DISEASE PREDICTION SYSTEM

Project Report

Submitted by

AKHIL CB (MEC19CS001)

AMARRISH MS (MEC19CS004)

SAHAD SHAMSUDHEEN (MEC19CS019)

VARTAK ANNURAG PARAG (MEC19CS023)

To

The APJ Abdul Kalam Technological University in the partial fulfilment of the

Requirements for the award of the Degree

Of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

JULY 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALABAR COLLEGE OF ENGINEERING AND TECHNOLOGY,

PALLUR (P.O), THRISSUR, KERALA - 679 532

DISEASE PREDICTION SYSTEM

Project Report

Submitted by

AKHIL CB (MEC19CS001)

AMARRISH MS (MEC19CS004)

SAHAD SHAMSUDHEEN (MEC19CS019)

VARTAK ANNURAG PARAG (MEC19CS023)

To

The APJ Abdul Kalam Technological University in the partial fulfilment of the

Requirements for the award of the Degree

Of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

JULY 2023

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALABAR COLLEGE OF ENGINEERING AND TECHNOLOGY,

PALLUR (P.O), THRISSUR, KERALA - 679 532

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MALABAR COLLEGE OF ENGINEERING AND TECHNOLOGY,

PALLUR (P.O), THRISSUR, KERALA -679 532

JULY 2023

CERTIFICATE

Certified that the Project Report entitled **"DISEASE PREDICTION SYSTEM"** is the bonafide record of the work carried out by **AHKIL CB (MEC19CS001), AMARRISH MS (MEC19CS004), SAHAD SHAMSUDHEEN (MEC19CS019), VARTAK ANNURAG PARAG (MEC19CS023)** in partial fulfilment of the award of Bachelor of Technology in Computer Science and Engineering of APJ Abdul Kalam Technological University, during the year 2019-2023.

Project Coordinator	**Project Guide**	**Head of Department**
Mrs Nufaila Thasni	Ms Meenu	Mr Arun G
Asst. Professor	Asst. Professor	Asst. Professor & HOD
Dept. of CSE	Dept. of CSE	Dept. of CSE

ACKNOW LEDGEM ENT

We thank almighty God, who laid the foundation for knowledge and wisdom and has always been the source of our strength, and inspiration, and who guides us throughout.

We express our gratitude to **Dr. MANZOOR**, The Director, for fostering an excellent academic climate in the college and for his support and encouragement throughout the course period.

We express our sincere thanks to **Prof. Dr. P BABU**, The Principal, for giving us the permission to present this project and providing us with all facilities for the completion of the same.

We express our deep sense of gratitude to **Mr ARUN G**, HOD, Dept. of Computer Science and Engineering for providing us with the best facilities and atmosphere for the creative work guidance and encouragement.

We would like to thank our coordinator **Mrs NUFAILA THASNI**, Asst. Professor of Computer Science and Engineering Dept., and our project guide **Ms MEENU**, Asst. Professor of Computer Science and Engineering Dept., for the expert suggestion and support during every stage of this work. We express our sincere thanks to all lecturers in the CSE dept. For providing us with the valuable guidance and encouragement through the work.

We also express our sincere thanks to all our classmates, friends and indeed our loving parents for their support, prayer and inspiration during every stage of this work.

DECLARATION

We**, AHKIL CB (MEC19CS001), AMARRISH MS (MEC19CS004), SAHAD SHAMSUDHEEN (MEC19CS019), VARTAK ANNURAG PARAG (MEC19CS023)** hereby declare that the project report entitled **DISEASE PREDICTION SYSTEM** done by me under the guidance of **Ms. MEENU** (Assistant Professor, Computer Science department), MCET is submitted in partial fulfilment of the requirements for the award of Bachelor of Technology degree in Computer Science and Engineering.

DATE: **SIGNATURE OF THE CANDIDATES**

PLACE: **AHKIL CB**

AMARRISH MS

SAHAD SHAMSUDHEEN

VARTAK ANNURAG PARAG

ABSTRACT

Machine learning is used extensively in medical diagnosis to predict the existence of diseases. Existing classification algorithms are frequently used for automatic detection of diseases. But most of the times, they do not give 100% accurate results. Boosting techniques are often used in Machine learning to get maximum classification accuracy. Though several boosting techniques are in place but the Ada Boost and XGBoost algorithm is doing extremely well for some selected data sets. Building an XGBoost model is simple but improving the model by tuning the parameters is a challenging task. There are many parameters to the XGBoost algorithm and deciding what set of parameters to tune and the ideal values of these parameters is a cumbersome and time taking task. We, in this paper, tuned the Ada Boost and XGBoost model for the first time for disease prediction and got 99% accuracy by tuning some of the hyper parameters. It is observed that the model proposed by us exhibited highest classification accuracy compared to all other models built till now by machine learning researchers and some regularly used algorithms like Support Vector Machines (SVM), Naive Bayes (NB), C4.5 Decision tree, Random Belief Networks, Alternating Decision Trees (ADT) experimented by us.

Keywords: Machine Learning, XG Boost, Ada Boost, Prediction, Accuracy.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Every year approximately 10 lakh patients are diagnosed who are suffering with liver cirrhosis. Liver disease is the tenth most usual reason of deaths in India as stated by the studies of World Health Organization. Identifying infections at an early stage is a difficult job as the functions in a normally way, even when it is partially damaged In this scenario, automatic diagnosis tools will support the doctors a lot for timely detection of the disease and improves the patients' survival rate. Conventional classification methods are used frequently in many automatic medical diagnosis tools which are less accurate. Hence there is a need for sophisticated classifiers and techniques like boosting for more accurate prediction. In gradient boosting, the loss function is minimized using the gradients .In each round of training, a weak learner is constructed and the predictions are compared with the expected outcome. Error rate of the model is the difference between predicted outcome and expected outcome. Now the gradients or partial derivative of the loss function can be calculated by using the errors. So it represents the steepness of the error function. The parameters of the model can be altered in order to decrease the error in the next round of training by decreasing the gradient. In Gradient Boosting, we merge the predictions given by multiple models and the boosted model predictions are optimized. Thus, the gradients used in the current training by fitting the next tree to these values. Extreme Gradient Boosting is sequential ensemble technique. It calculates second partial derivatives of the loss function to comprehend the trends of the gradients and to obtain the minimum of the loss function. Normal gradient boosting methods use the loss function of the base decision tree as a substitute so as to minimize the error of the overall model whereas for estimation; XG Boost uses the 2nd order derivative. It uses sophisticated L1 and L2 regularization technique to improve model generalization. Moreover, training XG Boost model is very fast and can be parallelized too.

DESCRIPTION

An expert system based on machine learning can reduce the medical test's associated costs and it also enhances the process of diagnosis. In the previous studies, researchers have developed

various diagnostic systems for the prediction of disease based on different techniques by the development of various diagnostic systems to disease diagnostic barriers and improve predictive accuracy, we are trying to develop a diagnostic system based on XG Boost (Extreme Gradient Boosting) Classifier. XG Boost Algorithm is the advanced implementation of gradient boosting algorithm and has been successfully applied to some studies. The early diagnosis of heart disease plays a vital role in making decisions on lifestyle changes in high-risk patients and in turn reduce the complications. In this project, we have developed and researched about models for heart disease prediction through the various heart attributes of patient and detect impending heart disease using Machine learning techniques on the dataset available in the UCI repository, further evaluating the results using confusion matrix and cross validation. The author had used various machine learning techniques to predict the accuracy of test model such as XG Boost and AdaBoost Classifier and we have clean dataset which can be readily used without cleaning or modifying the dataset.

MOTIVATIONS

The motivation behind developing the disease prediction system is to address the challenges and limitations of traditional disease diagnosis methods in healthcare. The project seeks to leverage the power of machine learning algorithms and online platforms to improve the accuracy, accessibility, and efficiency of disease prediction and consultation processes. The following factors contribute to the motivation for this project:

1. Early Disease Detection: Early detection of diseases is crucial for timely intervention and effective treatment. Traditional diagnostic methods often rely on manual examination and subjective judgment, leading to potential delays or misdiagnosis. By utilizing machine learning algorithms, the disease prediction system aims to improve the accuracy and efficiency of disease detection, allowing for early intervention and better patient outcomes.

2. Increasing Healthcare Demand: The demand for healthcare services continues to rise, posing challenges to healthcare systems worldwide. The disease prediction system provides an innovative approach to healthcare delivery by leveraging technology to bridge the gap between patients and doctors. It facilitates online consultations, reducing the need for physical visits and enabling patients to access healthcare services conveniently.

3. Limited Access to Specialists: Accessing specialized doctors or experts can be challenging, especially for patients residing in remote areas or facing mobility constraints. The disease prediction system aims to overcome this limitation by recommending suitable doctors based on predicted diseases. This feature enables patients to connect with the right specialists for consultation and receive appropriate medical advice irrespective of their geographical location.

4. Personalized Healthcare: Every individual may exhibit unique symptoms, and accurate disease prediction requires considering various factors and symptom patterns. The machine learning algorithms used in the project can analyse large datasets and identify complex relationships between symptoms and diseases. This enables the system to provide personalized disease predictions, improving the accuracy and effectiveness of healthcare services.

5. Convenience and Cost Reduction: Traditional healthcare processes often involve long waiting times and incur significant costs. The disease prediction system offers an online platform that allows patients to input their symptoms, receive disease predictions, and consult doctors remotely. This reduces waiting times, eliminates the need for unnecessary visits, and potentially lowers healthcare costs for patients.

6. Technological Advancements: The advancements in machine learning, web development, and data management technologies have opened up new possibilities in healthcare. Leveraging these technologies, the disease prediction system can harness the power of data analysis and predictive modelling to enhance disease prediction accuracy and streamline the healthcare process.

PROPOSED SYSTEM

The proposed system aims to develop a comprehensive disease prediction and consultation platform that leverages machine learning algorithms (XG Boost and Ada Boost) to accurately predict diseases based on user-provided symptoms. The system will provide a user-friendly and accessible interface for doctors, patients, and administrators, enabling efficient disease prediction, doctor recommendations, and online consultations. The key components and functionalities of the proposed system include:

1. User Management:

• Implement user authentication to ensure secure access to the system.

• Define different user roles, such as doctors, patients, and administrators, with role-based access control to restrict system functionalities based on user roles.

2. Symptom Input and Disease Prediction:

• Allow patients to input their symptoms through an intuitive interface.

• Utilize machine learning algorithms (XG Boost and Ada Boost) trained on historical data to predict diseases based on the input symptoms.

• Present the predicted disease(s) along with the confidence level or probability to the patient.

3. Doctor Recommendation:

• Based on the predicted disease, provide a list of suitable doctors who specialize in treating the identified disease.

• Include doctor profiles, including qualifications, areas of expertise, experience, and contact information.

• Enable patients to select a preferred doctor for further consultation.

4. Online Consultation:

• Facilitate online consultations between doctors and patients through real-time messaging and video conferencing features.

• Allow patients to schedule appointments with doctors based on their availability.

• Provide a secure and private platform for confidential communication between doctors and patients.

5. Dashboard and Reporting:

• Develop a user-friendly dashboard for doctors and administrators to manage appointments, view patient information, and access reports.

• Generate reports and analytics on disease trends, patient demographics, and consultation history for monitoring and analysis purposes.

6. Integration and Technology Stack:

• Utilize HTML, CSS, and Bootstrap for designing an intuitive and responsive user interface.

• Implement the Django web framework (Python-based) for back-end development and database management.

• Utilize PostgreSQL as the database management system for efficient storage and retrieval of user data.

The proposed system aims to provide an accurate disease prediction platform, facilitate seamless doctor-patient interactions, and enhance the overall healthcare experience. It leverages machine learning algorithms and modern web technologies to deliver personalized and convenient healthcare services to patients while assisting doctors in providing timely and targeted treatments.

SCOPE OF PROJECT

The scope of the Disease Prediction System project encompasses the development of a web-based application that utilizes machine learning algorithms to predict diseases based on patient symptoms. The project includes features such as user authentication, role-based access control, doctor suggestions for predicted diseases, online consultation capabilities, an intuitive user interface, database management, an administration module, and thorough testing and quality assurance. The primary goal is to create a comprehensive and user-friendly system that accurately predicts diseases, facilitates online consultations between doctors and patients, and ensures secure access and data management. The project scope also involves deploying the system on a suitable platform to ensure its availability and accessibility. Throughout the project, close collaboration with stakeholders and continuous communication will be maintained to define and refine the scope based on specific requirements and constraints.

