DIAGNOBOT

**BACHELOR OF TECHNOLOGY**

## Computer Science And Engineering (2021-2025)



**CGC COLLEGE OF ENGINEERING, MOHALI**

SUBMITTED BY

Name: Manikpreet Singh Roll No. 2121359

Name: Nishant Sekhri Roll No. 2121390

.

**INDEX**

* Introduction……………………………………….………..1
* Project Overview…………………..……………………….2
* Tech Stack………………………………………………….3
* Implementation……………………………………………..4
* Use Cases…………………………………………………...5
* Conclusion…………………………………………………..6
* Source Code + Snapshots………………………………7-15
* Bilbography………………………………………………16

# Introduction

Diagnobot is a tech-savvy web app that uses smart algorithms to predict various diseases. It's like having a personal health assistant right on your screen. The goal of this project is to make it easy for anyone to check their risk for major health issues like heart disease, diabetes, Parkinson's, and breast cancer.

**Harnessing Technology for Health:**

Diagnobot is all about using the latest tech tools to help people stay healthy. Instead of waiting for problems to show up, Diagnobot helps catch them early by analyzing your health data. It's like having a digital health coach that keeps an eye on your well-being.

**Simplifying Disease Prediction:**

We've made Diagnobot super user-friendly. You just input your health info, and it does the rest. It's like having a chat with your doctor, but quicker and easier. Plus, Diagnobot keeps your data safe and private, so you can trust the results.

**Key Tech Features:**

Easy-to-use interface: Simple design that makes health checks a breeze.

Disease-specific tools: Specialized algorithms for each disease prediction.

Data security: Strong measures to protect your privacy and keep your info safe.

Continuous updates: We're always improving Diagnobot based on feedback and new tech advancements.

**Innovation in Healthcare:**

With Diagnobot, we're blending technology and healthcare to create a brighter, healthier future. It's about making health checks accessible and empowering individuals to take control of their well-being. This project report dives into how Diagnobot works, its impact, and how it's shaping the future of healthcare.

**Conclusion:**

Diagnobot is not just a tool; it's a tech-driven solution that's changing how we think about health. By leveraging technology, we're making disease prediction easier, faster, and more accurate. This project report explores the tech behind Diagnobot and how it's revolutionizing healthcare for everyone.

**PROJECT OVERVIEW**

The heart of the project revolves around crafting a user-friendly interface that welcomes users to input their medical data concerning various diseases. Once the user enters their data, the system employs advanced algorithms such as Naive Bayes, K-Nearest Neighbors (KNN), Random Forest, and Decision Trees – instead of pre-trained models – to analyze the data comprehensively. These algorithms are adept at learning from the input data and making predictions regarding the presence or absence of diseases with a high level of accuracy.

**Algorithmic Intelligence:**

Rather than relying on pre-packaged solutions, Diagnobot harnesses the power of algorithmic intelligence to process user data. Here's a glimpse into how these algorithms work:

* **Naive Bayes:** This algorithm calculates the probability of a patient having a disease based on the occurrence of symptoms. It's like having a probabilistic detective that connects symptoms to diseases.
* **K-Nearest Neighbors (KNN):** KNN looks at similar patient profiles in the dataset to predict disease outcomes. It's akin to consulting your neighbors to get an idea of what might be going on health-wise.
* **Random Forest:** This algorithm creates multiple decision trees to analyze data from different angles, providing a more robust prediction. Think of it as a forest of decision-making trees working together for accurate results.
* **Decision Trees:** Decision trees break down complex data into simple yes/no questions, making them easy to interpret and apply for disease prediction. They're like a flowchart of health-related questions leading to a diagnosis.

**Interactive Data Processing:**

The user-friendly interface of Diagnobot allows users to seamlessly input their medical data related to specific diseases. Each algorithm then processes this data independently but collectively contributes to the prediction process. This interactive approach not only enhances user experience but also ensures that the predictions are based on a comprehensive analysis of the input data.

**Enhanced Accuracy and Reliability:**

By leveraging these advanced algorithms, Diagnobot achieves a high level of accuracy and reliability in disease prediction. The system's ability to learn and adapt from the data ensures that predictions are continually refined and improved over time. This dynamic approach to data analysis sets Diagnobot apart as a robust and effective tool for healthcare prediction.

