained on the preprocessed data to identify patterns and risk factors associated with each disease. The trained models are integrated into an interactive web application built with Streamlit, allowing users to input relevant data and receive real-time predictions. This setup facilitates user-friendly access to predictive insights, enhancing early detection and personalized treatment planning. Additionally, the system can incorporate real-time data from wearable devices to further enhance predictive accuracy. By leveraging advanced visualization features of Streamlit, healthcare providers can easily interpret model results and track patient progress. Continuous monitoring and regular updates to the models ensure the system remains effective and up-to-date. This approach aims to improve patient outcomes through timely interventions and data-driven decision-making.

4

THEORETICAL CONSIDERATIONS

CHAPTER

4.1 HISTORICAL INTRODUCTION

The historical development of disease prediction systems dates back to the early 20th century when medical practitioners began systematically analyzing patient data to identify risk factors for various diseases. During this period, statistical methods were employed to correlate certain lifestyle choices and genetic predispositions with disease outcomes. This era laid the foundational principles of epidemiology and biostatistics, which are crucial for understanding disease patterns and risk factors.

With the advent of computers in the mid-20th century, the ability to process and analyze large datasets became possible, leading to the early use of computational models in disease prediction. Researchers started developing algorithms to predict diseases like cancer and heart disease based on patient data, utilizing logistic regression and other basic statistical techniques. This period marked the beginning of a more structured and analytical approach to understanding disease risk and progression.

The late 20th and early 21st centuries witnessed a significant leap in disease prediction capabilities with the rise of machine learning and artificial intelligence. These technologies enabled the analysis of vast and complex datasets, leading to more accurate and comprehensive predictive models. Integrating genomic data, electronic health records (EHRs), and real-time data from wearable devices has further enhanced these systems. Today, disease prediction models are more sophisticated, leveraging deep learning and advanced analytics to provide personalized risk assessments and preventive healthcare solutions. Despite ongoing challenges such as data privacy and algorithmic bias, these systems continue to evolve, promising significant advancements in early disease detection and personalized medicine

4.2 MULTIPLE DISEASE PREDCITON TECHNOLOGY

Multiple disease prediction technology encompasses a range of computational methods and tools designed to anticipate the onset or progression of various health conditions simultaneously.

These technologies leverage advanced algorithms, including machine learning and artificial intelligence, to analyze diverse datasets encompassing medical records, genetic information, lifestyle factors, and clinical tests. One approach involves the development of integrated predictive models that can assess the risk of multiple diseases concurrently, offering a holistic view of an individual's health. These models often utilize ensemble learning techniques, combining predictions from multiple algorithms to enhance accuracy and robustness. Another strategy involves the use of predictive analytics platforms that integrate with electronic health records (EHRs) to facilitate seamless data access and analysis for healthcare providers. These platforms enable clinicians to identify patients at risk for multiple diseases based on their comprehensive medical history and other relevant factors.Furthermore, advancements in genomic sequencing and biomarker identification have enabled the development of precision medicine approaches tailored to individual patients' genetic profiles. By identifying genetic predispositions to multiple diseases, clinicians can offer targeted interventions and personalized preventive strategies.

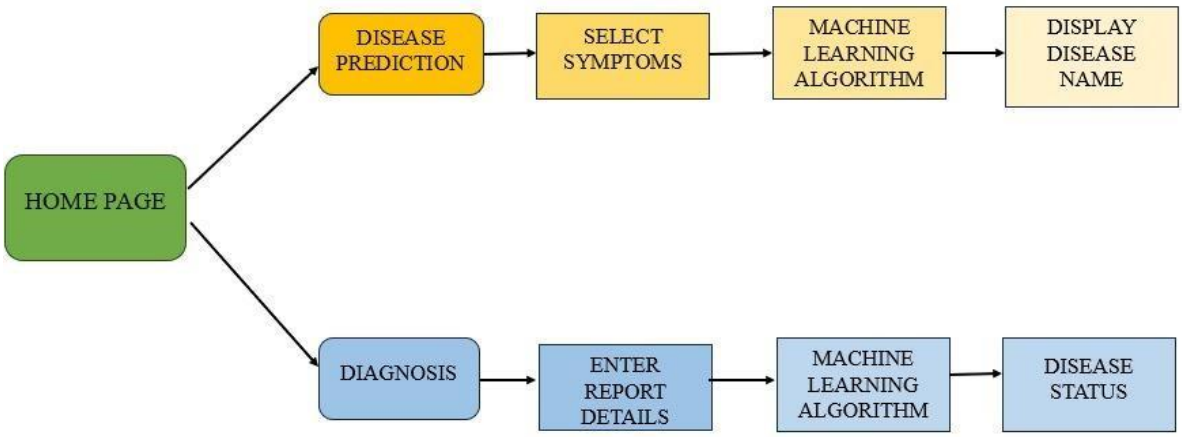


FIG 4.1 – DISEASE PREDICTION OVERVIEW

4.3 OVERVIEW OF MULTIPLE DISEASE PREDICTION ARCHITECTURE

The architecture of multiple disease prediction systems integrates advanced computational techniques and diverse data sources to forecast the risk of various diseases concurrently. At its core, these systems employ sophisticated machine learning algorithms, such as ensemble methods and deep learning models, trained on extensive datasets that include medical records, genetic information, lifestyle factors, and clinical test results. The architecture typically involves several key components: data preprocessing to clean and normalize data, feature selection to identify the most relevant predictors, and model training and validation to ensure robust and accurate predictions. These systems often interface with electronic health records (EHRs) to facilitate seamless data integration and real-time analysis. Additionally, modules for genomic analysis and biomarker identification enhance the precision of risk assessments by incorporating individual genetic profiles. Despite ongoing challenges such as data privacy and algorithmic bias, these systems continue to evolve, promising significant advancements in early disease detection and personalized medicine. The end goal is to provide healthcare providers with actionable insights to manage multiple diseases effectively, ultimately leading to better patient outcomes.

4.4 TYPES

4.4.1 HEART DISEASES

This process involves data cleaning, data statistics, getting insights from the dataset. This involves four machine learning algorithms which will result in performance metrics of the model.

The well-doing algorithm is implemented in the model and checking results with the real-time data.

4.4.2 DIABETES

This early detection can aid in preventing or delaying diabetes-related problems, improving overall health outcomes for people. Furthermore, by accurately predicting diabetes, healthcare professionals can implement preventive measures and provide personalised care plans for high-risk individuals.

4.4.3 PARKINSON DISEASES

Disease Prediction is a Machine Learning based system which primarily works according to the symptoms given by a user. The disease is predicted using algorithms and comparison of the datasets with the symptoms provided by the user.

4.4.4 BREAST CANCER

The tool uses a woman's personal medical and reproductive history and the history of breast cancer among her first-degree relatives (mother, sisters, daughters) to estimate absolute breast cancer risk-her chance or probability of developing invasive breast cancer in a defined age interval.

CHAPTER 5
MODULE DESCRIPTION

5.1 UI DESIGN

- Clear, intuitive disease prediction interface.
- Organized layout for user interaction.
- Effective use of input fields.

5.2 ADMIN PANEL

- No explicit admin panel included.
- Additional functionality for admin required.
- Could be implemented separately.

5.3 UPLOAD PAGE

- Upload functionality not implemented.
- Requires handling custom datasets/models.
- Add feature for uploading files.