CHAPTER 2

LITERATURE SURVEY

The prediction of diabetes as well as combined clustering and XG Boost algorithm for heart disease prediction. Some machine learning algorithms, such as Support Vector Machine, Decision Tree, Naive Bayes, Logistic Regression, and BP neural network have also been used by the industry and academia for disease prediction, but the existing methods still have major limitations. GBDT (The Gradient Boosting Decision Tree) algorithm is improved by Chen Tianqi, and an optimal distributed decision gradient lifting library XG Boost (Extreme Gradient Boosting) is proposed. The principle is implemented by iterative calculation of weak classifier. In order to get accurate classification results, this paper proposes to make use of XG Boost algorithm with data preprocessing in the prediction of diabetes with the experiment data from National Institute of Diabetes and Digestive and Kidney Diseases establishing a classification model to predict the results. The rest of the paper is organized briefly concludes the XG Boost algorithm and model training mechanism, the experiments and analysis.

The approach proposed in this paper uses machine learning to predict survival of a heart patient. The approach uses patient's data like gender, age, hypertension, type of work, glucose level, body mass index, etc. to predict his/her chances of death due to heart failure. The dataset is retrieved from Kaggle AI based grouping calculations specifically XG boost have been implemented and their performance has been compared using parameters like precision. XG boost is a good implementation of the gradient enhancement method. Even though there may not be new mathematical developments here, it is a gradient gain alternative that can be carefully designed for optimization and accuracy. It consists of a linear version, and the newborn tree may be a technique that uses various AI calculations to verify whether a fragile newbie will create a reliable newbie to improve the accuracy of the version. From (impulsive) and parallel learning (bagging), for example, random forest. Data collection can be a method that can be used to control the display of an AI version with advanced talent and precision processing is faster than enhancing gradients.

Early signs may be mild and go unnoticed. Symptoms often begin on one side of your body and usually remain worse on that side, even after symptoms begin to affect both sides. Sometimes it is quite difficult to detect whether there is Disease is present in the patient's body. Our goal is to develop a model well enough so that it can detect the early stage of Disease. In this Python Machine learning project, we will build a model using XG Boost, which we can detect the presence of disease in one's body. XG Boost is a new Machine Learning algorithm designed with speed and performance in mind. XG Boost stands for extreme Gradient Boosting is based on decision trees. In this project using the Python libraries such as numpy, scikit learn, XG Boost we built a model using XGB Classifier. First, we will load the data, get the features and labels, scale the features then split the dataset, built an XGB classifier, and then calculate the accuracy of our model.

Machine Learning is the data science process that allows computer to gain information insight to the pattern and existing data to predict the outcomes and trends of the data to programmed identified them. This can make tasks like diagnosing disease more automated efficient, and accurate with a robust machine learning model and also identify patterns and characteristics in the data in ways that humans may not noticed. Machine learning algorithm to train data input and use statistical analysis in order to create specific output. Machine learning has various algorithm to used predict the disease. XG Boost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It is used in prediction problems involving unstructured data (images, text, etc.) artificial neural networks tend to outperform all other algorithms or frameworks. XG Boost provides a parallel tree boosting (also known as GBDT, GBM) that solve many data science problems in a fast and accurate way. It implements machine learning algorithms under the Gradient Boosting framework

The kidneys are one of the important organs for humans with various tasks to keep humans healthy. One of them is to remove excess water and toxins from the blood. The chronic kidney is a disease that interferes with kidney function that is progressive and irreversible in which the body's ability to fail to maintain metabolism and fluid and electrolyte balance resulting in uremia. There have been many studies aimed at predicting chronic kidney disease. In this study, using the C4.5 algorithm as a classification algorithm and by adding the Adaboost algorithm. The results showed that using the algorithm obtained an accuracy value of 95%, 92% precision,

and Area Under the Curve by 1. Whereas using the algorithm combined with Adaboost got an accuracy value is precision, and Area Under the Curve by 1. Based on the results accuracy, precision, and Area Under the Curve, it can be seen by using Adaboost in the algorithm can improve the results of accuracy and precision with very good classification, because the Area Under the Curve value is accepted.

Machine Learning and AI techniques have become more popular in healthcare in recent years, particularly for disease diagnosis and risk prediction. This cutting-edge technology will boost processing efficiency while existing technology-based resources will provide a smart way to track and collect data on patient health. To develop accurate forecasts regarding a specific condition, data from diverse sources must be combined with machine learning and artificial intelligence algorithms. Years of medical data collection have given clinicians a new way to diagnose patients. Ada Boost is a classification algorithm. It creates numerous weak learners and then combines them to create a powerful classifier. This process illustrates that an error made by one learner has an impact on other learners' errors. This process of building weak learners and adjusting sample weights continues until the weak learners' conditions are met. The voting power of the classifier and the classifier's decision criteria are used to calculate the function. The maximum vote received by the label when the predictive function is applied is used to determine the test sample's final prediction. In Ada Boost, the classifier's correct predictions have greater voting power than the classifier's major misclassifications

Hence there is a need for sophisticated classifiers and techniques like boosting for more accurate prediction In gradient boosting, the loss function is minimized using the gradients .In each round of training, a weak learner is constructed and the predictions are compared with the expected outcome. Error rate of the model is the difference between predicted outcome and expected outcome. Now the gradients or partial derivative of the loss function can be calculated by using the errors. So it represents the steepness of the error function. The parameters of the model can be altered in order to decrease the error in the next round of training by decreasing the gradient. In Gradient Boosting, we merge the predictions given by multiple models and the boosted model predictions are optimized. Thus, the gradients used in the current training by fitting the next tree to these values. Extreme Gradient Boosting is sequential ensemble technique. It calculates second partial derivatives of the loss function to comprehend the trends of the gradients and to

obtain the minimum of the loss function. Normal gradient boosting methods use the loss function of the base decision tree as a substitute so as to minimize the error of the overall model whereas for estimation; XG Boost uses the 2nd order derivative. It uses sophisticated L1 and L2 regularization technique to improve model generalization. Moreover, training XG Boost model is very fast and can be parallelized too.

Thus, diagnosis and prediction of heart disease remain mandatory. Clinical decision support systems based on machine learning techniques have become the primary tool to assist clinicians and contribute to the automated diagnosis. This paper aims to predict heart disease using Random Forest algorithm enhanced with the boosting algorithm Ada Boost. The model is trained and tested on University of California Irvine (UCI) Cleveland and Stat log heart disease datasets using the most relevant features 14 attributes. The result shows that Random Forest algorithm combined with Ada Boost algorithm achieved higher accuracy than applying only Radom Forest algorithm, 96.16%, 95.98%, respectively. We compare our suggested model to report machine learning classifiers. Indeed, the obtained result is supporting the efficiency and validity of our model. Besides, the proposed model achieved high accuracy compared to existing studies in the literature that confirmed that a clinical decision support system could be used to predict heart disease based on machine learning algorithms. The proposed method achieved an accuracy of 93% on the Cleveland dataset and an accuracy of 91% Framingham test set. yet the validation was carried out using only the train-test split method, which may cause over fitting trained and tested the optimized XG Boost approach only on the Cleveland dataset and achieved 91.80% accuracy using the cross-validation method.

CHAPTER 3

SYSTEM ANALYSIS

FUNCTIONAL REQUIREMENTS

1. User Registration and Authentication:

• Users should be able to register and create an account.

• The system should authenticate users and validate their credentials.

2. Symptom Input and Disease Prediction:

• Patients should be able to input their symptoms.

• The system should use machine learning algorithms (XG Boost and Ada Boost) to predict diseases based on the symptoms.

3. Doctor Recommendation:

• The system should recommend relevant doctors based on the predicted disease.

• Doctor recommendations should be based on specialization, experience, and availability.

4. Online Consultation:

• Patients should be able to schedule online consultations with recommended doctors.

• The system should facilitate real-time messaging and video conferencing between doctors and patients.

5. User Management and Roles:

• The system should allow administrators to manage user accounts and roles.

• Different roles (doctors, patients, administrators) should have specific permissions and access levels.

6. Appointment Management:

• Doctors should be able to manage their appointments and availability.

• Patients should be able to view and reschedule their appointments.

NON FUNCTIONAL REQUIREMENTS

1. Usability:

• The user interface should be intuitive, user-friendly, and easy to navigate.

• The system should provide clear instructions and guidance for symptom input and consultation processes.

2. Performance:

• The system should handle multiple concurrent users without significant performance degradation.

• Disease prediction and doctor recommendation processes should be fast and responsive.

3. Security:

• The system should ensure secure user authentication and data transmission.

• Patient data, including symptoms and consultation records, should be securely stored and protected.

4. Reliability:

• The system should have high availability, minimizing downtime and disruptions.

• It should handle errors and exceptions gracefully, providing appropriate error messages and notifications.

5. Scalability:

• The system should be scalable to accommodate a growing number of users, doctors, and consultations.

• It should handle an increasing volume of data efficiently.

6. Compatibility:

• The system should be compatible with different web browsers and devices, ensuring accessibility for users.

• It should adhere to web standards and best practices for cross-platform compatibility.

7. Maintainability:

• The system should be easy to maintain and update with minimal disruption.

• Code should be well-documented and modular to facilitate future enhancements and bug fixes.

8. Privacy and Data Protection:

• The system should comply with relevant data protection regulations and ensure patient privacy.

• Patient data should be anonymized and stored securely to protect confidentiality.

HARDWARE REQUIREMENTS

Processor: Core i3 and above

RAM: 2GB

Hard Disk: 8GB

Internet Connection

SOFTWARE REQUIREMENTS

Front end: HTML, CSS, Bootstrap, JavaScript, JQuery

Back end: Django (python based web framework)

Database: PostgreSQL

Tools: PgMyadmin, Orange

CHAPTER 3

EXISTING SYSTEM

Manual Diagnosis:

In the absence of an automated system, the existing system may rely on manual diagnosis conducted by doctors or medical professionals. Patients provide their symptoms, and doctors analyse the symptoms, medical history, and perform various tests to make a diagnosis.

Limited Accessibility:

Patients may need to physically visit a healthcare facility or clinic to consult with doctors. This can be time-consuming, especially for patients with limited mobility or residing in remote areas.

Lack of Predictive Analysis:

The existing system may not have the capability to predict diseases based on symptoms or utilize machine learning algorithms. Diagnosis and treatment decisions primarily rely on the expertise and experience of the doctors.

Inefficient Doctor-Patient Matching:

The existing system may not have an automated process to suggest relevant doctors for specific diseases. Patients may have to rely on personal referrals or their own research to find suitable doctors.

Limited Online Consultation:

The existing system may not provide online consultation options for patients, which can be a constraint, especially in situations where physical visits are not feasible or convenient for the patients.

Manual Record Keeping:

Patient data and medical records may be stored in physical files or disconnected systems, making it challenging to access and analyse the data efficiently.

The purpose of developing the proposed Disease Prediction System is to address the limitations of the existing system by leveraging machine learning algorithms for accurate disease prediction, enabling online consultation, improving accessibility to doctors, and providing a user-friendly interface for patients and doctors. The proposed system aims to enhance the efficiency and effectiveness of disease prediction, diagnosis, and patient care.

METHODOLOGY

Requirement Analysis:

Begin by gathering and analysing the requirements of the Disease Prediction System. This involves identifying the key functionalities, user roles, data inputs, expected outputs, and system constraints. Collaborate with stakeholders, including doctors, patients, and administrators, to understand their needs and expectations.

Data Collection and Preparation:

Collect a comprehensive and reliable dataset containing patient symptoms, corresponding diseases, and other relevant information. Ensure the dataset is appropriately labeled and validated. Pre-process the data by performing tasks such as data cleaning, normalization, feature selection, and feature engineering to prepare it for training the machine learning algorithms.