**Conclusion:**

In summary, Diagnobot's utilization of algorithms like Naive Bayes, KNN, Random Forest, and Decision Trees marks a significant leap forward in disease prediction technology. The project's emphasis on interactive data processing and algorithmic intelligence ensures accurate, reliable, and user-friendly predictions for individuals seeking to assess their health risks.

# TECH STACK

**Python:** Python is a versatile programming language known for its simplicity and readability. In the context of Diagnobot, Python serves multiple purposes:

* **Backend Development:** Python is used to develop the backend infrastructure of the web application. This includes handling user requests, data processing, and interfacing with the machine learning models.
* **Machine Learning Model Training:** Python's rich ecosystem of libraries such as NumPy, SciPy, and scikit-learn provides powerful tools for training machine learning models. Algorithms like Naive Bayes, KNN, Random Forest, and Decision Trees are implemented using Python.
* **Data Processing:** Python's libraries such as Pandas and NumPy are instrumental in data manipulation, cleaning, and preprocessing. These libraries enable efficient handling of large datasets and ensure that the input data is in the right format for model training and prediction.

**Streamlit:** Streamlit is a popular framework used for building interactive web applications in Python. In the context of Diagnobot, Streamlit plays a crucial role in creating the user interface that allows users to interact with the system. Some key features of Streamlit include:

* **Easy-to-use Interface:** Streamlit provides a straightforward syntax that developers can use to create interactive components such as sliders, text inputs, and buttons.
* **Real-time Updates:** Streamlit automatically updates the web interface in real-time as users input their data or make selections, providing a seamless user experience.
* **Data Visualization:** Streamlit integrates well with libraries like Matplotlib and Plotly, allowing developers to create interactive charts and graphs to visualize data and model predictions.

**Scikit-learn:** Scikit-learn is a powerful machine learning library in Python that provides a wide range of algorithms and tools for building predictive models. In the Diagnobot project, Scikit-learn is utilized for:

* **Machine Learning Model Development:** Scikit-learn offers implementations of various machine learning algorithms such as classification, regression, clustering, and dimensionality reduction. Algorithms like Logistic Regression, Support Vector Machines (SVM), and Random Forest Classifier are used for disease prediction in Diagnobot.
* **Model Evaluation:** Scikit-learn provides metrics and functions for evaluating the performance of machine learning models. This includes metrics like accuracy, precision, recall, F1-score, and area under the ROC curve (AUC-ROC).

**Pandas:** Pandas is a data manipulation and analysis library in Python that provides data structures like DataFrame and Series. In Diagnobot, Pandas is employed for:

* Data Preprocessing: Pandas is used to preprocess the input data before feeding it into the machine learning models. This includes tasks such as handling missing values, encoding categorical variables, and scaling numerical features.
* Data Exploration: Pandas facilitates data exploration and descriptive statistics, allowing developers to gain insights into the characteristics and distributions of the input data.

**Pickle:** Pickle is a serialization library in Python that is used for saving and loading Python objects, including machine learning models. In Diagnobot, Pickle is utilized for:

* **Model Persistence:** Trained machine learning models are serialized and saved using Pickle, allowing them to be stored as binary files. This ensures that the models can be easily loaded and reused without needing to retrain them each time the application is run.
* **Deployment:** Pickle also enables the deployment of machine learning models within the

**Implementation Details:**

Diagnobot is designed to cater to a wide range of health conditions through its various disease prediction modules. Let's delve into the implementation details of each module

**1. Heart Disease Prediction Module:**

* **Input Features:** Age, sex, chest pain type, blood pressure, cholesterol levels, fasting blood sugar, ECG results, heart rate, exercise-induced angina, ST depression, slope of the ST segment, number of major vessels, and thalassemia type.
* **Machine Learning Model:** Trained using historical data to classify individuals as having or not having heart disease based on input features.

2. **Diabetes Prediction Module:**

* **Input Features:** Number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, BMI, diabetes pedigree function, and age

.

* **Machine Learning Model:** Trained to predict whether a person is diabetic or not based on input features.

3. **Parkinson's Disease Prediction Module:**

* **Input Features:** Various voice-related parameters such as frequency, jitter, shimmer, noise-to-harmonics ratio, etc.
* **Machine Learning Model:** Analyzes voice features to determine the presence or absence of Parkinson's disease.