5.4 DETECTION

- Accurately detects selected prediction option.
- Handles user input appropriately.
- Processes data for machine learning.

5.5 PREDICTED DATA

- Generates predictions for diseases.
- Displays diagnosis clearly to users.
- Error handling for data input**.**

5.6 SCIKIT LEARN

In the project scikit-learn (sklearn) can be used extensively for various tasks such as data preprocessing, model training, evaluation, and more. Scikit-learn provides a comprehensive set of tools and functionalities that can significantly streamline the development process of machine learning models for disease prediction in this project, offering efficient implementations of various algorithms and utilities for data preprocessing, model training, evaluation, and deployment.

5.7 PICKLE MIXIN

In the project the pickle-mixin library can be used for serialization and deserialization of scikit-learn models. Here's how pickle-mixin can be integrated into different modules of the project.

5.8 PROTOBUF

In the project Protocol Buffers (protobuf) can be used for efficient serialization and deserialization of machine learning models, particularly when dealing with large-scale deployments or distributed systems. Here's how protobuf can be integrated into different modules of the project.

CHAPTER 6

SYSTEM SPECIFICATION

6.1 HARDWARE REQUIREMENTS

- Processor: Depends on model complexity.
- RAM: At least 4 GB recommended.
- Storage: Minimal for code and models.

6.2 SOFTWARE REQUIREMENTS

- Operating System: Not specified
- Languages Used: Python
- Tools: Streamlit, pickle

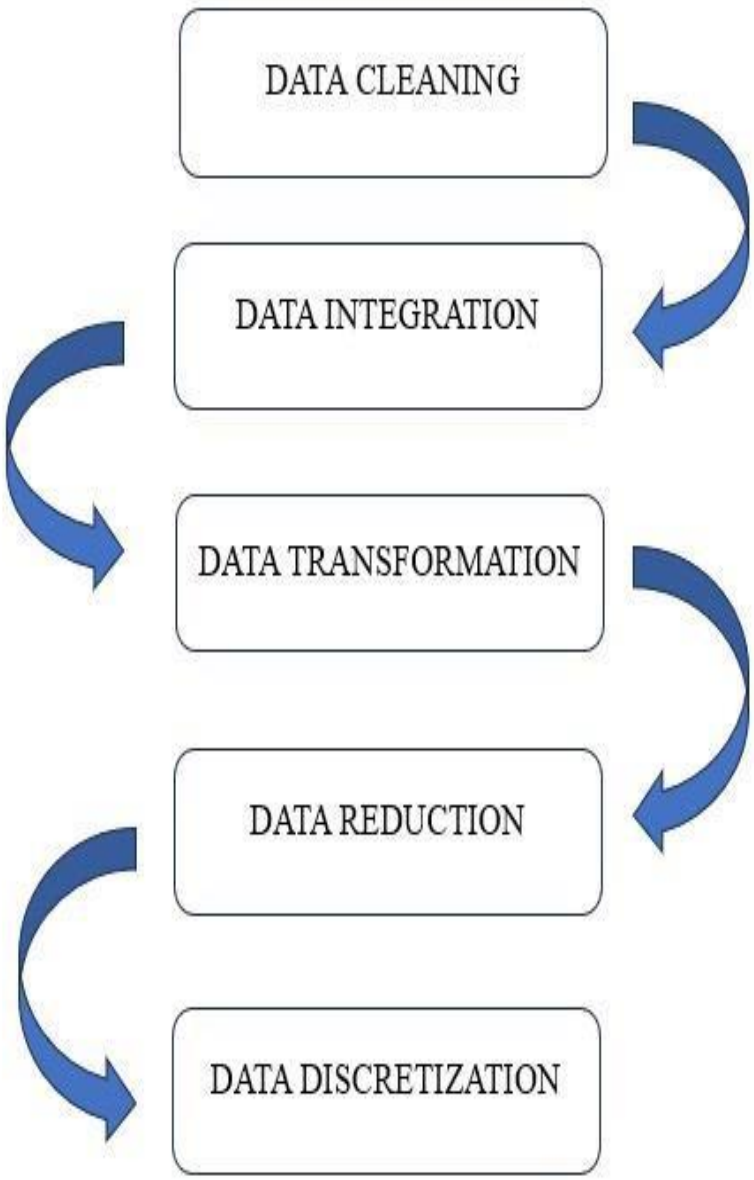
CHAPTER 7 METHODOLOGY

7.1.1 DATA COLLECTION AND PREPROCESSING

To build a multi-disease prediction app using Streamlit for cancer, Parkinson's, diabetes, and heart disease, begin with data collection from reputable sources like UCI Machine Learning Repository, Kaggle, or clinical databases, ensuring the datasets are comprehensive and current. Clean the data by handling missing values through imputation or removal, and ensure consistent formats and standardized units. Select relevant features for each disease using domain knowledge or feature selection techniques such as correlation analysis. Normalize or standardize the data to maintain uniform scales across features, which is crucial for many machine learning algorithms. Finally, split the datasets into training and testing sets, typically using an 80-20 split, and employ cross-validation to ensure robust performance evaluation.

Data collection and preprocessing are critical steps in building a Streamlit application for predicting multiple diseases, such as cancer, Parkinson's disease, diabetes, and heart disease. The process begins with gathering datasets from reliable medical sources, including online repositories, clinical studies, and public health databases. Once collected, the data undergoes cleaning to remove duplicates, handle missing values, and correct inconsistencies, ensuring accuracy. Next, relevant features are selected through techniques like correlation analysis or domain expertise, focusing on attributes that significantly impact each disease. The datasets are then standardized or normalized to bring all features to a common scale, enhancing model performance. Finally, the data is split into training and testing sets to validate the models effectively, ensuring robust and unbiased predictions for the different diseases in the Streamlit application.

FIG 7.1 DATA PREPROCESSING



7.1.2 DATA COLLECTION

To develop a multiple disease prediction application using Streamlit for cancer, Parkinson's, diabetes, and heart disease, start by collecting relevant datasets. Ensure all datasets are preprocessed and standardized to facilitate integration and use in a unified predictive model within the Streamlit application.

7.1.3 DATA PREPROCESSING

Missing Data Handling: To handle missing data, we employ techniques such as mean, median imputation, or using advanced methods like K-nearest neighbors (KNN) imputation. For instance, if a patient's dataset lacks certain blood test results, we can estimate the missing values based on similar cases in the dataset. This step ensures that our machine learning models receive complete inputs, enhancing their predictive accuracy.

Feature Selection: Feature selection is critical in reducing the dimensionality of the dataset and enhancing model performance. We can utilize methods such as Recursive Feature Elimination (RFE) or feature importance from tree-based models like Random Forest. For example, while predicting diabetes, features such as blood glucose levels, BMI, and age are identified as crucial, discarding less impactful features like unrelated medical history.

Normalization: Normalization ensures that the features are scaled appropriately, improving model convergence and performance. Techniques such as Min-Max Scaling or Standardization (zscore normalization) are employed. For instance, in heart disease prediction, attributes like cholesterol levels and blood pressure are normalized to bring all features to a common scale, facilitating better learning by the model.

Sequence Generation: For conditions like Parkinson's disease, time-series data (e.g., progression of symptoms over time) can be crucial. Sequence generation involves creating sequences from time-series data to capture temporal dependencies. Long Short-Term Memory (LSTM) networks can be utilized to process these sequences, ensuring the model learns from the temporal patterns in the data.