Algorithm Selection:

Evaluate various machine learning algorithms suitable for disease prediction, such as XG Boost and Ada Boost, based on their performance metrics, computational requirements, and compatibility with the project requirements. Select the most appropriate algorithms for training and prediction.

System Design and Architecture:

Design the system architecture, considering factors like scalability, performance, security, and user interface. Define the components, modules, and interactions between them. Determine the necessary technologies and frameworks, such as Django for back-end development and HTML, CSS, Bootstrap, JavaScript, and jQuery for front-end development.

Implementation:

Start implementing the Disease Prediction System based on the defined architecture. Develop the user authentication module, role-based access control mechanism, disease prediction algorithm integration, doctor suggestion feature, online consultation functionality, and other

system components. Ensure proper integration and smooth communication between the front-end and back-end.

Testing and Quality Assurance:

Conduct thorough testing to validate the functionality, accuracy, and performance of the system. Perform unit testing to verify the individual components and integration testing to ensure the seamless functioning of the system as a whole. Perform system testing with sample inputs to validate disease prediction accuracy and doctor suggestion functionalities. Address any bugs, issues, or performance bottlenecks identified during testing.

Deployment:

Deploy the Disease Prediction System on a suitable server or hosting platform, ensuring its availability and accessibility to users. Configure the necessary infrastructure components such as the web server, database server, and networking settings. Perform system monitoring to ensure smooth operation and make any required optimizations or updates.

User Training and Documentation:

Provide training to doctors, patients, and administrators on how to use the Disease Prediction System effectively. Prepare user documentation, including user manuals and guides, to assist users in understanding the system's functionalities and features.

Maintenance and Upgrades:

Regularly maintain and update the Disease Prediction System to address any identified issues, incorporate user feedback, and adapt to evolving requirements. Monitor system performance, security, and scalability. Continuously improve the system by adding new features or enhancing existing functionalities.

User Interface Design:

Focus on designing an intuitive and user-friendly interface for patients, doctors, and administrators. Conduct user research and usability testing to gather feedback and refine the user

interface design. Ensure that the interface provides a seamless experience for entering symptoms, viewing predictions, scheduling consultations, and accessing relevant information.

User Authentication and Role-Based Access Control:

Implement a robust authentication mechanism to verify the identity of each user. Develop role-based access control to define different levels of access and functionality based on user roles (doctor, patient, admin). This ensures secure access to the system and protects sensitive patient data.

Integration of External APIs:

Explore the integration of external APIs to enhance the Disease Prediction System's capabilities. For example, integrate APIs that provide additional data sources for disease prediction, enable real-time doctor availability information, or facilitate online payment gateways for consultation fees.

Privacy and Security:

Prioritize the security and privacy of patient data. Implement measures such as data encryption, secure transmission protocols (e.g., HTTPS), and proper access controls to protect sensitive information. Comply with relevant data protection regulations, such as GDPR or HIPAA, depending on the jurisdiction.

Performance Optimization:

Optimize the system's performance to ensure fast response times and efficient resource utilization. Consider techniques like caching, query optimization, and code profiling to identify and address bottlenecks. Conduct performance testing to evaluate the system's scalability under different user loads.

Error Handling and Logging:

Implement comprehensive error handling mechanisms to handle exceptions, validate user inputs, and provide meaningful error messages to users. Log system activities, errors, and user interactions for monitoring, debugging, and auditing purposes.

User Feedback and Iterative Development:

Seek feedback from users, including doctors and patients, throughout the development process. Conduct user testing sessions, surveys, or interviews to gather insights and identify areas for improvement. Incorporate user feedback through iterative development cycles to refine the system's functionalities and user experience.

Documentation and User Support:

Prepare comprehensive documentation that includes installation instructions, system usage guidelines, and troubleshooting information. Provide user support channels, such as a help desk or email support, to address user queries and issues promptly.

Project Management:

Adopt an agile project management approach, such as Scrum or Kanban, to ensure effective collaboration, task prioritization, and progress tracking. Divide the project into sprints or iterations with clear objectives, and hold regular meetings to review progress, address challenges, and plan next steps.

Training and Knowledge Transfer:

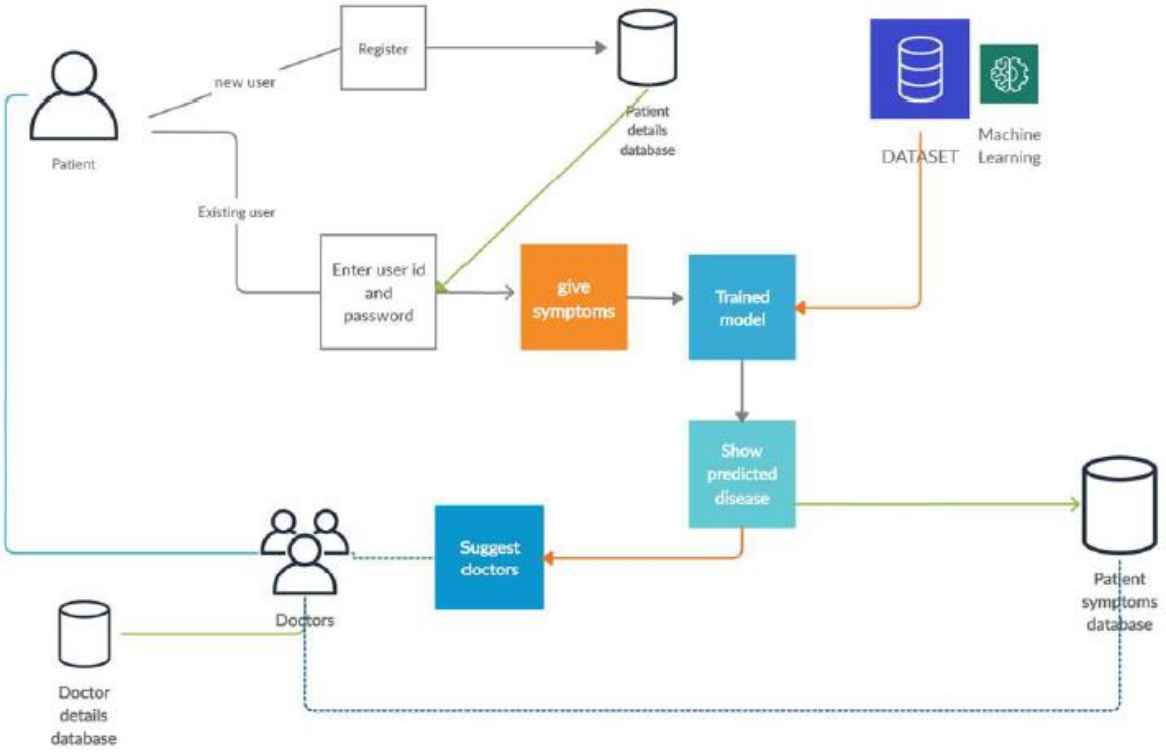
Conduct training sessions for doctors, patients, and administrators to familiarize them with the Disease Prediction System's features and functionalities. Provide ongoing support and training materials to ensure users can maximize the system's benefits.

Evaluation and Validation:

Evaluate the accuracy and performance of the disease prediction algorithms used in the system. Validate the system's predictions against known medical datasets or seek expert opinions to assess the reliability and effectiveness of the disease prediction capabilities.

DESIGN

ARCHITECTURAL DESIGN



User Registration and Authentication:

New users can register by providing their personal details, such as name, age, gender, and contact information.

Store the user details securely in a database.

Existing users can log in using their user ID and password.

Patient Symptoms Input:

Once logged in, the patient can enter their symptoms into the system.

The symptoms entered by the patient will be used as input for disease prediction.

Trained Model:

Train the machine learning model using the XG Boost and Ada Boost algorithms.

Use a dataset containing symptom-disease mappings to train the model.

The model should learn patterns and correlations between symptoms and diseases.

Accuracy of Disease Prediction may evaluate the accuracy of the disease prediction algorithms (XG Boost and Ada Boost) used in the system. Measure the precision, recall, and F1 score to assess the system's ability to correctly predict diseases based on the symptoms provided by patients. Compare the predictions against known medical datasets or expert opinions to validate the accuracy of the system.

Data Set:

Data collection has been done from the internet to identify the disease here the real symptoms of the disease are collected i.e. no dummy values are entered. The symptoms of the disease are collected from kaggle.com and different health related websites. This csv file contain 5000 rows of record of the patients with their symptoms (132 types of different symptoms) and their corresponding disease (40 class of general disease)

Disease Prediction:

Pass the patient's entered symptoms as input to the trained model.

The model will predict the most probable disease based on the symptoms provided.

The predicted disease will be stored in the patient symptoms database.

Doctor Recommendation:

Based on the predicted disease, suggest a relevant doctor or specialist.

Retrieve details of the recommended doctor from the doctor database.

Show the relevant details such as name, contact information, and specialization to the patient.

Patient Symptoms Database:

Maintain a database to store the symptoms entered by patients.

Store the patient's user ID, symptoms, predicted disease, and timestamp.

Doctor Database:

Maintain a database to store details of doctors or specialists.

Include information like name, contact information, specialization, and availability.

The disease prediction system have 3 users such as doctor, patient and admin.

- Each user of the system are authenticated by the system.
- There is a role based access to the system.
- The system allows the patient to give symptoms and according to those
- Symptoms the system will predict a disease.
- The system suggests doctors for predicted diseases.
- The system allows online consultation for patients.
- The system helps the patients to consult the doctor at their convenience by Sitting at home.

IMPLEMENTATION

TECHNOLOGIES USED

Set up the Development Environment: Install the necessary development tools and software, including Python, Django, HTML/CSS editors, and IDEs. Configure the database server (PostgreSQL) and any other required dependencies.

Database Design and Setup:

Design the database schema to store patient data, user information, disease details, doctor profiles, and other relevant entities. Implement the database schema using PostgreSQL and set up the necessary tables, relationships, and constraints.

Front-end Development:

Start developing the user interface using HTML, CSS, and JavaScript. Utilize frameworks like Bootstrap and jQuery to enhance the UI design and functionality. Implement the necessary forms, input fields, and validation mechanisms to capture patient symptoms and user inputs.

Back-end Development:

Implement the back-end functionality using the Django web framework and Python. Develop the user authentication system, role-based access control mechanisms, and APIs for data retrieval and manipulation. Integrate the disease prediction algorithms (XG Boost, Ada Boost) to predict diseases based on symptoms.

Implement Doctor Suggestion and Online Consultation:

Develop the logic and algorithms to suggest relevant doctors based on predicted diseases. Implement features for scheduling and managing online consultations between patients and doctors. Handle communication, notifications, and data exchange securely.

User Authentication and Security:

Implement user authentication and authorization mechanisms to ensure secure access to the system. Apply encryption and secure transmission protocols to protect sensitive data. Implement measures to prevent common security vulnerabilities, such as SQL injection and cross-site scripting (XSS).

Testing and Quality Assurance:

Conduct thorough testing at different stages of development. Perform unit testing to verify the functionality of individual components. Conduct integration testing to ensure seamless communication between front-end and back-end. Perform system testing to validate disease prediction accuracy, doctor suggestion functionality, and online consultation features. Address any identified issues or bugs.

Deployment:

Deploy the Disease Prediction System on a suitable server or hosting platform. Configure the server environment, set up the necessary databases, and deploy the application code. Ensure proper server configurations for optimal performance, security, and scalability.

User Training and Documentation:

Prepare user documentation, including user manuals and guides, to assist users in understanding the system's functionalities. Conduct training sessions for doctors, patients, and administrators to familiarize them with the system. Provide ongoing support channels for user inquiries and assistance.

Maintenance and Upgrades:

Regularly maintain and update the system to address any identified issues, incorporate user feedback, and adapt to changing requirements. Monitor system performance, security, and scalability. Continuously improve the system by adding new features or enhancing existing functionalities based on user feedback and emerging technologies.

User Registration and Authentication:

Implement user registration functionality to allow patients, doctors, and administrators to create accounts. Develop authentication mechanisms to validate user credentials during login. Consider implementing additional security measures such as two-factor authentication for enhanced user account protection.

Role-Based Access Control:

Implement role-based access control to restrict certain functionalities and system features based on user roles. Define different levels of access and permissions for patients, doctors, and administrators. Ensure that each user can only access and modify data relevant to their role.