4**. Breast Cancer Prediction Module:**

* **Input Features:** Features extracted from mammogram images including mean radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, fractal dimension, and their errors.
* **Machine Learning Model:** Utilized to predict the likelihood of breast cancer based on mammogram data.

**5. Multi Symptom Disease Predictor Module:**

* **Input Features**: Symptoms such as fever, headache, nausea, vomiting, abdominal pain, fatigue, cough, sore throat, muscle aches, joint pain, skin rash, shortness of breath, and any other relevant symptoms based on the specific acute disease being predicted.
* **Machine Learning Model:** Developed using classification algorithms trained on a dataset containing symptoms and corresponding acute disease diagnoses.

**Implementation Approach:**

The implementation of each module follows a similar approach:

1. **Data Collection:** Gather relevant data for each disease module, including historical health records, symptom datasets, and medical images (for breast cancer).
2. **Data Preprocessing**: Clean, preprocess, and format the data to ensure consistency and compatibility with machine learning algorithms.
3. **Model Training:** Utilize machine learning algorithms such as KNN, Random Forest, Decision Trees, and others to train predictive models for each disease module.
4. **Feature Selection:** Identify and select key features from the input data that contribute most significantly to disease prediction accuracy.
5. **Model Evaluation:** Evaluate the trained models using metrics like accuracy, precision, recall, and F1-score to assess their performance and reliability.
6. **Integration with Diagnobot:** Integrate the trained models into the Diagnobot web application, providing users with a seamless and interactive platform for disease prediction and risk assessmen

**Benefits and Use Cases:**

* **Comprehensive Health Assessment:** Diagnobot offers a comprehensive assessment of various health conditions, ranging from chronic diseases like diabetes and heart disease to acute conditions.
* **Early Detection and Intervention:** The predictive models enable early detection of diseases, allowing for timely medical intervention and treatment planning.
* **User-Friendly Interface:** The web-based interface of Diagnobot makes it accessible to users from anywhere, promoting remote healthcare and telemedicine services.
* **Educational Resource:** Diagnobot serves as an educational resource for healthcare professionals, students, and individuals seeking to understand health risks and disease prediction.

**CONCLUSION**

In conclusion, Diagnobot represents a significant leap forward in the realm of predictive healthcare. By harnessing cutting-edge technologies such as Python for backend development, machine learning model training, and data processing, Streamlit for crafting the interactive web interface, Scikit-learn for developing robust machine learning models, Pandas for efficient data manipulation and preprocessing, and Pickle for model persistence, Diagnobot offers a comprehensive and user-centric platform for disease prediction and risk assessment.

The project's framework encompasses five distinct disease prediction modules: Heart Disease, Diabetes, Parkinson's Disease, Breast Cancer, and the newly added Multi-Symptom Disease Predictor. This expansion ensures that users receive personalized insights into a wide range of health conditions, including chronic ailments and diseases characterized by multiple symptoms.

The implementation strategy, which includes meticulous data collection, preprocessing, model training, feature selection, model evaluation, and seamless integration with the Diagnobot platform, guarantees the accuracy, reliability, and real-time updates crucial for effective disease prediction and risk management.

Diagnobot's myriad benefits span from early disease detection and timely medical intervention to facilitating remote healthcare accessibility and serving as an educational tool for healthcare professionals and individuals seeking to understand their health risks better.

With its intuitive interface, robust machine learning algorithms, continuous updates, and emphasis on personalized medicine, Diagnobot exemplifies the transformative power of technology in healthcare. It not only revolutionizes patient care by promoting proactive health management but also sets a new standard for how technology can enhance healthcare outcomes and improve overall well-being.

In essence, Diagnobot heralds a future where predictive analytics and personalized medicine converge to empower individuals, healthcare providers, and the healthcare system at large, ushering in an era of data-driven and patient-centric healthcare solutions.