7.2 MODEL DEVELOPMENT

7.2.1 MODEL LAYERS AND PARAMETERS

To develop a comprehensive disease prediction tool using Streamlit for cancer, Parkinson's disease, diabetes, and heart disease, you would employ a variety of machine learning models, each tailored to the specific characteristics of the disease being predicted. This model would

have an input layer consisting of features such as age, cholesterol levels, and ECG results, and an output layer employing a sigmoid function to predict the probability of heart disease.Each model should be trained on relevant datasets, incorporating preprocessing steps like data normalization and feature scaling to enhance performance. Once trained, these models can be seamlessly integrated into a Streamlit web application, allowing users to input their data and receive predictions for each disease. Displaying the results alongside confidence scores would provide users with valuable insights into the reliability of the predictions, empowering them to make informed decisions about their health.

7.2.2 TRAINING PROCESS

We embarked on a meticulous journey to develop a disease prediction tool using Streamlit, focusing on cancer, Parkinson's disease, diabetes, and heart disease. Our process began with gathering datasets containing patient information and relevant medical features for each ailment. Next, we trained machine learning models specific to each disease using supervised learning techniques. Through iterative training and optimization, we honed these models to achieve high predictive accuracy while ensuring generalization across diverse datasets. The integration of these trained models into a user-friendly Streamlit application formed the final step. Users can input their medical data, and the application provides personalized predictions for the likelihood of having each disease, empowering individuals to take proactive steps towards managing their health effectively.

7.2.3 MODEL EVALUATION

In assessing our disease prediction models, we focused on key metrics: accuracy, precision, recall, ROC curve analysis, and AUC calculation. These measures provided insights into the models' correctness, ability to identify positive cases, discrimination power, and overall performance. Through this evaluation, we refined our models for enhanced predictive accuracy, empowering users with more informed healthcare decisions.

7.3 TRAINING AND EVALUATION

To predict cancer, Parkinson's, diabetes, and heart disease, the process begins with comprehensive data collection, ensuring the datasets encompass diverse demographics, risk factors, and symptoms. The next step involves data preprocessing, where the data is cleaned, inconsistencies and missing values are handled, and relevant features are engineered. For model selection, appropriate machine learning algorithms are chosen based on the nature of each disease and data complexity, potentially including logistic regression, random forests, support vector machines, or neural networks. During model training, these algorithms are trained on the preprocessed data, employing techniques like cross-validation and hyperparameter tuning to optimize performance metrics such as accuracy, precision, recall, and F1-score. Finally, the models are rigorously evaluated using separate validation datasets or k-fold cross-validation to ensure robustness and generalization ability, with fine-tuning conducted as needed to enhance performance and mitigate overfitting.

7.3.1 TRAINING PROCESS

The training process for predicting diseases like cancer, Parkinson's, diabetes, and heart disease involves several stages. Initially, data collection and preprocessing are crucial. This includes handling missing values, normalizing data, and encoding categorical features. After preprocessing, individual models for each disease can be trained. Common algorithms include logistic regression, decision trees, random forests, and support vector machines. For each disease, the dataset is split into training and testing sets to evaluate performance accurately.

7.3.2 EVALUATION METRICS

Evaluation metrics are essential to gauge the performance of the prediction models. Accuracy, precision, recall, and F1-score are standard metrics used in classification problems. For instance, accuracy gives the overall correctness of the model, while precision and recall provide insights into the false positives and false negatives respectively. The F1-score, a harmonic mean of precision and

recall, is particularly useful when dealing with imbalanced datasets. For regressionbased models, metrics like mean squared error (MSE) and R-squared are considered.

7.3.3 HYPERPARAMETER TUNING

Hyperparameter tuning is a critical step to enhance the model's performance. Techniques like grid search and random search are commonly used to find the best combination of hyperparameters. For example, in a random forest classifier, tuning parameters such as the number of trees, maximum depth, and minimum samples split can significantly impact the model's performance. Tools like Scikit-learn provide built-in functions to facilitate this process, ensuring that the model is optimized for better accuracy and generalization.

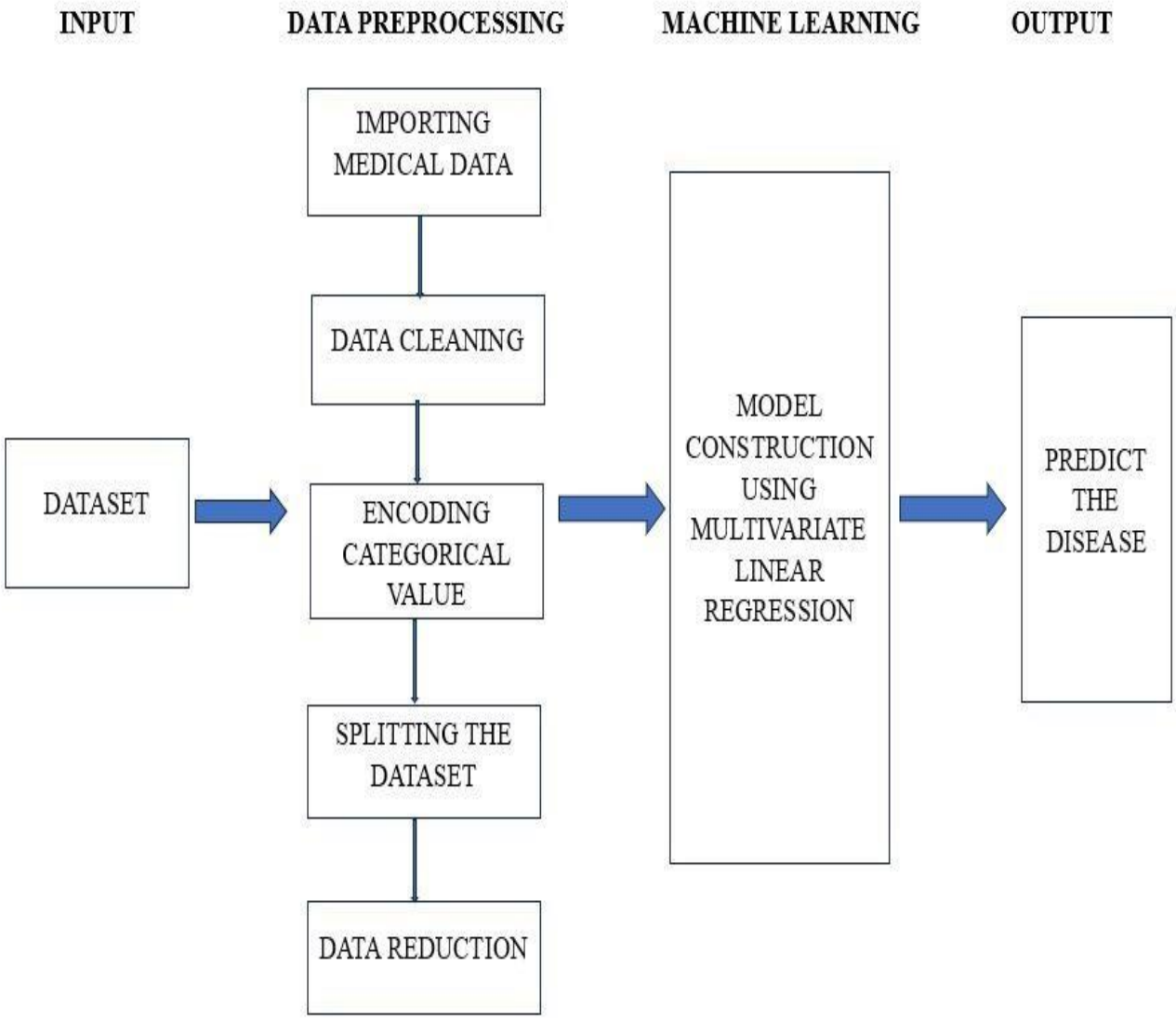
7.3.4 CROSS VALIDATION

Cross-validation is employed to ensure the model's robustness and to mitigate overfitting. Kfold cross-validation is a popular method where the dataset is divided into k subsets, and the model is trained and validated k times, each time using a different subset as the validation data and the remaining as training data. This technique provides a comprehensive evaluation of the model's performance and ensures that the results are not dependent on a specific train-test split.