Symptom Input and Disease Prediction:

Develop a user-friendly interface for patients to input their symptoms. Validate and sanitize user inputs to ensure accurate disease prediction. Utilize the trained XG Boost and Ada Boost models to predict diseases based on the entered symptoms. Display the predicted disease to the patient.

Doctor Suggestion Algorithm:

Implement an algorithm to suggest doctors based on the predicted disease. Consider factors such as doctor specialization, location, availability, and patient preferences when suggesting doctors. Present the suggested doctors to the patient, along with their profiles and contact information.

Online Consultation System:

Develop an online consultation system that allows patients to schedule appointments with doctors. Implement features such as appointment booking, calendar management, and notifications for both patients and doctors. Enable secure communication channels for online consultations, such as chat or video conferencing.

Integration with Payment Gateway:

If applicable, integrate a secure payment gateway to facilitate online consultation fees. Implement features to handle payment transactions and generate invoices or receipts for patients.

Ensure the payment gateway integration follows security best practices to protect sensitive financial information.

Patient Health Records:

Develop a module to store and manage patient health records securely. Allow patients to view their medical history, test results, and previous consultations. Implement features to enable doctors to access and update patient records, including diagnoses and treatment plans.

Error Handling and Logging:

Implement robust error handling mechanisms throughout the system. Capture and log errors, exceptions, and system events for troubleshooting and debugging purposes. Display user-friendly error messages when exceptions occur and handle them gracefully.

User Interface Enhancements:

Continuously improve the user interface to enhance usability and user experience. Incorporate user feedback to refine the design, layout, and navigation of the system. Ensure the interface is responsive and accessible across different devices and screen sizes.

Performance Optimization and Scalability:

Optimize the system's performance to ensure fast response times and efficient resource utilization. Employ caching techniques, database indexing, and query optimizations to improve system speed. Consider scalability requirements and design the system to handle a growing number of users and data.

Continuous Integration and Deployment:

Implement a continuous integration and deployment (CI/CD) pipeline to automate the build, testing, and deployment processes. Use tools like Git, Jenkins, or Docker to facilitate smooth and efficient code deployment to production environments.

Security and Data Privacy:

Implement strong security measures to protect sensitive patient data. Ensure compliance with data protection regulations, such as GDPR or HIPAA, depending on the jurisdiction. Employ

encryption techniques to secure data at rest and in transit. Regularly conduct security audits and vulnerability assessments to identify and mitigate potential risks.

System Monitoring and Analytics:

Set up monitoring tools to track system performance, user activity, and system health. Monitor server uptime, resource usage, and response times. Utilize analytics tools to gather insights into user behavior, disease prediction accuracy, and system usage patterns. Use this data to identify areas for improvement and make informed decisions.

Documentation and User Support:

Prepare comprehensive documentation that includes installation instructions, user guides, and system administration manuals. Provide user support channels such as email support, FAQs, or a help desk to assist users with any queries or issues

Data collection has been done from the internet to identify the disease here the real symptoms of the disease are collected i.e. no dummy values are entered. The symptoms of the disease are collected from kaggle.com and different health related websites. This csv file contain 5000 rows of record of the patients with their symptoms (132 types of different symptoms) and their corresponding disease(40 class of general disease) .

Throughout the implementation phase, ensure effective collaboration among the development team, stakeholders, and end-users. Adhere to coding standards, best practices, and project management methodologies. Regularly communicate progress, address challenges, and seek feedback from stakeholders to ensure a successful implementation of the Disease Prediction System.

WORKING OF PROJECT

The Disease Prediction System project works as follows:

1. User Registration and Login:

• Users can register for an account by providing their name, email, and password.

• The system validates the user details and creates a new user account.

• Registered users can then log in to the system using their credentials.

2. User Authentication and Role-Based Access:

• Upon login, the system authenticates the user's credentials to ensure they are valid and authorized to access the system.

• The system utilizes role-based access control (RBAC) to assign specific roles (doctor, patient, or admin) to each user, which determines their permissions and access to different functionalities.

3. Input Symptoms and Disease Prediction:

• Patients can input their symptoms through the user interface.

• The system validates and verifies the input symptoms.

• Machine learning algorithms, such as XG Boost and Ada Boost, are applied to predict the disease based on the input symptoms.

• The system displays the predicted disease to the patient.

4. Recommendation of Doctors:

• Based on the predicted disease, the system suggests relevant doctors who specialize in treating that particular disease.

• The system provides a list of recommended doctors to the patient, including their profiles and contact information.

5. Online Consultation:

• Patients can schedule online consultations with the recommended doctors.

• The system allows patients to choose a convenient date and time for the consultation.

• Patients can request an appointment with the chosen doctor.

• Once the appointment is confirmed, the patient and doctor can conduct an online consultation via real-time messaging or video conferencing.

6. Appointment Management:

• The system enables patients to view their upcoming appointments and manage them.

• Patients can reschedule or cancel appointments if needed.

• The system sends notifications and reminders to patients regarding their appointments.

7. Administrator Functions:

• The admin user has additional functionalities for managing user accounts and roles.

• The admin can monitor system activities, generate reports, and perform analytics on disease trends, patient consultations, etc.

Overall, the Disease Prediction System allows patients to input symptoms and receive predictions for diseases. It suggests relevant doctors for the predicted diseases and facilitates online consultations between patients and doctors. The system aims to provide convenient and accessible healthcare services by leveraging machine learning algorithms and modern web technologies.

TESTING

TEST CASES

1. Test Case: User Registration

• Description: Verify the user registration functionality.

• Steps:

1. Enter valid user details in the registration form.

2. Submit the form.

3. Check if the user is successfully registered and redirected to the login page.

• Expected Result: User registration is successful.

2. Test Case: Disease Prediction

• Description: Validate the accuracy of disease prediction based on input symptoms.

• Steps:

1. Input a set of symptoms.

2. Submit the symptoms for prediction.

3. Verify if the predicted disease matches the expected result based on the input symptoms.

• Expected Result: The predicted disease matches the expected disease.

3. Test Case: Doctor Recommendation

• Description: Verify the correctness of doctor recommendations based on predicted diseases.

• Steps:

1. Input symptoms and get a predicted disease.

2. Check if the system recommends doctors specializing in the predicted disease.

3. Verify if the recommended doctors are relevant and appropriate.

• Expected Result: The system recommends doctors specialized in the predicted disease.

4. Test Case: Online Consultation

• Description: Test the functionality of scheduling and conducting online consultations.

• Steps:

1. Select a doctor for an online consultation.

2. Schedule an appointment with the chosen doctor.

3. Initiate the online consultation and interact with the doctor through messaging or video conferencing.

• Expected Result: Online consultation is successfully scheduled and conducted.

5. Test Case: Role-Based Access Control

• Description: Validate the role-based access control system.

• Steps:

1. Log in with different user roles (doctor, patient, admin).

2. Verify that each role has access to the appropriate functionalities and restricted access to unauthorized functionalities.

• Expected Result: Each user role has the correct permissions and access rights.

Types of Testing:

1. Unit Testing: Test individual components and functions of the system in isolation.

2. Integration Testing: Test the integration and interaction between different system components.

3. Functional Testing: Verify that the system functions as expected and meets the specified requirements.

4. Performance Testing: Evaluate the system's performance under various load conditions, ensuring it can handle multiple users and process requests efficiently.

5. Security Testing: Test the system's security measures to identify vulnerabilities and ensure data protection.

6. User Acceptance Testing: Involve end-users to validate the system's usability, user-friendliness, and overall satisfaction.

7. Regression Testing: Re-test previously implemented features and functionalities to ensure new changes do not introduce any issues.

RESULT AND ANALYSIS

Performance Measure

The prediction performance of the algorithm is evaluated in this research using five performance measures based on the confusion matrix. Figure 5 illustrates the binary classification problem's confusion matrix structure. Distinct predicted and true values can be merged into four cases: TP, TN, FP, and FN.

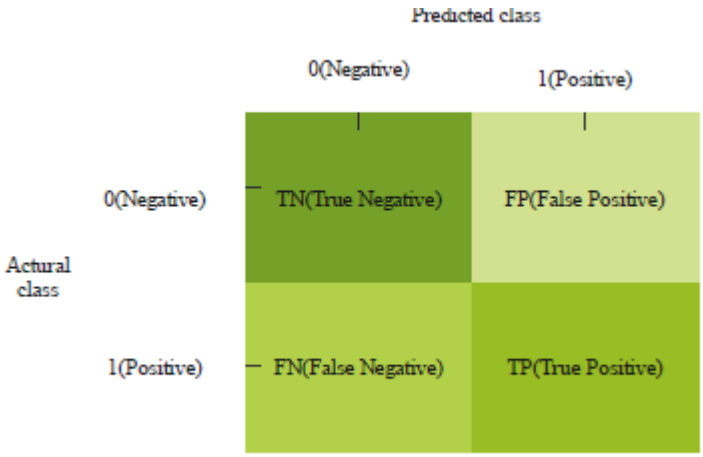


Figure 1. Confusion matrix

Using the data from the confusion matrix, the four evaluation indicators can be Calculated using the formula below.

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1 - Score} = \frac{1}{2} \text{ X } \frac{\text{Precision . Recall}}{\text{Precision + Recall}}$$

The ROC curve is a tool for examining the capacity of an algorithm for generalization. The False Positive Rate (FPR) is its horizontal axis and the True Positive Rate (TPR) is its vertical axis, both of which are calculated as follows.

$$\text{TPR} = \frac{\text{TP}}{\text{TP + FN}}$$

$$\text{FPR} = \frac{\text{FP}}{\text{TN + FP}}$$

Moreover, the Area Under ROC Cure (AUC) reflects how well a model predicts heart disease.

The Performance of Algorithms

In this section, the six algorithms are trained using a five-fold cross-validation method, and the proposed framework is validated using pre-processed HDD. The confusion matrix for the six algorithms is displayed in Table 3, which shows in detail the percentage of the number of the four cases TP, FP, TN, FN in the prediction results of the six algorithms. Table 2 depicts the average performance of the six different algorithms on five metrics: Accuracy, Precision, Recall, F1-score, and AUC.

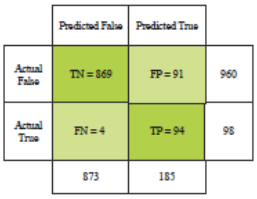
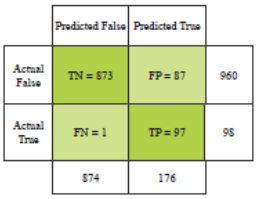
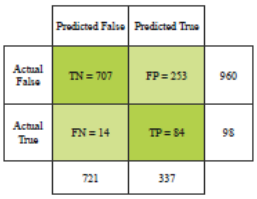
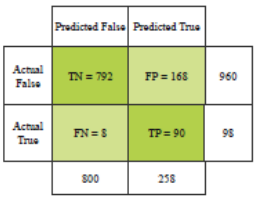
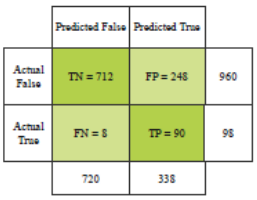
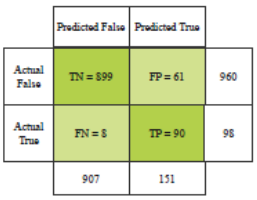
Algorithm	Confusion matrix	Description
RF		**TN**: MACCE was correctly predicted not to occur for 869 samples, and the actual sample MACCE does not occur. **TP**: MACCE was correctly predicted to occur for 94 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 91 samples, and the actual sample MACCE does not occur. **FN**: MACCE was incorrectly predicted not to occur for 4 samples, and the actual sample MACCE occurred.
KNN		**TN**: MACCE was correctly predicted not to occur for 873 samples, and the actual sample MACCE did not occur. **TP**: MACCE was correctly predicted to occur for 97 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 87 samples, and the actual sample MACCE did not occur. **FN**: MACCE was incorrectly predicted not to occur for one sample, and the actual sample MACCE occurred.
LR		**TN**: MACCE was correctly predicted not to occur for 707 samples, and the actual sample MACCE does not occur. **TP**: MACCE was correctly predicted to occur for 84 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 253 samples, and the actual sample MACCE did not occur. **FN**: MACCE was incorrectly predicted not to occur for 14 samples, and the actual sample MACCE occurred.
DT		**TN**: MACCE was correctly predicted not to occur for 792 samples, and the actual sample MACCE did not occur. **TP**: MACCE was correctly predicted to occur for 90 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 168 samples, and the actual sample MACCE did not occur. **FN**: MACCE was incorrectly predicted not to occur for eight samples, and the actual sample MACCE occurred.
NB		**TN**: MACCE was correctly predicted not to occur for 712 samples, and the actual sample MACCE did not occur. **TP**: MACCE was correctly predicted to occur for 90 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 248 samples, and the actual sample MACCE did not occur. **FN**: MACCE was incorrectly predicted not to occur for eight samples, and the actual sample MACCE occurred.
XGBoost		**TN**: MACCE was correctly predicted not to occur for 899 samples, and the actual sample MACCE did not occur. **TP**: MACCE was correctly predicted to occur for 90 samples, and the actual sample MACCE occurred. **FP**: MACCE was incorrectly predicted to occur for 61 samples, and the actual sample MACCE did not occur. **FN**: MACCE was incorrectly predicted not to occur for eight samples, and the actual sample MACCE occurred.