**SOURCE CODE**

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.text\_input('Sex')

    with col3:

        cp = st.text\_input('Chest Pain types')

    with col1:

        trestbps = st.text\_input('Resting Blood Pressure')

    with col2:

        chol = st.text\_input('Serum Cholestoral in mg/dl')

    with col3:

        fbs = st.text\_input('Fasting Blood Sugar > 120 mg/dl')

    with col1:

        restecg = st.text\_input('Resting Electrocardiographic results')

    with col2:

        thalach = st.text\_input('Maximum Heart Rate achieved')

    with col3:

        exang = st.text\_input('Exercise Induced Angina')

    with col1:

        oldpeak = st.text\_input('ST depression induced by exercise')

    with col2:

        slope = st.text\_input('Slope of the peak exercise ST segment')

    with col3:

        ca = st.text\_input('Major vessels colored by flourosopy')

    with col1:

        thal = st.text\_input('thal: 1 = normal; 2 = fixed defect; 3 = reversible defect')

    # code for Prediction

    heart\_diagnosis = ''

    # creating a button for Prediction

    if st.button('Heart Disease Test Result'):

        if not all([age, sex, cp, trestbps, chol, fbs, restecg,thalach,exang,oldpeak,slope,ca,thal]):

            st.warning("Please fill in all the fields.")

        else:

            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'

    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')

        compactness\_error = st.text\_input('Compactness Error')

    with 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]])

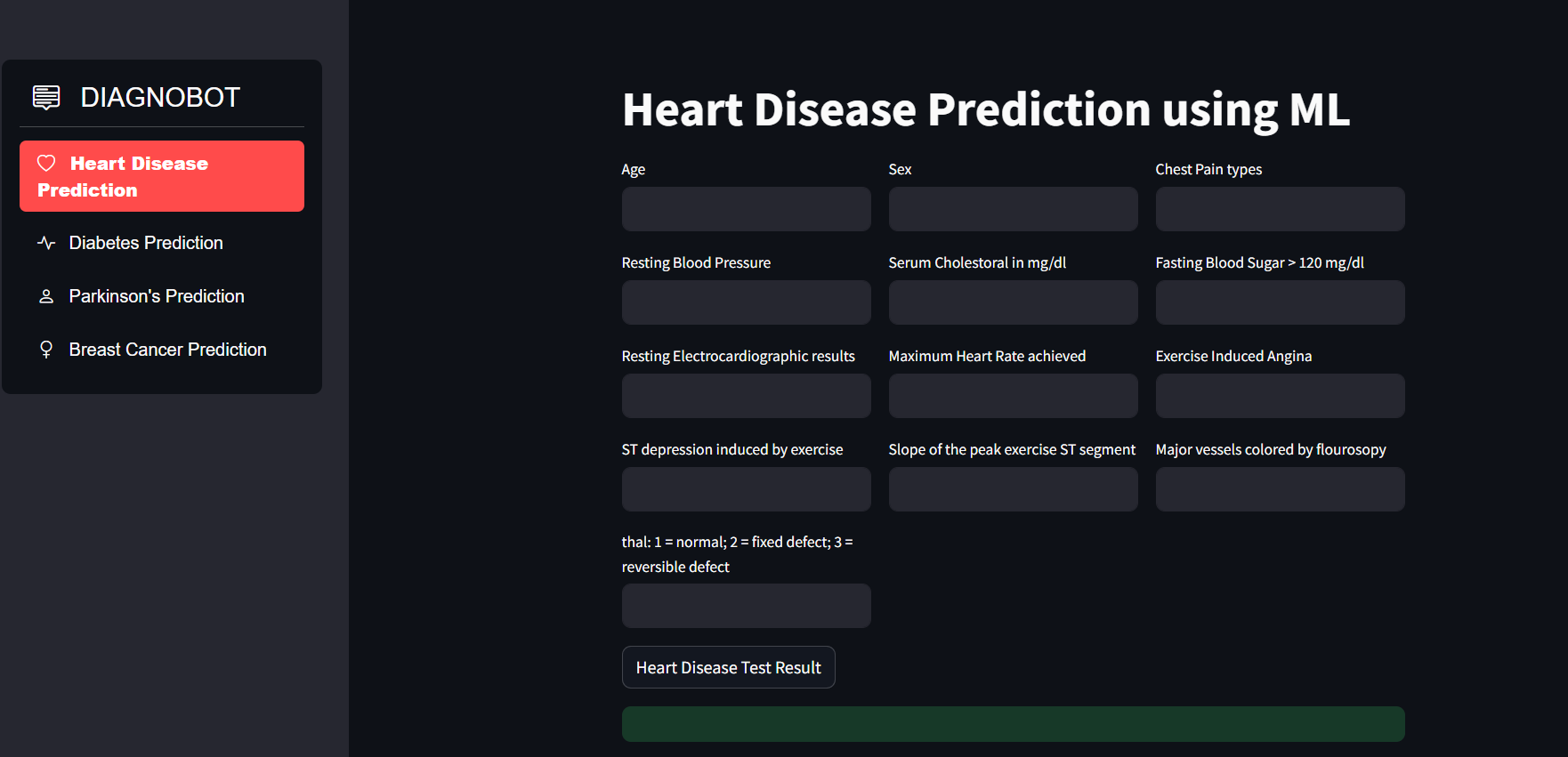
            if cancer\_prediction[0] == 1:

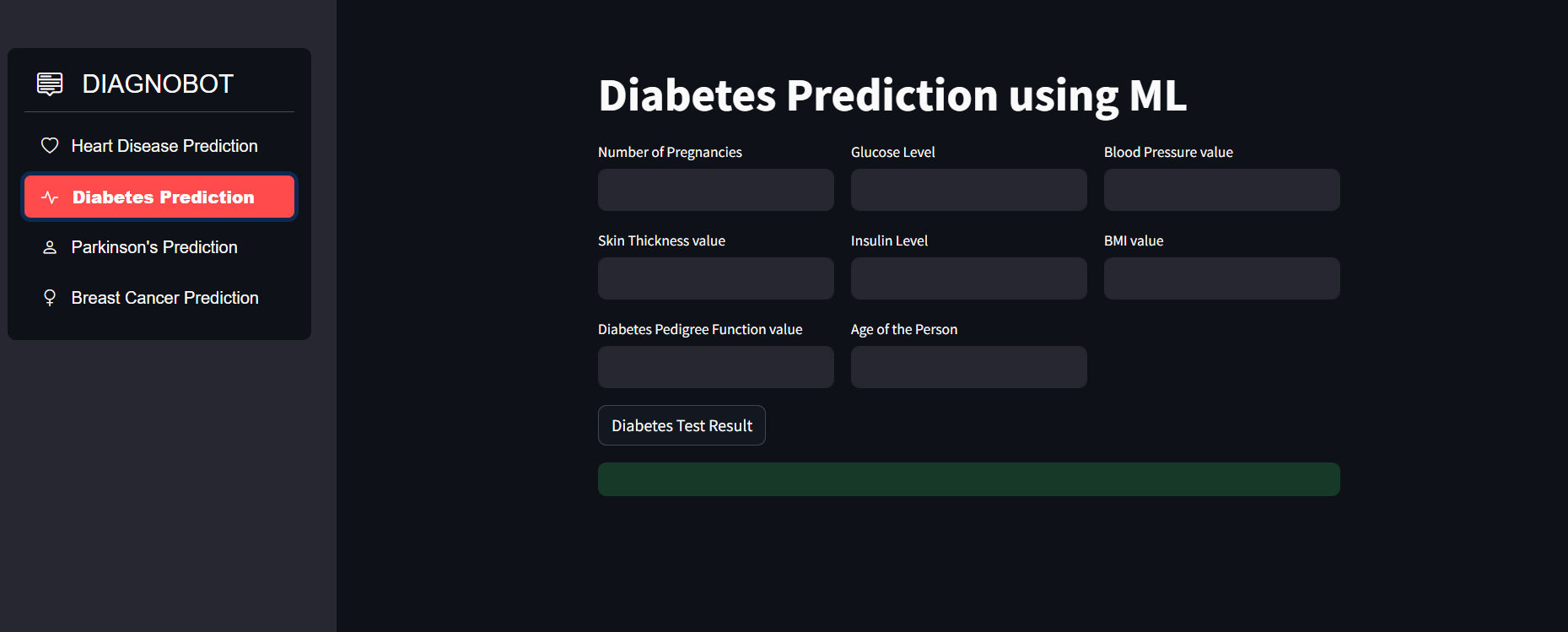
                cancer\_diagnosis = 'The person is diagnosed with breast cancer.'