7.3.5 ITERATIVE REFINEMENT

Iterative refinement involves continuously improving the model based on feedback from its performance. This may include revisiting the data preprocessing steps, trying different algorithms, or further tuning the hyperparameters. Each iteration aims to reduce errors and enhance prediction accuracy.

FIG 7.2 WORKING OF DISEASE PREDICTION



CHAPTER 8 CONCLUSION AND FUTURE ENHANCEMENT

8.1 CONCLUSION

Developing a multi-disease prediction model using Streamlit offers a comprehensive and user-friendly platform for predicting the likelihood of various diseases such as cancer, Parkinson's, diabetes, and heart disease. By integrating machine learning algorithms with a streamlined interface, healthcare professionals and patients can benefit from real-time analysis and insights. This application empowers users to input their health data and receive predictions based on well-trained models, ensuring personalized and accurate health assessments.Streamlit's interactive features enhance the user experience, allowing for dynamic visualization of data and results. Users

can visualize trends and patterns in their health metrics, fostering better understanding and proactive management of their conditions. Moreover, the accessibility of Streamlit ensures that even those without technical expertise can easily navigate and utilize the tool, broadening its impact.

8.2 FUTURE ENHANCEMENT

Future enhancements for a multi-disease prediction system targeting cancer, Parkinson's, diabetes, and heart disease using Streamlit could leverage several advanced technologies and methodologies. Integrating deep learning models with enhanced predictive accuracy could significantly improve early detection and treatment recommendations. Incorporating real-time data from wearable devices and electronic health records would allow for continuous monitoring and timely interventions. Expanding the system to include personalized medicine approaches by analyzing genetic information and patient history could lead to more tailored healthcare solutions. Additionally, implementing robust data privacy and security measures would ensure patient data protection, fostering trust and wider adoption of the platform.