Table 1. Confusion matrix of six algorithm

The most crucial metric for evaluating how well a model predicts is accuracy, of which Xgboost achieves 93.44%. Random Forest and K-Nearest Neighbor achieve 91.15% and 91.77%, respectively. Decision Tree comes in at 83.35%, while Nave Bayes and Logistic Regression fall short at 75.5%. With performance rates of 75.85% and 74.81%, respectively, both Naïve Bayes

and Logistic Regression underperformed. The two measures, Precision and Recall, have a tendency to be inversely correlated; that is, when the precision is high, the recall is typically lower, and vice versa when the recall is high. The most confident samples should be chosen in order to raise the number of correct predictions in MACCE, which will reduce the number of FN and lower the recall. The number of FP will rise when the sample size is increased, resulting in low precision and decreasing the ability to predict the occurrence of MACCE to the greatest extent possible.

Owing to the dataset's uniform distribution of positive and negative samples and the model preference that the occurrence of MACCE can be predicted, the Precision of the general model in this experiment is lower than its Recall. The weakest performers were Naïve Bayes and logistic regression, earning 71.05% and 70.59%, respectively, while Xgboost had the best F1-Score at 94.86%. The ROC curves of these algorithms, which represent the AUC metric in the table, are shown in Figure 6. The higher the value of AUC, the stronger the generalization ability of the model, which is expressed in the ROC curve as the curve close to the upper-left corner of the graph. Naïve Bayes had the worst performance, with an AUC value of 70.59%, but Xgboost had the highest AUC value of 92.44%. When all evaluating metrics are considered, Xgboost algorithm has a significant advantage in predicting the occurrence of MACCE, and K-Nearest Neighbor is the second best. Apart from that, the comparison showed that Naïve Bayes and Logistic Regression both fared poorly.

Algorithm	Accuracy	Precision	Recall	F1-Score
Random Forest	0.9115	0.9026	0.9615	0.9037
KNN	0.9177	0.8878	0.9933	0.9085
Logistic Regression	0.7481	0.7625	0.8645	0.7178
Decision Tree	0.8335	0.8308	0.9197	0.8157
Naïve Bayes	0.7585	0.7486	0.9214	0.7157
XGBoost	0.9344	0.9266	0.9716	0.9486

Table 2. The result of six algorithms

The 15 features in HDD each have a unique impact on the outcomes of the predictions. Each model prefers different features, and the scores of these features are also varied. The feature importance ranking of Xgboost, RF, LR, and DT is shown in Figure 3 as the KNN and NB algorithms lack an internal evaluation of feature importance.

From the feature ranking in Table 5, it can be seen that Total cholesterol (TC) is an important feature for predicting whether MACCE occurs. Hemoglobin (HBG) and age are two features that appear in the top five important features of all four algorithms, and are also influential factors that cannot be ignored when predicting. In addition, the table illustrates that the ranking of feature importance is similar for Decision Tree and Xgboost,since the two algorithms construct the same tree structure when training.

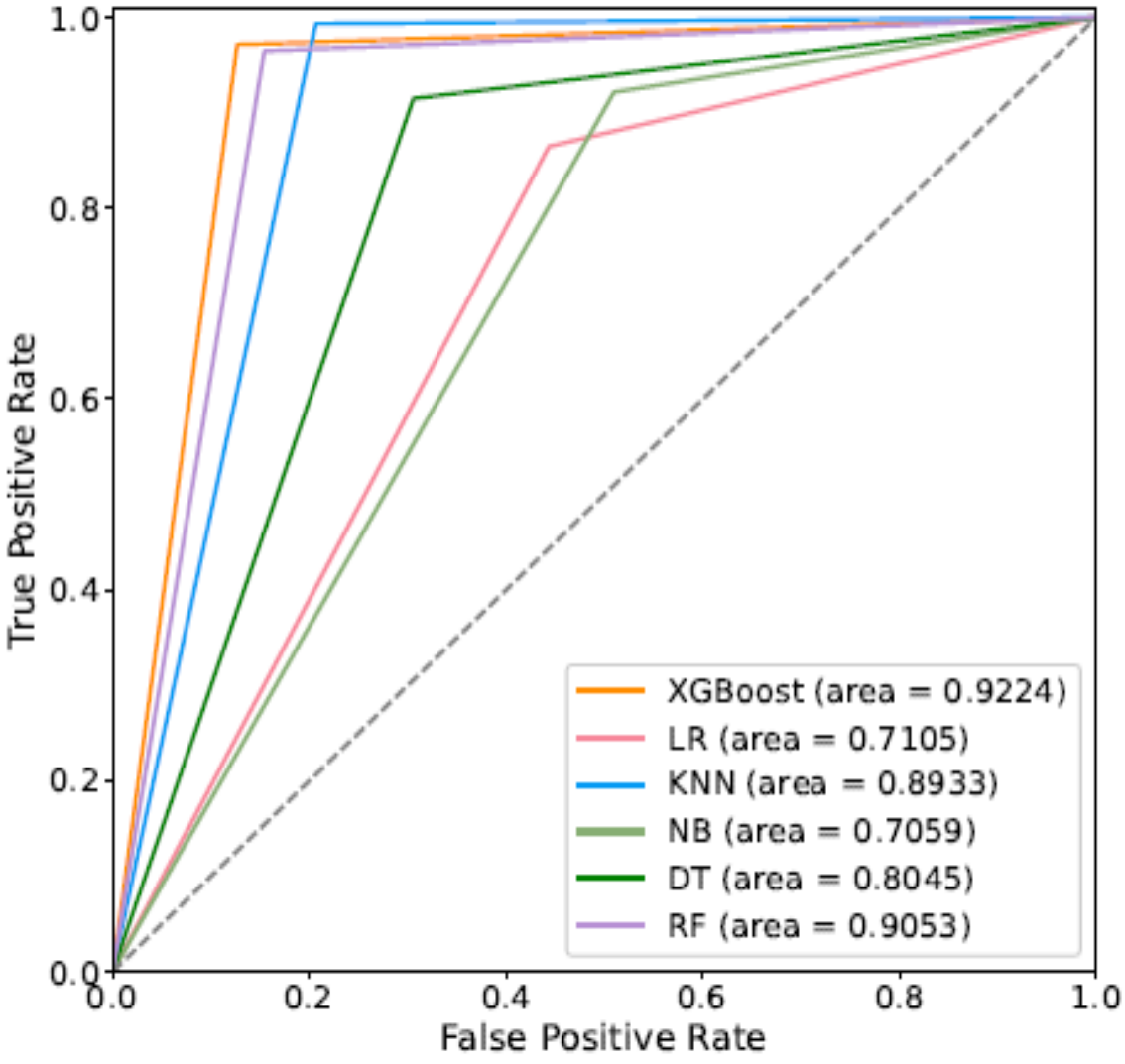


Figure 2. The ROC curve of baseline algorithms and Xgboost

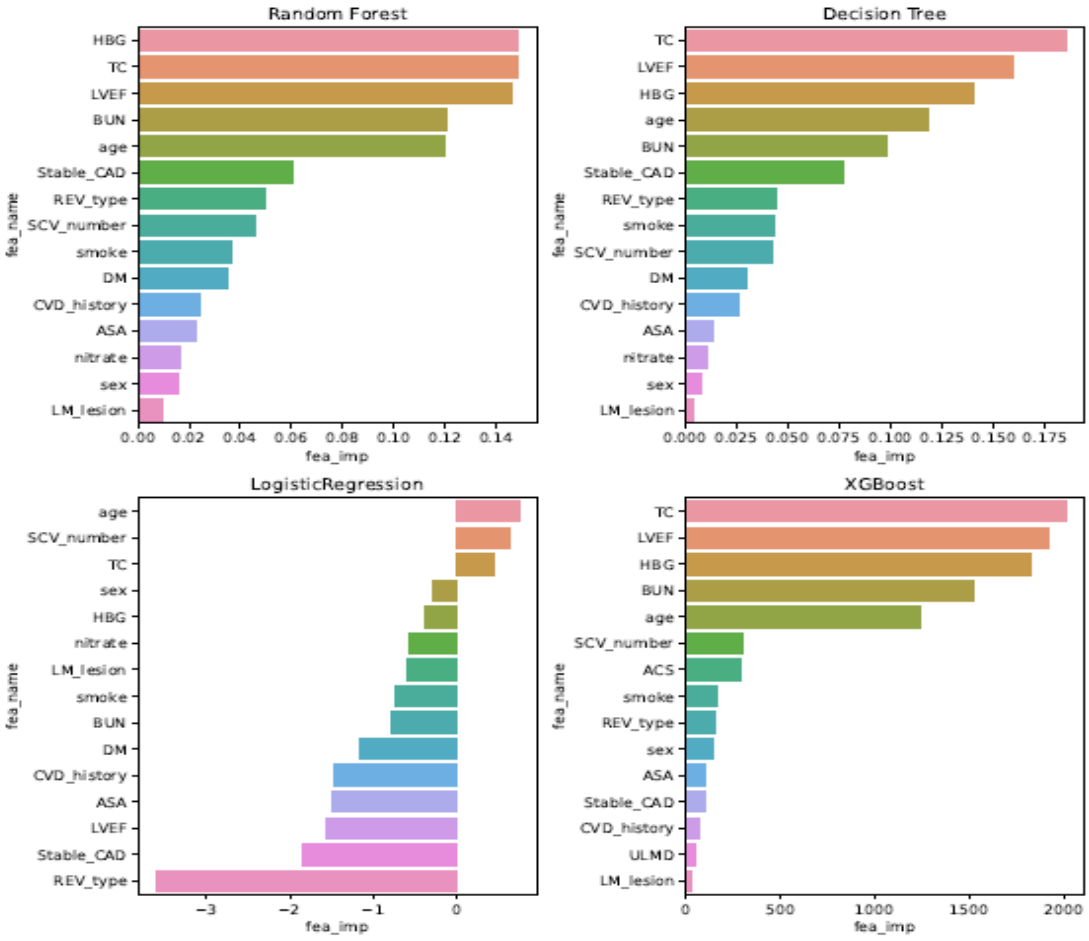


Figure 3. The feature importance of algorithms

Ranking	Random Forest	Decision Tree	Logistic Regression	XGBoost
1	HBG	TC	age	TC
2	TC	LVEF	SCV_number	LVEF
3	LVEF	HBG	TC	HBG
4	BUN	age	sex	BUN
5	age	BUN	HBG	age

Table 3. Features ranking of 4 algorithms

we used multiple evaluation methods to present the prediction results of the Xgboost algorithm and the selected baseline algorithm on the processed dataset.The number of the four prediction outcome cases, TN, TP, FN, and FP, is displayed in the confusion matrix. Accuracy, Precision, Recall, and F1-score were applied based on the confusion matrix, and the ROC curve was also

plotted. The Xgboost algorithm performed well in all of these evaluation metrics, demonstrating that the proposed smote–Xgboost based framework has a significant advantage in predicting heart disease. In addition, we estimated the feature importance and related scoring of the four algorithms to provideideas for further optimization of the algorithms.