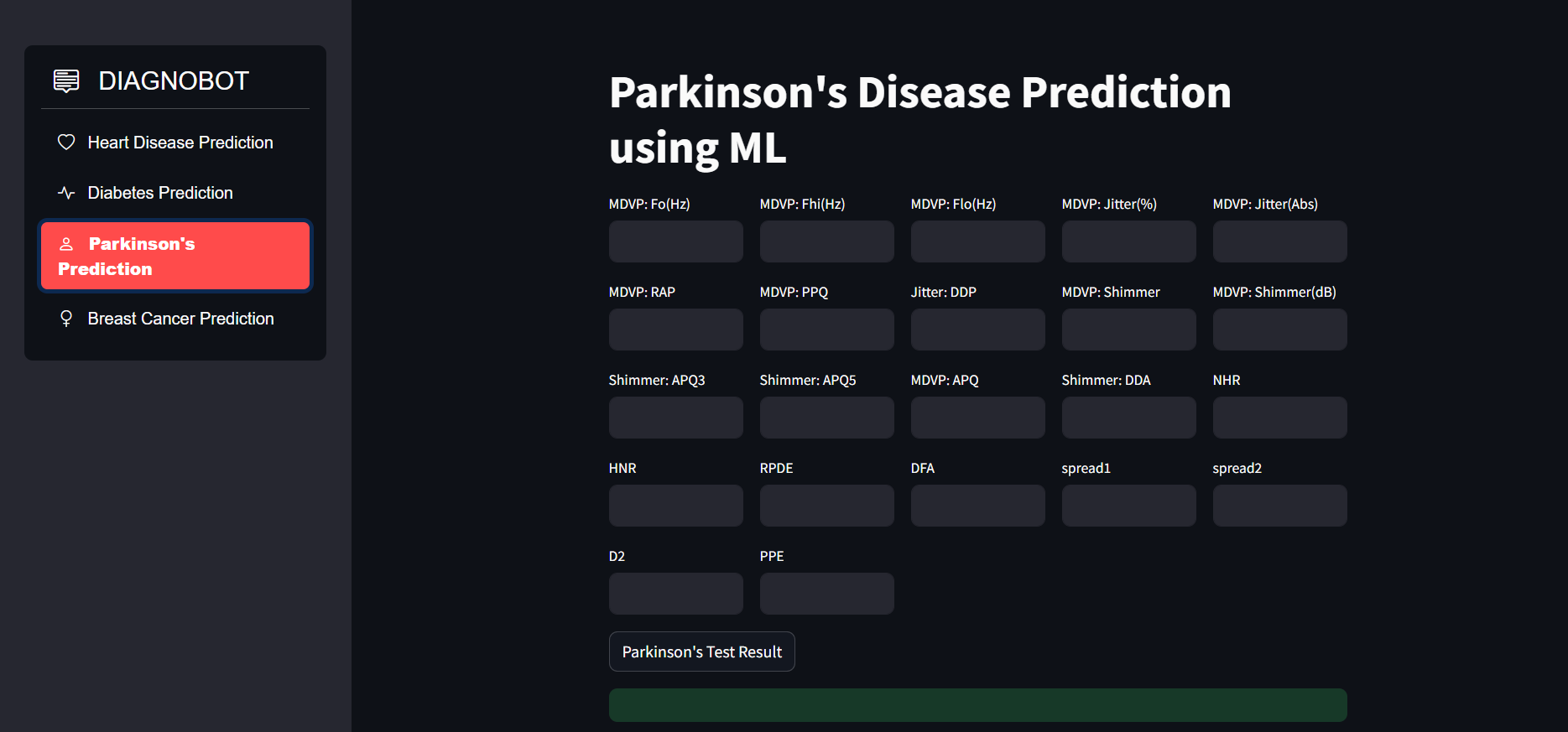
            else:

                cancer\_diagnosis = 'The person is not diagnosed with breast cancer.'

    st.success(cancer\_diagnosis)

****

****

****

**Bibliography**

1. **Python Software Foundation.** (2022). Python 3 Documentation. Retrieved from Python Software Foundation. (2022). Python 3 Documentation. Retrieved from <https://docs.python.org/3/>
2. **Jupyter.** (2022). Project Jupyter Documentation. Retrieved from <https://jupyter.org/documentation>
3. **Streamlit Documentation.** (2022). Streamlit Documentation. Retrieved from <https://docs.streamlit.io/>
4. **Pickle Documentation. (**2022). Pickle Documentation. Retrieved from <https://docs.python.org/3/library/pickle.html>
5. **Scikit-learn Documentation.** (2022). Scikit-learn Documentation. Retrieved from <https://scikit-learn.org/stable/documentation.html>
6. **Kaggle**. (2022). Kaggle Documentation. Retrieved from <https://www.kaggle.com/docs>
7. **Stack Overflow.** (2022). <https://stackoverflow.com/>: For code-related queries and community support.