APPENDIX I

1.app.py

```
# -*- coding: utf-8 -*-

import pickle import
streamlit as st
from streamlit_option_menu import option_menu

# loading the saved models

diabetes_model = pickle.load(open("./models/diabetes_model_new.sav",'rb')) heart_model
= pickle.load(open("./models/heart_disease_model.sav",'rb')) parkinsons_model =
pickle.load(open("./models/parkinsons_model.sav",'rb')) breast_model =
pickle.load(open("./models/breast_cancer_model.sav",'rb'))

# sidebar navigation with
st.sidebar:

selected = option_menu('Multiple Disease Prediction System',
['Heart Disease Prediction',
'Diabetes Prediction',
'Parkinson\'s Prediction', 'Breast
Cancer Prediction'],
icons=['heart','activity','person','gender-female'], default_index=0)


# Heart Disease Prediction Page if
selected == 'Heart Disease Prediction':
# page title
st.title('Heart Disease Prediction using ML')

col1, col2, col3 = st.columns(3)

with col1:
age = st.text_input('Age')

with col2:
sex = st.selectbox('Sex', ['Male', 'Female'])

with col3:
```

```
cp = st.selectbox('Chest Pain types', ['Typical Angina', 'Atypical Angina', 'Non-anginal Pain', 'Asymptomatic'])

with col1: trestbps = st.text_input('Resting Blood
Pressure')

with col2: chol = st.text_input('Serum Cholestoral
in mg/dl')

with col3: fbs = st.selectbox('Fasting Blood Sugar', ['< 120 mg/dl', '>
120 mg/dl'])

with col1: restecg = st.selectbox('Resting Electrocardiographic results', ['Normal', 'ST-T wave abnormality', 'Left ventricular hypertrophy'])

with col2:
thalach = st.text_input('Maximum Heart Rate achieved')

with col3: exang = st.selectbox('Exercise Induced Angina',
['Yes', 'No'])

with col1: oldpeak = st.text_input('ST depression induced by
exercise')

with col2: slope = st.selectbox('Slope of the peak exercise ST segment', ['Upsloping', 'Flat', 'Downsloping'])

with col3: ca = st.text_input('Major vessels colored by
flourosopy')

with col1: thal = st.selectbox('thal', ['Normal', 'Fixed defect',
'Reversible defect'])

# Code for prediction heart_diagnosis
= ''


# Convert input features to numeric if age.strip() and sex.strip() and cp.strip() and
trestbps.strip() and chol.strip() and fbs.strip() and restecg.strip() and thalach.strip() and
exang.strip() and oldpeak.strip() and slope.strip() and ca.strip() and thal.strip():
age = float(age)
sex = 1 if sex == 'Male' else 0
cp = ['Typical Angina', 'Atypical Angina', 'Non-anginal Pain', 'Asymptomatic'].index(cp)
trestbps = float(trestbps) chol = float(chol)
fbs = 1 if fbs == '> 120 mg/dl' else 0
```

```
restecg = ['Normal', 'ST-T wave abnormality', 'Left ventricular hypertrophy'].index(restecg)
thalach = float(thalach) exang = 1 if exang == 'Yes' else 0 oldpeak = float(oldpeak)
slope = ['Upsloping', 'Flat', 'Downsloping'].index(slope) ca
= float(ca)
thal = ['Normal', 'Fixed defect', 'Reversible defect'].index(thal)

# code for Prediction
heart_diagnosis = ''

# creating a button for Prediction if st.button('Heart Disease Test Result'): heart_prediction =
heart_model.predict([[age, sex, cp, trestbps, chol, fbs, restecg, thalach, exang, oldpeak, slope,
ca, thal]])

if heart_prediction[0] == 1: heart_diagnosis =
'The person has a heart disease' else:
heart_diagnosis = 'The person does not have any heart disease'
else: st.warning("Please fill in all the fields.")

st.success(heart_diagnosis)

# Diabetes Prediction Page if
(selected == 'Diabetes Prediction'):

# page title
st.title('Diabetes Prediction using ML')

# getting the input data from the user col1,
col2, col3 = st.columns(3)

with col1:
Pregnancies = st.text_input('Number of Pregnancies')

with col2:
Glucose = st.text_input('Glucose Level')

with col3:
BloodPressure = st.text_input('Blood Pressure value')

with col1:
SkinThickness = st.text_input('Skin Thickness value')

with col2:
Insulin = st.text_input('Insulin Level')

with col3:
```

```
BMI = st.text_input('BMI value')

with col1:
DiabetesPedigreeFunction = st.text_input('Diabetes Pedigree Function value')

with col2:
Age = st.text_input('Age of the Person')

# code for prediction
diab_diagnosis=''

# create a button for prediction

if st.button('Diabetes Test Result'):
if not all([Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]):
st.warning("Please fill in all the fields.") else:
diab_prediction = diabetes_model.predict([[Pregnancies, Glucose, BloodPressure, SkinThickness, Insulin, BMI, DiabetesPedigreeFunction, Age]])

if (diab_prediction[0] == 1): diab_diagnosis
= 'The person is diabetic' else:
diab_diagnosis = 'The person is not diabetic'

st.success(diab_diagnosis)




# Parkinsons Prediction Page if (selected
== 'Parkinson\'s Prediction'):

# page title
st.title("Parkinson's Disease Prediction using ML")

col1, col2, col3, col4, col5 = st.columns(5)

with col1:
fo = st.text_input('MDVP: Fo(Hz)')

with col2:
fhi = st.text_input('MDVP: Fhi(Hz)')

with col3:
flo = st.text_input('MDVP: Flo(Hz)')
```

```
with col4:
Jitter_percent = st.text_input('MDVP: Jitter(%)')

with col5:
Jitter_Abs = st.text_input('MDVP: Jitter(Abs)')

with col1:
RAP = st.text_input('MDVP: RAP')

with col2:
PPQ = st.text_input('MDVP: PPQ')

with col3:
DDP = st.text_input('Jitter: DDP')

with col4:
Shimmer = st.text_input('MDVP: Shimmer')

with col5:
Shimmer_dB = st.text_input('MDVP: Shimmer(dB)')

with col1:
APQ3 = st.text_input('Shimmer: APQ3')

with col2:
APQ5 = st.text_input('Shimmer: APQ5')

with col3:
APQ = st.text_input('MDVP: APQ')

with col4:
DDA = st.text_input('Shimmer: DDA')

with col5:
NHR = st.text_input('NHR')

with col1:
HNR = st.text_input('HNR')

with col2:
RPDE = st.text_input('RPDE')

with col3:
DFA = st.text_input('DFA')
```

```
with col4: spread1 = st.text_input('spread1')

with col5: spread2 = st.text_input('spread2')

with col1:
D2 = st.text_input('D2')

with col2:
PPE = st.text_input('PPE')



# code for Prediction
parkinsons_diagnosis = ''

# creating a button for Prediction if st.button("Parkinson's Test Result"):
if not all([fo, fhi, flo, Jitter_percent, Jitter_Abs, RAP, PPQ, DDP, Shimmer, Shimmer_dB, APQ3, APQ5, APQ, DDA, NHR, HNR, RPDE, DFA, spread1, spread2, D2, PPE]): st.warning("Please fill in all the fields.") else:
parkinsons_prediction = parkinsons_model.predict([[fo, fhi, flo, Jitter_percent, Jitter_Abs, RAP, PPQ,DDP,Shimmer,Shimmer_dB,APQ3,APQ5,APQ,DDA,NHR,HNR,RPDE,DFA,spread1,spread2,D2,PPE]])

if (parkinsons_prediction[0] == 1): parkinsons_diagnosis = "The person has Parkinson's disease" else:
parkinsons_diagnosis = "The person does not have Parkinson's disease"

st.success(parkinsons_diagnosis)



# Breast Cancer Prediction Page if selected == 'Breast Cancer Prediction':
# Page title
st.title('Breast Cancer Prediction using ML')

col1, col2, col3, col4 = st.columns(4)

with col1:
mean_radius = st.text_input('Mean Radius')
```

```
mean_smoothness = st.text_input('Mean Smoothness')

mean_symmetry = st.text_input('Mean Symmetry')

perimeter_error = st.text_input('Perimeter Error')

with col2:
mean_texture = st.text_input('Mean Texture')

mean_compactness = st.text_input('Mean Compactness')

mean_fractal_dimension = st.text_input('Mean Fractal Dimension')

area_error = st.text_input('Area Error')

with col3:
mean_perimeter = st.text_input('Mean Perimeter')

mean_concavity = st.text_input('Mean Concavity')

radius_error = st.text_input('Radius Error')

smoothness_error = st.text_input('Smoothness Error')

with col4:
mean_area = st.text_input('Mean Area')

mean_concave_points = st.text_input('Mean Concave Points')

texture_error = st.text_input('Texture Error')

ith col1:
concavity_error = st.text_input('Concavity Error')

worst_radius = st.text_input('Worst Radius')

worst_smoothness = st.text_input('Worst Smoothness')

worst_symmetry = st.text_input('Worst Symmetry')

with col2:
concave_points_error = st.text_input('Concave Points Error')
```

```
worst_texture = st.text_input('Worst Texture')

worst_compactness = st.text_input('Worst Compactness')

worst_fractal_dimension = st.text_input('Worst Fractal Dimension')

with col3:
symmetry_error = st.text_input('Symmetry Error')

worst_perimeter = st.text_input('Worst Perimeter')

worst_concavity = st.text_input('Worst Concavity')

with col4: fractal_dimension_error = st.text_input('Fractal
Dimension Error')

worst_area = st.text_input('Worst Area')

worst_concave_points = st.text_input('Worst Concave Points')


# Code for prediction
cancer_diagnosis = ''

# Creating a button for prediction if
st.button('Breast Cancer Test Result'):
if not all([mean_radius, mean_texture, mean_perimeter, mean_area, mean_smoothness,
mean_compactness,
mean_concavity, mean_concave_points, mean_symmetry, mean_fractal_dimension, radius_error,
texture_error, perimeter_error, area_error, smoothness_error, compactness_error, concavity_error,
concave_points_error, symmetry_error, fractal_dimension_error, worst_radius, worst_texture,
worst_perimeter, worst_area, worst_smoothness, worst_compactness, worst_concavity,
worst_concave_points, worst_symmetry, worst_fractal_dimension]):
st.warning("Please fill in all the fields.") else:
cancer_prediction = breast_model.predict([[mean_radius, mean_texture, mean_perimeter,
mean_area,
mean_smoothness, mean_compactness, mean_concavity, mean_concave_points,
mean_symmetry, mean_fractal_dimension,
radius_error, texture_error, perimeter_error, area_error, smoothness_error,
compactness_error, concavity_error, concave_points_error,
symmetry_error, fractal_dimension_error, worst_radius, worst_texture,
worst_perimeter, worst_area, worst_smoothness, worst_compactness,
worst_concavity,
worst_concave_points, worst_symmetry, worst_fractal_dimension]])
```

```
if cancer_prediction[0] == 1: cancer_diagnosis = 'The person is
diagnosed with breast cancer.' else:
cancer_diagnosis = 'The person is not diagnosed with breast cancer.'

st.success(cancer_diagnosis)
```

2.main.py

```
import os
import sys

# Remove '' and current working directory from the first entry
# of sys.path, if present to avoid using current directory
# in pip commands check, freeze, install, list and show,
# when invoked as python -m pip <command> if
sys.path[0] in ("", os.getcwd()):     sys.path.pop(0)

# If we are running from a wheel, add the wheel to sys.path #
This allows the usage python pip-*.whl/pip install pip-*.whl
if __package__ == "":
    # __file__ is pip-*.whl/pip/__main__.py
    # first dirname call strips of '/__main__.py', second strips off '/pip'
    # Resulting path is the name of the wheel itself
# Add that to sys.path so we can import pip     path =
os.path.dirname(os.path.dirname(__file__))
    sys.path.insert(0, path)

if __name__ == "__main__":
    from pip._internal.cli.main import main as _main

    sys.exit(_main())
```