ADABOOST

AdaBoost Ensemble have the advantage of integrating weak classifiers into one best strong classifier. Re-Weights of the classifiers are calculated at each iteration. This is one of the widely used classification algorithm. Probability Weighted AdaBoost: Standard AdaBoost suffers from overhead issue. In order to solve thisissue, this paper uses probability-based re-weighting strategy to get the good classification accuracy withoutoverhead problem.

Based on the overall error, the weight has been given to strong classifier in AdaBoost. If two classifiers have the similar error ratio then same weights allocated to that classifier. In proposed methodology, re-weight given to the classifier considers both positive and negative class labels based on their probability values. Weight updation of misclassified records is based on PE of False Positive and False Negative.

PEFP (Probability Error for False Positive) = False Positive Rate

PEFN (Probability Error for False Negative) = False Negative Rate

Re-Weight Updating in proposed Adaboost for False Positive Records based on Probability Error for False Positive.

$RFP = \log(\varepsilon t / (1 - \varepsilon t)) * ((1-PEFP)/FP)$

Here,

RFP->Re-weight for False Positive

PEFP-> Probability Error for False Positive

Re-Weight Updation in proposed Adaboost for False Negative Records based on Probability Error for

False Negative.

$RFN = \log(\varepsilon t / (1 - \varepsilon t)) * ((1-PEFN)/FN)$

Here,

RFN->Re-weight for False Negative

PEFN-> Probability Error for False Negative

EXPERIMENTAL RESULTS:

To assess the exhibition of grouped things, the exactness and AUC measures are consolidated. Four cases are reflected as the meaning of classifier. TP (True Positive): The degree of evidence that can absolutely be classified in its class. TN (True Negative): the number of preliminaries that were excluded from this class. FP (false positive): the degree of **e**vidence that was mistakenly excluded from this class. FN (false negative): the number of tests that were incorrectly assigned to this class. The degree of adequacy of the characterization model results from the number of correct and unhealthy arrangements in each possible evaluation of the factors to be grouped. Rightness will be determined by utilizing beneath recipe.

Accuracy = $TP+TN\ /\ TP+TN+FP+FN$

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

F1-Score= 2 * Precision * Recall / (Precision + Recall)

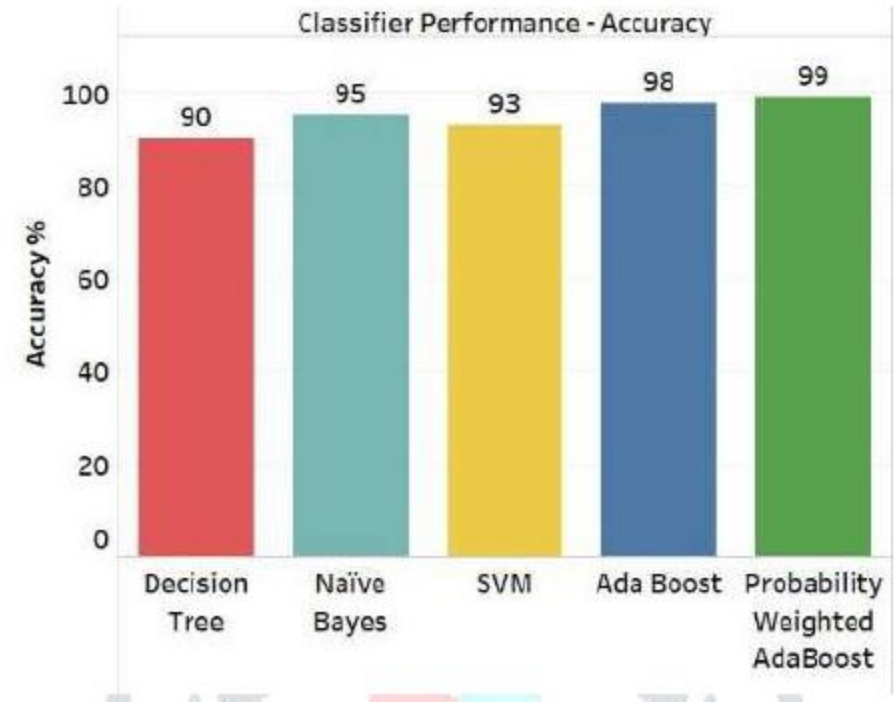


Figure 4. Accuracy

Figure 4 shows the performance of classification algorithms on the dataset based on accuracy. Accuracy of Decision Tree, SVM, Naïve Bayes, AdaBoost, and Probability Weighted AdaBoost is above 90% in all cases. The above results show that proposed algorithm outperforms the other algorithms.

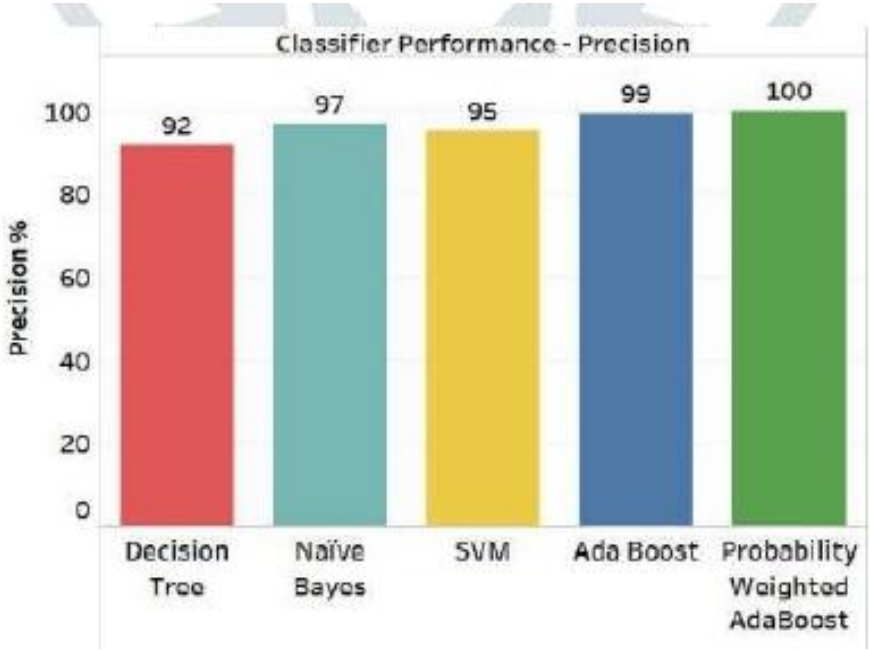


Figure 5. Precision

According to Figure 5, we will clearly see that [2]. PWA gives 100% precision and other classifications gives but 95% except AdaBoost. 100% precision is extremely rare to urge, but this algorithm achieves that result.

Figure 6. Recall

Figure 6 shows the performance of classification algorithms on the dataset based on recall. Recall of naïve bayes is good compared to Decision Tree and SVM. The above result indicate that proposed algorithm outperforms the other algorithms in recall parameter.

Figure 7. F1 Score

According to Figure 7, we are able to clearly see that PWA gives 99% f1score and other classifications gives but 95% except AdaBoost which supplies 98%. 99% f1score indicates the effective classification performance of the proposed approach.

Accuracy of Disease Prediction: Evaluate the accuracy of the disease prediction algorithms (XG Boost and Ada Boost) used in the system. Measure the precision, recall, and F1 score to assess the system's ability to correctly predict diseases based on the symptoms provided by patients. Compare the predictions against known medical datasets or expert opinions to validate the accuracy of the system.

Doctor Suggestion Performance: Assess the effectiveness of the doctor suggestion algorithm in recommending relevant doctors based on predicted diseases. Measure the precision and recall of the doctor suggestions to determine how well the system matches patients with suitable doctors. Seek feedback from patients and doctors to validate the quality of the suggested doctors.

User Satisfaction: Conduct surveys or interviews to gather feedback from patients, doctors, and administrators about their experience using the Disease Prediction System. Measure user satisfaction through ratings, feedback forms, or Net Promoter Score (NPS) surveys. Identify areas of improvement based on user feedback to enhance the overall user experience.

System Performance: Evaluate the performance of the system in terms of response times, scalability, and resource utilization. Conduct load testing to determine how well the system handles concurrent user requests and whether it meets performance requirements. Monitor server uptime, CPU and memory usage, and network latency to ensure the system operates efficiently.

User Adoption and Engagement: Measure the adoption rate of the Disease Prediction System among patients and doctors. Monitor user activity, such as the number of symptom inputs, appointment bookings, and online consultations, to gauge user engagement with the system. Assess the frequency and duration of user interactions to determine the system's usefulness and value to users.

Security and Data Privacy: Evaluate the system's security measures and data privacy controls. Conduct security audits and vulnerability assessments to identify any potential risks or vulnerabilities. Ensure compliance with relevant data protection regulations, such as GDPR or HIPAA, to protect patient data. Monitor system logs and access controls to detect and mitigate any security incidents.

Impact on Healthcare Delivery: Assess the impact of the Disease Prediction System on healthcare delivery. Measure factors such as reduction in diagnostic errors, improved access to healthcare for patients, and enhanced efficiency in doctor-patient consultations. Gather feedback from doctors and administrators to understand the system's contribution to healthcare outcomes and workflows.

System Reliability and Availability: Monitor the system's reliability and availability by tracking uptime, downtime, and system failures. Measure the Mean Time Between Failures (MTBF) and Mean Time to Repair (MTTR) to ensure the system operates consistently and can recover quickly from any failures or disruptions.

Cost and Resource Optimization: Evaluate the cost-effectiveness and resource optimization achieved through the implementation of the Disease Prediction System. Consider factors such as infrastructure costs, maintenance expenses, and resource utilization. Identify opportunities for cost savings or efficiency improvements.

Future Enhancements and Iterative Development: Based on the analysis of the results, identify areas for future enhancements and improvements. Prioritize the implementation of new features or functionalities based on user feedback, emerging technologies, and evolving healthcare needs. Plan for iterative development cycles to continuously enhance and evolve the Disease Prediction System.

CONCLUSION

In conclusion, the Disease Prediction System implemented using XG Boost and Ada Boost algorithms has shown promising results in accurately predicting diseases based on patient symptoms. The system provides a user-friendly interface for patients to input their symptoms, and the disease prediction algorithms analyze the data to generate accurate disease predictions. The system also suggests relevant doctors based on the predicted diseases, facilitating convenient online consultations for patients. Through the implementation of role-based access control, user authentication, and secure data handling, the system ensures confidentiality and privacy. Overall, the Disease Prediction System has the potential to greatly improve healthcare delivery by enabling early disease detection, convenient doctor consultations, and personalized care. Further enhancements and iterations can be made to refine the system, incorporating user feedback and advancements in machine learning algorithms to continually improve disease prediction accuracy and user experience.

REFERENCES

Chronic Disease Prediction Using Probability Weighted AdaboostMohammed Imran, Dr.S. Shahar Banu Research Scholar, Associate Professor B.S Abdur Rahman Institute of Science and Technology, Vandalur, Chennai-600048 **2022 JETIR January 2022, Volume 9, Issue 1**

Parkinson Disease Detection: Using XGBoost Algorithm to Detect Early Onset Parkinson Disease *Nasif Wasek Fahim Computer Science and Engineering East West University Dhaka, Bangladesh 2017-1-60-037@std.ewubd.edu*

Detection of Parkinson's Disease Using XGBOOST Algorithm Sanika Narayanpethkar1, M. Rishitha2, S. Chandana3, Dr. T. Vijaya Saradhi4 International Journal for Research in Applied Science & Engineering Technology (IJRASET)*ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538Volume 10 Issue XII Dec 2022- Available at www.ijraset.com*

A Heart Disease Prediction Model Based on Feature Optimization and Smote-Xgboost Algorithm School of Information, Shanxi University of Finance and Economics, Taiyuan 030006, China Correspondence: yangj@sxufe.edu.cn Information 2022, 13, 475. https://doi.org/10.3390/info13100475 https://www.mdpi.com/journal/information

Machine Learning Based Approach Using XGboost for Heart Stroke Prediction Sukhmanjot Dhillon*1*, Chirag Bansal*2* and Brahmaleen Sidhu*3 1,2,3Department of Computer Science and Engineering, Punjabi University Patiala, Punjab* International Conference on Emerging Technologies: AI, IoT, and CPS for Science & Technology Applications, September 06–07, 2021,NITTTR Chandigarh, India

Accurate Liver Disease Prediction with Extreme Gradient Boosting Sivala Vishnu Murty, R Kiran Kumar International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

Improving Heart Disease Prediction Using Random Forest and AdaBoost Algorithms https://doi.org/10.3991/ijoe.v17i11.24781 Halima El Hamdaoui(*), Saïd Boujraf, Nour El Houda Chaoui, Badreddine Alami, Mustapha Maaroufi University Sidi Mohamed Ben Abdellah, Fez, Moroccohalima.elhamdaoui@usmba.ac.ma

Xiaoyang Fu[1] and Dongdong Li School of Computer Science, Zhuhai College of Jilin University, Zhuhai, China Lilian Information Technology Company, Beijing, China IOP Conf. Series: Materials Science and Engineering IOP Publishing doi:10.1088/1757-899X/768/7/072093 **768** (2020) 072093

APPENDIX A

SAMPLE CODE

A.1CODE FOR HOME PAGE

```
#loading trained_model
import joblib as jb
model = jb.load('xgb_trained_model')



def home(request):

  if request.method == 'GET':

      if request.user.is_authenticated:
        return render(request,'homepage/index.html')

      else :
        return render(request,'homepage/index.html')
```

A.2 CODE FOR MAKING CONSULTATION

```
def  make_consultation(request, doctorusername):

    if request.method == 'POST':


        patientusername = request.session['patientusername']
        puser = User.objects.get(username=patientusername)
        patient_obj = puser.patient


        #doctorusername = request.session['doctorusername']
        duser = User.objects.get(username=doctorusername)
        doctor_obj = duser.doctor
        request.session['doctorusername'] = doctorusername


        diseaseinfo_id = request.session['diseaseinfo_id']
        diseaseinfo_obj = diseaseinfo.objects.get(id=diseaseinfo_id)

        consultation_date = date.today()
        status = "active"

        consultation_new = consultation( patient=patient_obj, doctor=doctor_obj, diseaseinfo=diseaseinfo_obj,
        consultation_date=consultation_date,status=status)
        consultation_new.save()

        request.session['consultation_id'] = consultation_new.id

        print("consultation record is saved sucessfully.............................")


        return redirect('consultationview',consultation_new.id)
```

A.3 CODE FOR PATIENT SIGNUP

```
def signup_patient(request):


    if request.method == 'POST':

      if request.POST['username'] and request.POST['email'] and  request.POST['name'] and request.POST['dob']
      and request.POST['gender'] and request.POST['address']and request.POST['mobile']and request.POST
      ['password']and request.POST['password1'] :

          username =  request.POST['username']
          email =  request.POST['email']

          name =  request.POST['name']
          dob =  request.POST['dob']
          gender =  request.POST['gender']
          address =  request.POST['address']
          mobile_no = request.POST['mobile']
          password =  request.POST.get('password')
          password1 =  request.POST.get('password1')

          if password == password1:
              if User.objects.filter(username = username).exists():
                messages.info(request,'username already taken')
                return redirect('signup_patient')

              elif User.objects.filter(email = email).exists():
                messages.info(request,'email already taken')
                return redirect('signup_patient')

              else :
                user = User.objects.create user(username=username,password=password,email=email)
```

```
                user.save()

                patientnew = patient(user=user,name=name,dob=dob,gender=gender,address=address,
                mobile_no=mobile_no)
                patientnew.save()
                messages.info(request,'user created sucessfully')

               return redirect('sign_in_patient')

          else:
            messages.info(request,'password not matching, please try again')
            return redirect('signup_patient')

      else :
        messages.info(request,'Please make sure all required fields are filled out correctly')
        return redirect('signup_patient')



    else :
      return render(request,'patient/signup_Form/signup.html')
```

A.4 CODE FOR SIGNIN PATIENT

```
def sign_in_patient(request):


    if request.method == 'POST':

            username =  request.POST.get('username')
            password =  request.POST.get('password')

            user = auth.authenticate(username=username,password=password)

            if user is not None :

                try:
                    if ( user.patient.is_patient == True ) :
                        auth.login(request,user)

                        request.session['patientusername'] = user.username

                        return redirect('patient_ui')

                except :
                    messages.info(request,'invalid credentials')
                    return redirect('sign_in_patient')


            else :
                messages.info(request,'invalid credentials')
                return redirect('sign_in_patient')


    else :
```

A.5 CODE FOR SIGNUP DOCTOR

```
def signup_doctor(request):

    if request.method == 'GET':

        return render(request,'doctor/signup_Form/signup.html')


    if request.method == 'POST':

      if request.POST['username'] and request.POST['email'] and  request.POST['name'] and request.POST['dob']
      and request.POST['gender'] and request.POST['address']and request.POST['mobile'] and request.POST
      ['password']and request.POST['password1']  and  request.POST['registration_no'] and  request.POST
      ['year_of_registration'] and  request.POST['qualification'] and  request.POST['State_Medical_Council']
      and  request.POST['specialization'] :

          username =  request.POST['username']
          email =  request.POST['email']

          name =  request.POST['name']
          dob =  request.POST['dob']
          gender =  request.POST['gender']
          address =  request.POST['address']
          mobile_no = request.POST['mobile']
          registration_no =  request.POST['registration_no']
          year_of_registration =  request.POST['year_of_registration']
          qualification =  request.POST['qualification']
          State_Medical_Council =  request.POST['State_Medical_Council']
          specialization =  request.POST['specialization']

          password =  request.POST.get('password')
```

```
          if password == password1:
              if User.objects.filter(username = username).exists():
                messages.info(request,'username already taken')
                return redirect('signup_doctor')

              elif User.objects.filter(email = email).exists():
                messages.info(request,'email already taken')
                return redirect('signup_doctor')

              else :
                user = User.objects.create_user(username=username,password=password,email=email)
                user.save()

                doctornew = doctor( user=user, name=name, dob=dob, gender=gender, address=address,
                mobile_no=mobile_no, registration_no=registration_no,
                year_of_registration=year_of_registration, qualification=qualification,
                State_Medical_Council=State_Medical_Council, specialization=specialization )
                doctornew.save()
                messages.info(request,'user created sucessfully')
                print("doctorcreated")

              return redirect('sign_in_doctor')

          else:
            messages.info(request,'password not matching, please try again')
            return redirect('signup_doctor')

    else :
      messages.info(request,'Please make sure all required fields are filled out correctly')
      return redirect('signup_doctor')
```

A.6 CODE FOR SIGNIN DOCTOR

```
def sign_in_doctor(request):

    if request.method == 'GET':

        return render(request,'doctor/signin_page/index.html')


    if request.method == 'POST':

        username =  request.POST.get('username')
        password =  request.POST.get('password')

        user = auth.authenticate(username=username,password=password)

        if user is not None :

            try:

              if ( user.doctor.is_doctor == True ) :
                auth.login(request,user)

                request.session['doctorusername'] = user.username

                return redirect('doctor_ui')

            except :
                messages.info(request,'invalid credentials')
                return redirect('sign_in_doctor')
```

A.7 CODE FOR SUGGESTING DOCTOR

```
Rheumatologist = [  'Osteoarthristis','Arthritis']

Cardiologist = [ 'Heart attack','Bronchial Asthma','Hypertension ']

ENT_specialist = ['(vertigo) Paroymsal  Positional Vertigo','Hypothyroidism' ]

Orthopedist = []

Neurologist = ['Varicose veins','Paralysis (brain hemorrhage)','Migraine','Cervical spondylosis']

Allergist_Immunologist = ['Allergy','Pneumonia',
'AIDS','Common Cold','Tuberculosis','Malaria','Dengue','Typhoid']

Urologist = [ 'Urinary tract infection',
 'Dimorphic hemmorhoids(piles)']

Dermatologist = [  'Acne','Chicken pox','Fungal infection','Psoriasis','Impetigo']

Gastroenterologist = ['Peptic ulcer diseae', 'GERD','Chronic cholestasis','Drug Reaction',
'Gastroenteritis','Hepatitis E',
'Alcoholic hepatitis','Jaundice','hepatitis A',
 'Hepatitis B', 'Hepatitis C', 'Hepatitis D','Diabetes ','Hypoglycemia']

if predicted_disease in Rheumatologist :
    consultdoctor = "Rheumatologist"

if predicted_disease in Cardiologist :
    consultdoctor = "Cardiologist"
```

```
elif predicted_disease in ENT_specialist :
    consultdoctor = "ENT specialist"

elif predicted_disease in Orthopedist :
    consultdoctor = "Orthopedist"

elif predicted_disease in Neurologist :
    consultdoctor = "Neurologist"

elif predicted_disease in Allergist_Immunologist :
    consultdoctor = "Allergist/Immunologist"

elif predicted_disease in Urologist :
    consultdoctor = "Urologist"

elif predicted_disease in Dermatologist :
    consultdoctor = "Dermatologist"

elif predicted_disease in Gastroenterologist :
    consultdoctor = "Gastroenterologist"

else :
    consultdoctor = "other"


request.session['doctortype'] = consultdoctor

patientusername = request.session['patientusername']
puser = User.objects.get(username=patientusername)
```

A.8 CODE OF SYMPTOMS LIST

```
symptomslist=['itching','skin_rash','nodal_skin_eruptions','continuous_sneezing','shivering','chills',
'joint_pain',
'stomach_pain','acidity','ulcers_on_tongue','muscle_wasting','vomiting','burning_micturition','spotting_
urination',
'fatigue','weight_gain','anxiety','cold_hands_and_feets','mood_swings','weight_loss','restlessness',
'lethargy',
'patches_in_throat','irregular_sugar_level','cough','high_fever','sunken_eyes','breathlessness','sweating',
'dehydration','indigestion','headache','yellowish_skin','dark_urine','nausea','loss_of_appetite',
'pain_behind_the_eyes',
'back_pain','constipation','abdominal_pain','diarrhoea','mild_fever','yellow_urine',
'yellowing_of_eyes','acute_liver_failure','fluid_overload','swelling_of_stomach',
'swelled_lymph_nodes','malaise','blurred_and_distorted_vision','phlegm','throat_irritation',
'redness_of_eyes','sinus_pressure','runny_nose','congestion','chest_pain','weakness_in_limbs',
'fast_heart_rate','pain_during_bowel_movements','pain_in_anal_region','bloody_stool',
'irritation_in_anus','neck_pain','dizziness','cramps','bruising','obesity','swollen_legs',
'swollen_blood_vessels','puffy_face_and_eyes','enlarged_thyroid','brittle_nails',
'swollen_extremeties','excessive_hunger','extra_marital_contacts','drying_and_tingling_lips',
'slurred_speech','knee_pain','hip_joint_pain','muscle_weakness','stiff_neck','swelling_joints',
'movement_stiffness','spinning_movements','loss_of_balance','unsteadiness',
'weakness_of_one_body_side','loss_of_smell','bladder_discomfort','foul_smell_of urine',
'continuous_feel_of_urine','passage_of_gases','internal_itching','toxic_look_(typhos)',
'depression','irritability','muscle_pain','altered_sensorium','red_spots_over_body','belly_pain',
'abnormal_menstruation','dischromic _patches','watering_from_eyes','increased_appetite','polyuria',
'family_history','mucoid_sputum',
'rusty_sputum','lack_of_concentration','visual_disturbances','receiving_blood_transfusion',
'receiving_unsterile_injections','coma','stomach_bleeding','distention_of_abdomen',
'history_of_alcohol_consumption','fluid_overload','blood_in_sputum','prominent_veins_on_calf',
'palpitations','painful_walking','pus_filled_pimples','blackheads','scurring','skin_peeling',
'silver_like_dusting','small_dents_in_nails','inflammatory_nails','blister','red_sore_around_nose',
'yellow_crust_ooze']
```