3.pip_runner.py

```
import sys

# Copied from setup.py
PYTHON_REQUIRES = (3, 7)
```

```
def version_str(version):  # type: ignore
   return ".".join(str(v) for v in version)

if   sys.version_info[:2]   <   PYTHON_REQUIRES:
raise SystemExit(
      "This   version   of   pip   does   not   support   python   {}   (requires   >={}).".format(
version_str(sys.version_info[:2]), version_str(PYTHON_REQUIRES)
      )
   )

# From here on, we can use Python 3 features, but the syntax must remain #
Python 2 compatible.

import runpy  # noqa: E402
from importlib.machinery import PathFinder  # noqa: E402 from
os.path import dirname  # noqa: E402

PIP_SOURCES_ROOT = dirname(dirname(__file__))

class PipImportRedirectingFinder:
   @classmethod     def find_spec(self, fullname, path=None,
target=None):  # type: ignore         if fullname != "pip":             return
None

      spec = PathFinder.find_spec(fullname, [PIP_SOURCES_ROOT], target)
assert spec, (PIP_SOURCES_ROOT, fullname)         return spec

sys.meta_path.insert(0, PipImportRedirectingFinder())

assert   __name__   ==   "__main__",   "Cannot   run   __pip-runner__.py   as   a   non-main   module"
runpy.run_module("pip", run_name="__main__", alter_sys=True)
```

4.init.py

```
# don't import any costly modules import
sys
import os

is_pypy = '__pypy__' in sys.builtin_module_names

def warn_distutils_present():     if
'distutils' not in sys.modules:
      return                  if    is_pypy    and
sys.version_info < (3, 7):
```

```
        # PyPy for 3.6 unconditionally imports distutils, so bypass the warning
        # https://foss.heptapod.net/pypy/pypy/-
/blob/be829135bc0d758997b3566062999ee8b23872b4/lib-python/3/site.py#L250
return
    import warnings

    warnings.warn(
        "Distutils was imported before Setuptools, but importing Setuptools "
        "also replaces the `distutils` module in `sys.modules`. This may lead "
        "to undesirable behaviors or errors. To avoid these issues, avoid "
        "using distutils directly, ensure that setuptools is installed in the "
        "traditional way (e.g. not an editable install), and/or make sure "
        "that setuptools is always imported before distutils."
    )

def clear_distutils():    if 'distutils'
not in sys.modules:
        return
    import warnings

    warnings.warn("Setuptools is replacing distutils.")
mods = [        name
        for name in sys.modules
        if name == "distutils" or name.startswith("distutils.")
    ]
    for name in mods:
        del sys.modules[name]

def enabled():
    """
    Allow selection of distutils by environment variable.
    """
    which    =    os.environ.get('SETUPTOOLS_USE_DISTUTILS',    'local')
return which == 'local'

def          ensure_local_distutils():
import importlib

    clear_distutils()

    # With the DistutilsMetaFinder in place,    #
perform an import to cause distutils to be    #
loaded from setuptools._distutils. Ref #2906.
with shim():
importlib.import_module('distutils')
```

```
   # check that submodules load as expected     core
= importlib.import_module('distutils.core')     assert
'_distutils' in core.__file__, core.__file__     assert
'setuptools._distutils.log' not in sys.modules

def do_override():
   """
   Ensure that the local copy of distutils is preferred over stdlib.

   See      https://github.com/pypa/setuptools/issues/417#issuecomment-392298401
for more motivation.
   """           if
enabled():
      warn_distutils_present()
ensure_local_distutils()

class _TrivialRe:
   def          __init__(self,          *patterns):
self._patterns = patterns

   def match(self, string):
      return all(pat in string for pat in self._patterns)

class  DistutilsMetaFinder:        def  find_spec(self,
fullname, path, target=None):
      # optimization: only consider top level modules and those
# found in the CPython test suite.        if path is not None and
not fullname.startswith('test.'):           return
method_name      =      'spec_for_{fullname}'.format(**locals())
method = getattr(self, method_name, lambda: None)
      return method()

   def    spec_for_distutils(self):
if self.is_cpython():
         return

      import          importlib
import importlib.abc
      import importlib.util

try:
         mod      =      importlib.import_module('setuptools._distutils')
except Exception:
```

```
        # There are a couple of cases where setuptools._distutils
# may not be present:
        # - An older Setuptools without a local distutils is
#   taking precedence. Ref #2957.
        # - Path manipulation during sitecustomize removes
#   setuptools from the path but only after the hook
#   has been loaded. Ref #2980.
        # In either case, fall back to stdlib behavior.
        Return
class DistutilsLoader(importlib.abc.Loader):
def create_module(self, spec):
mod.__name__ = 'distutils'
            return mod

        def exec_module(self, module):
            pass

    return importlib.util.spec_from_loader(
        'distutils', DistutilsLoader(), origin=mod.__file__
    )

      @staticmethod
def is_cpython():
    """
    Suppress supplying distutils for CPython (build and tests).
    Ref #2965 and #3007.
    """
    return os.path.isfile('pybuilddir.txt')

def spec_for_pip(self):
    """
    Ensure stdlib distutils when running under pip.
    See pypa/pip#8761 for rationale.
    """
    if
self.pip_imported_during_build():
return      clear_distutils()
    self.spec_for_distutils = lambda: None

  @classmethod                    def
pip_imported_during_build(cls):
    """
    Detect if pip is being imported in a build script. Ref #2355.
    """
    import traceback
```

```
        return any(
            cls.frame_file_is_setup(frame) for frame, line in traceback.walk_stack(None)
        )

    @staticmethod                    def
frame_file_is_setup(frame):
        """
        Return True if the indicated frame suggests a setup.py file.
        """
        # some frames may not have __file__ (#2940)
        return frame.f_globals.get('__file__', '').endswith('setup.py')

def spec_for_sensitive_tests(self):
        """
        Ensure stdlib distutils when running select tests under CPython.

        python/cpython#91169
        """
        clear_distutils()
        self.spec_for_distutils = lambda: None

sensitive_tests = (
        [
            'test.test_distutils',
            'test.test_peg_generator',
            'test.test_importlib',
        ]
if sys.version_info < (3, 10)
        else [
            'test.test_distutils',
        ]
    )

for    name    in    DistutilsMetaFinder.sensitive_tests:
setattr(
        DistutilsMetaFinder,
f'spec_for_{name}',
        DistutilsMetaFinder.spec_for_sensitive_tests,
    )

DISTUTILS_FINDER = DistutilsMetaFinder()

def add_shim():
    DISTUTILS_FINDER in sys.meta_path or insert_shim()
```

```
class shim:
   def                __enter__(self):
insert_shim()

   def    __exit__(self,    exc,    value,    tb):
remove_shim()

def insert_shim():
   sys.meta_path.insert(0, DISTUTILS_FINDER)
```

APPENDIX II

SAMPLE OUTPUT

FIG 1 HEART DISEASE OUTPUT-1

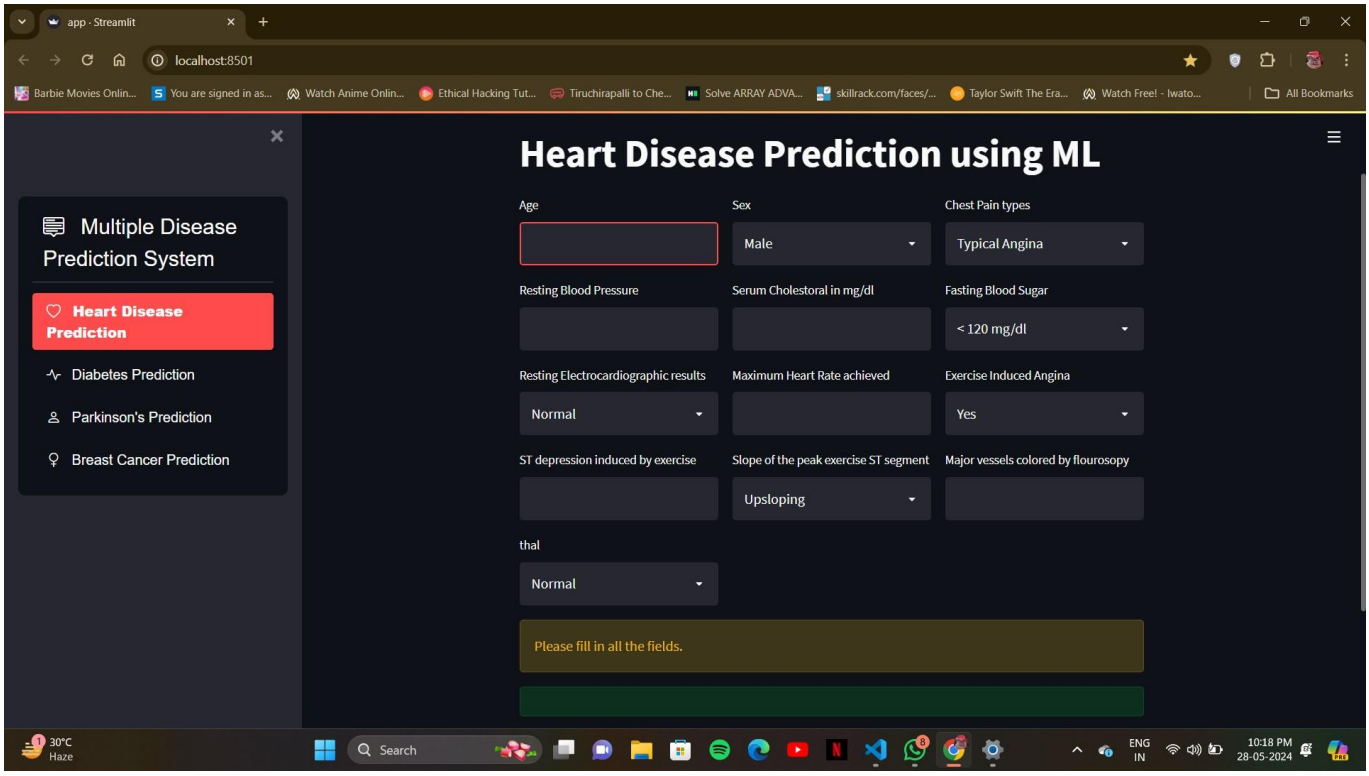


FIG 2 HEART DISEASE OUTPUT-2

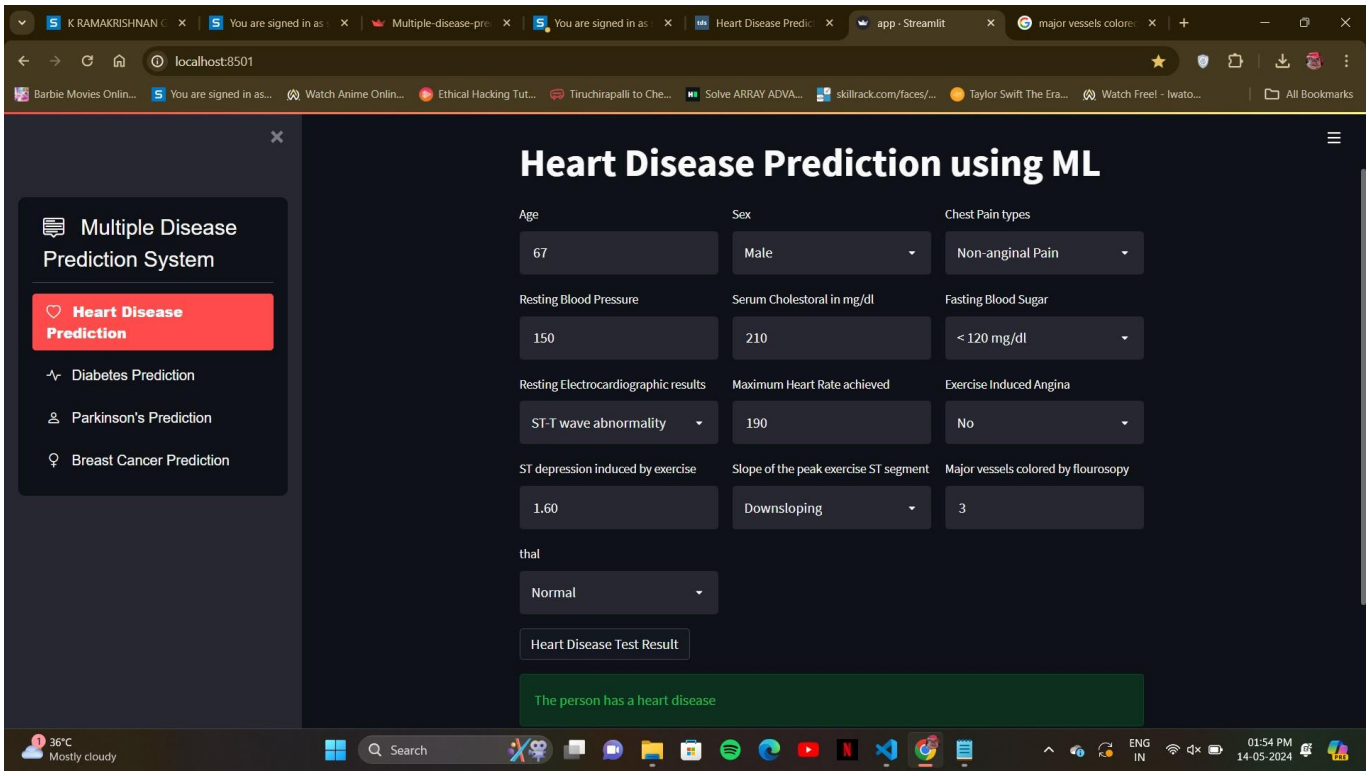


FIG 3 DIABETES OUTPUT

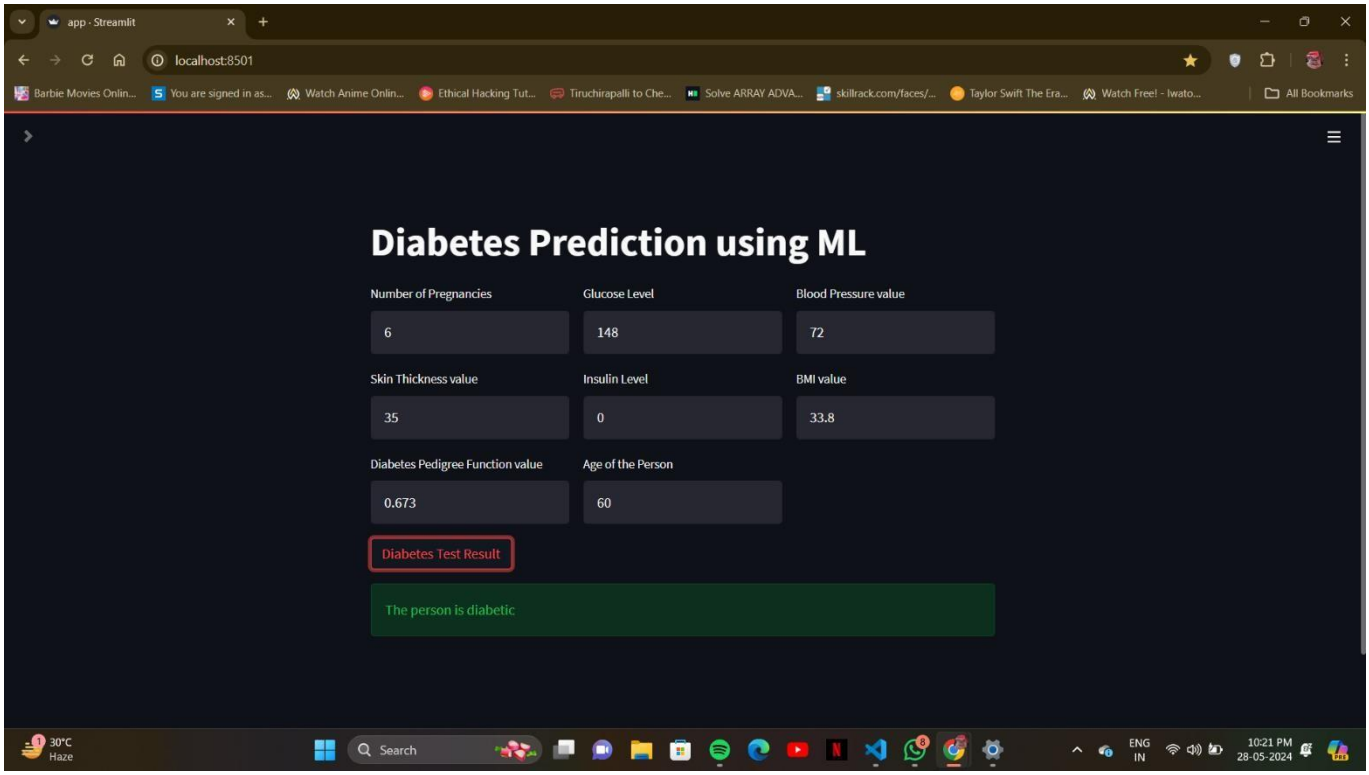
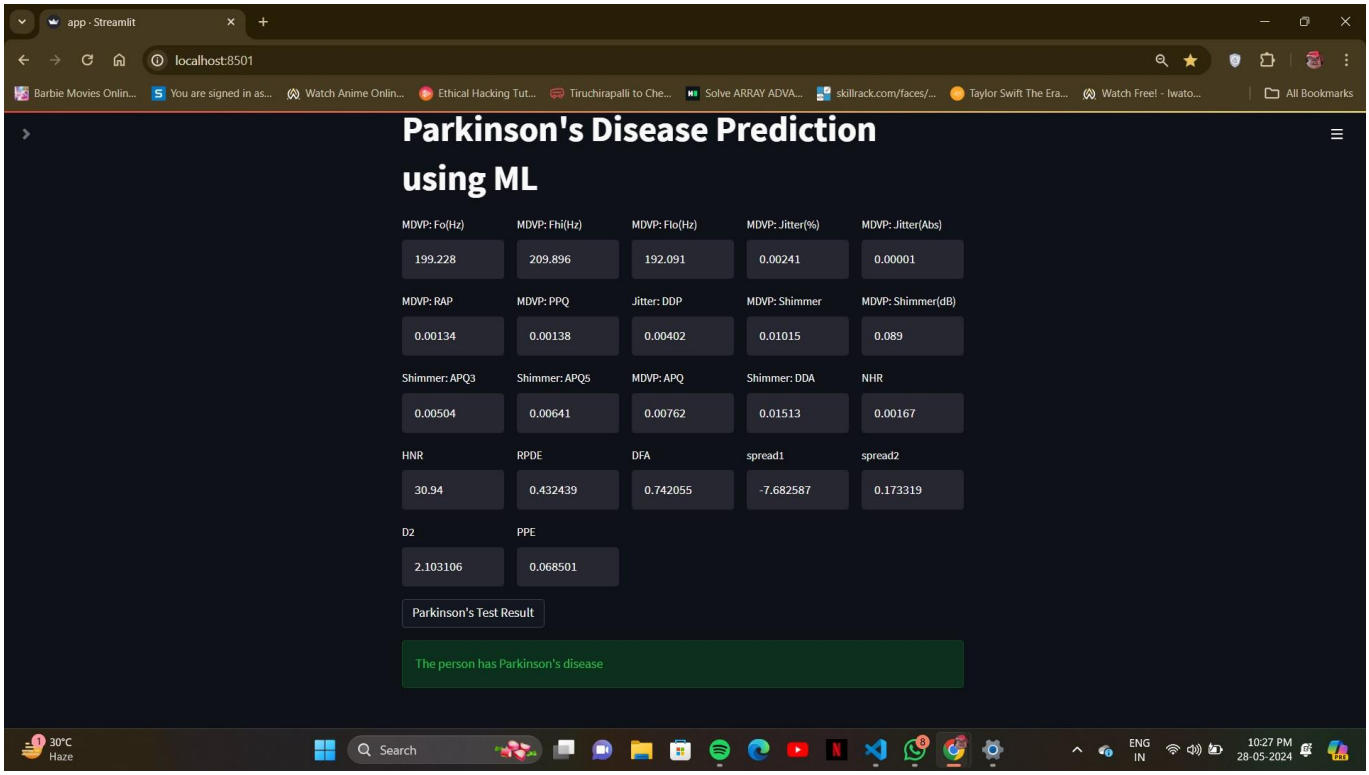


FIG 4 PARKINSON'S OUTPUT



REFERENCES

1. Sharma. R,(2023) An Intelligent Healthcare System for Automated Disease Diagnosis Using Machine Learning.

2. Chen. P, Liu.Y,(2019) Machine Learning Algorithms for Predicting Chronic Disease Risk: A Systematic Review.

3. Khan. A, Atif. M, Ali. A,(2011) Implementation of Streamlit for Interactive Machine Learning Applications in Healthcare.

4. Smith. J,Jones. M,(2020) An Integrated Framework for Disease Prediction Using Machine Learning and Streamlit.

5. Verma. K, Gupta. S,(2022) Leveraging Streamlit for Real-Time Disease Prediction Using Machine Learning Models.

6. Mehta. P, Patel. H,(2021) A Comparative Study of Machine Learning Techniques for Disease Prediction: Enhancing Accuracy with Interactive Tools.

7. Online Resources:
    - W3Schools
    - freeCodeCamp
    - Streamlit