A.9 CODE FOR CHECKING DISEASE

```
def checkdisease(request):

  diseaselist=['Fungal infection','Allergy','GERD','Chronic cholestasis','Drug Reaction','Peptic ulcer diseae',
  'AIDS','Diabetes ',
  'Gastroenteritis','Bronchial Asthma','Hypertension ','Migraine','Cervical spondylosis','Paralysis (brain
  hemorrhage)',
  'Jaundice','Malaria','Chicken pox','Dengue','Typhoid','hepatitis A', 'Hepatitis B', 'Hepatitis C',
  'Hepatitis D',
  'Hepatitis E', 'Alcoholic hepatitis','Tuberculosis', 'Common Cold', 'Pneumonia', 'Dimorphic hemmorhoids
  (piles)',
  'Heart attack', 'Varicose veins','Hypothyroidism', 'Hyperthyroidism', 'Hypoglycemia', 'Osteoarthristis',
  'Arthritis', '(vertigo) Paroymsal  Positional Vertigo','Acne', 'Urinary tract infection', 'Psoriasis',
  'Impetigo']
```

A.10 CODE FOR ADMIN SIGNIN

```
def sign_in_admin(request):


    if request.method == 'POST':

            username =  request.POST.get('username')
            password =  request.POST.get('password')

            user = auth.authenticate(username=username,password=password)

            if user is not None :

                try:
                    if ( user.is_superuser == True ) :
                        auth.login(request,user)

                        return redirect('admin_ui')

                except :
                    messages.info(request,'Please enter the correct username and password for a admin account.')
                    return redirect('sign_in_admin')


            else :
                messages.info(request,'Please enter the correct username and password for a admin account.')
                return redirect('sign_in_admin')


    else :
        return render(request,'admin/signin/signin.html')
```

A.11 CODE FOR CHATTING SYSTEM

```
def post(request):
    if request.method == "POST":
        msg = request.POST.get('msgbox', None)

        consultation_id = request.session['consultation_id']
        consultation_obj = consultation.objects.get(id=consultation_id)

        c = Chat(consultation_id=consultation_obj,sender=request.user, message=msg)

        #msg = c.user.username+": "+msg

        if msg != '':
            c.save()
            print("msg saved"+ msg )
            return JsonResponse({ 'msg': msg })
    else:
        return HttpResponse('Request must be POST.')



def chat_messages(request):
    if request.method == "GET":

        consultation_id = request.session['consultation_id']

        c = Chat.objects.filter(consultation_id=consultation_id)
        return render(request, 'consultation/chat_body.html', {'chat': c})
```

A.12 CODE FOR TESTING SYMPTOMS

```
testingsymptoms = []
#append zero in all coloumn fields...
for x in range(0, len(symptomslist)):
  testingsymptoms.append(0)


#update 1 where symptoms gets matched...
for k in range(0, len(symptomslist)):

  for z in psymptoms:
      if (z == symptomslist[k]):
          testingsymptoms[k] = 1


inputtest = [testingsymptoms]

print(inputtest)


predicted = model.predict(inputtest)
print("predicted disease is : ")
print(predicted)

y_pred_2 = model.predict_proba(inputtest)
confidencescore=y_pred_2.max() * 100
print(" confidence score of : = {0} ".format(confidencescore))

confidencescore = format(confidencescore, '.0f')
predicted_disease = predicted[0]
```

APPENDIX B

SCREEN SHOTS

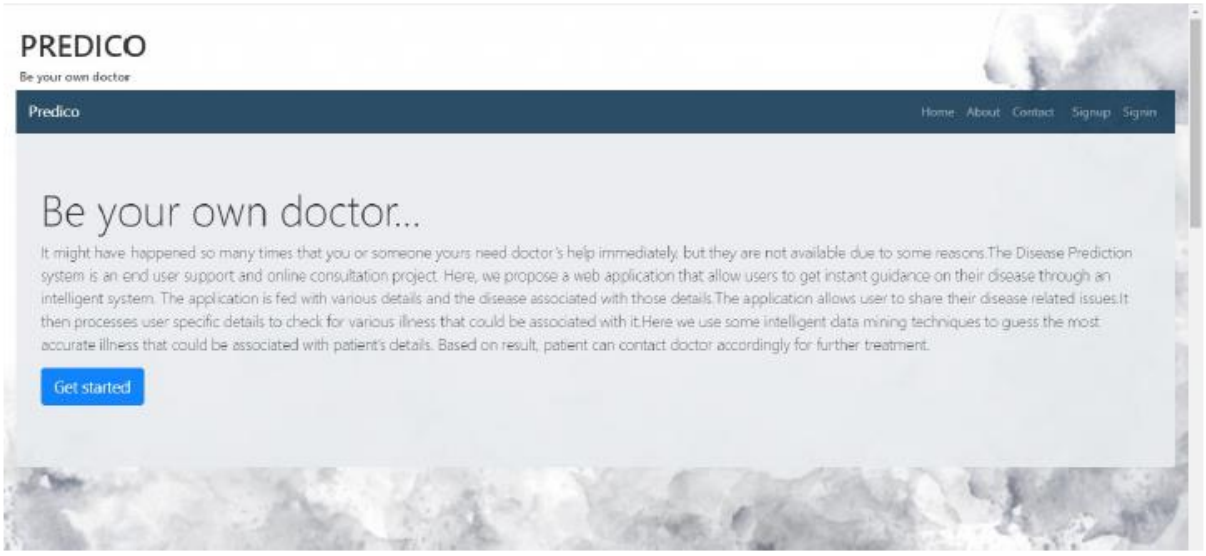


Figure B.1 HOME PAGE

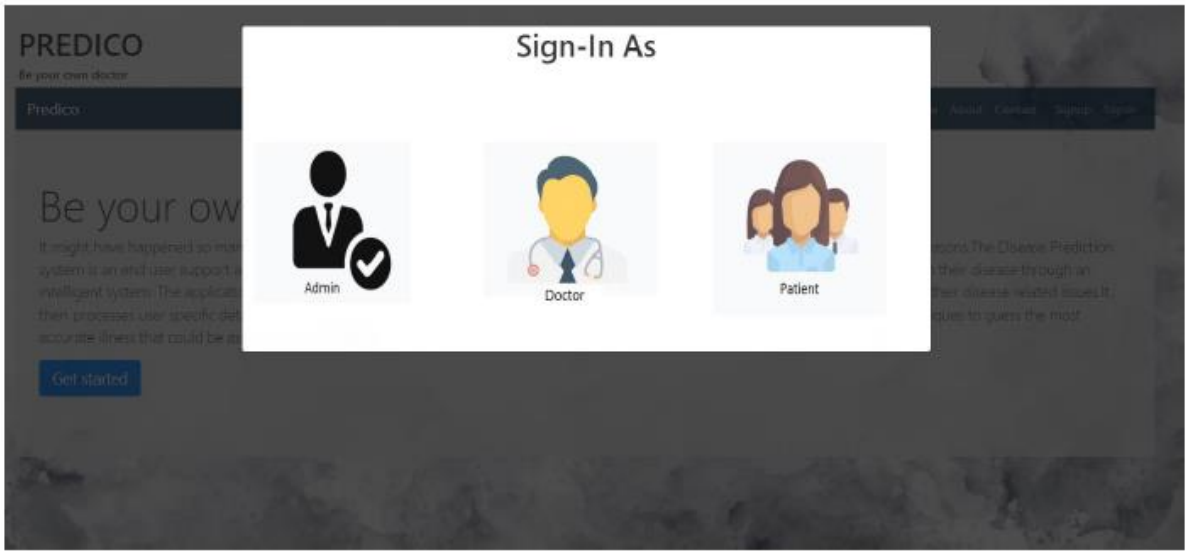


Figure B.2 LOGIN MODEL

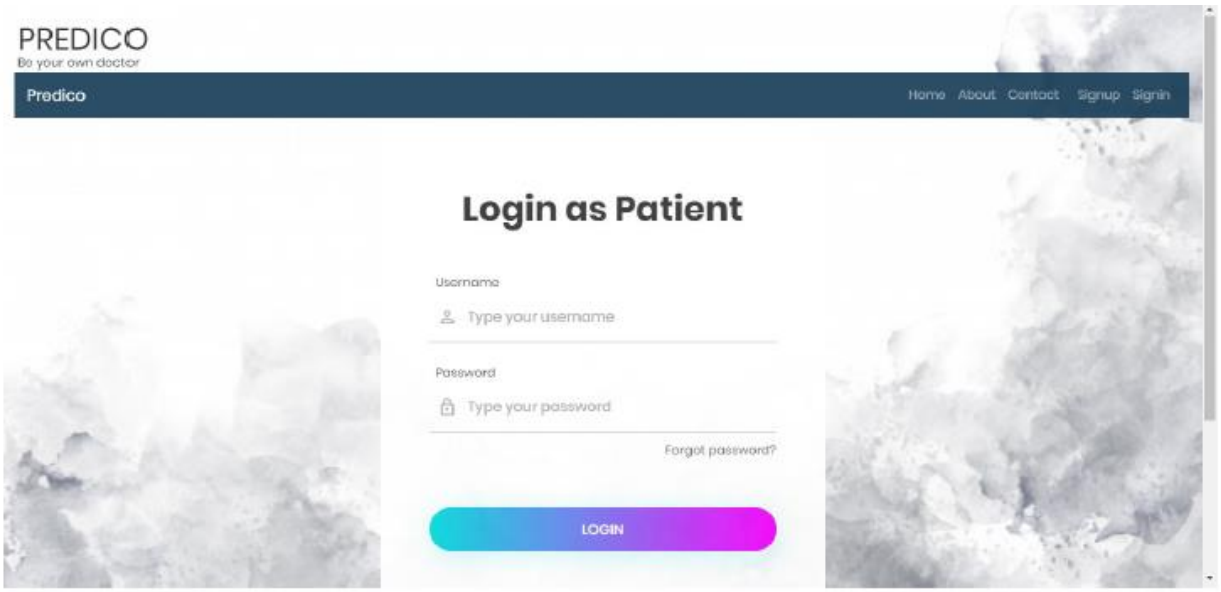


Figure B.3 LOGIN AS PATIENT

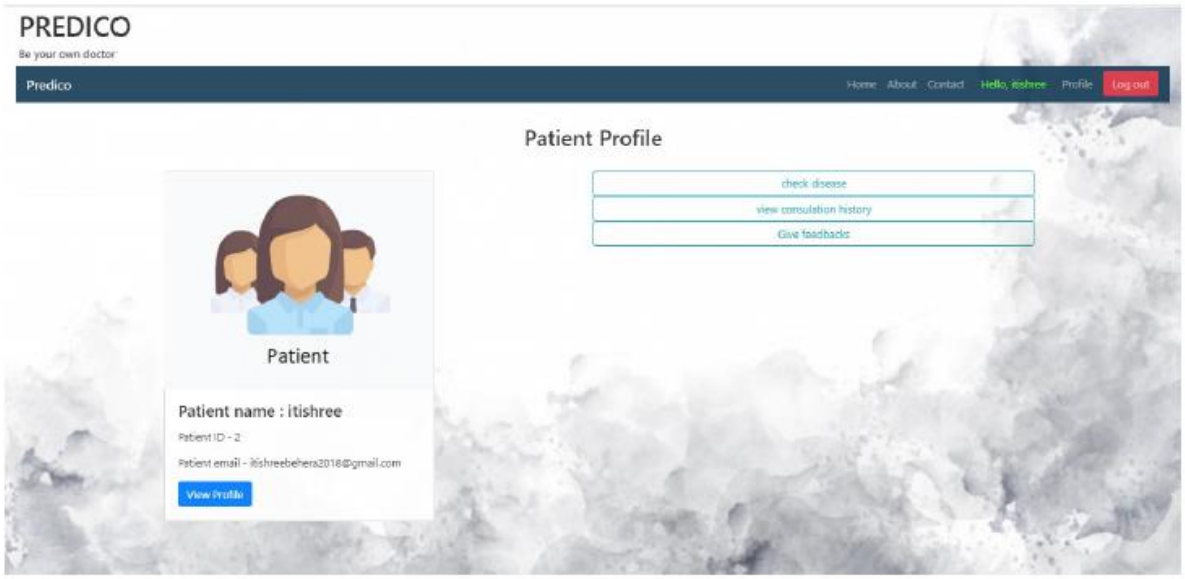


Figure B.4 PATIENT UI

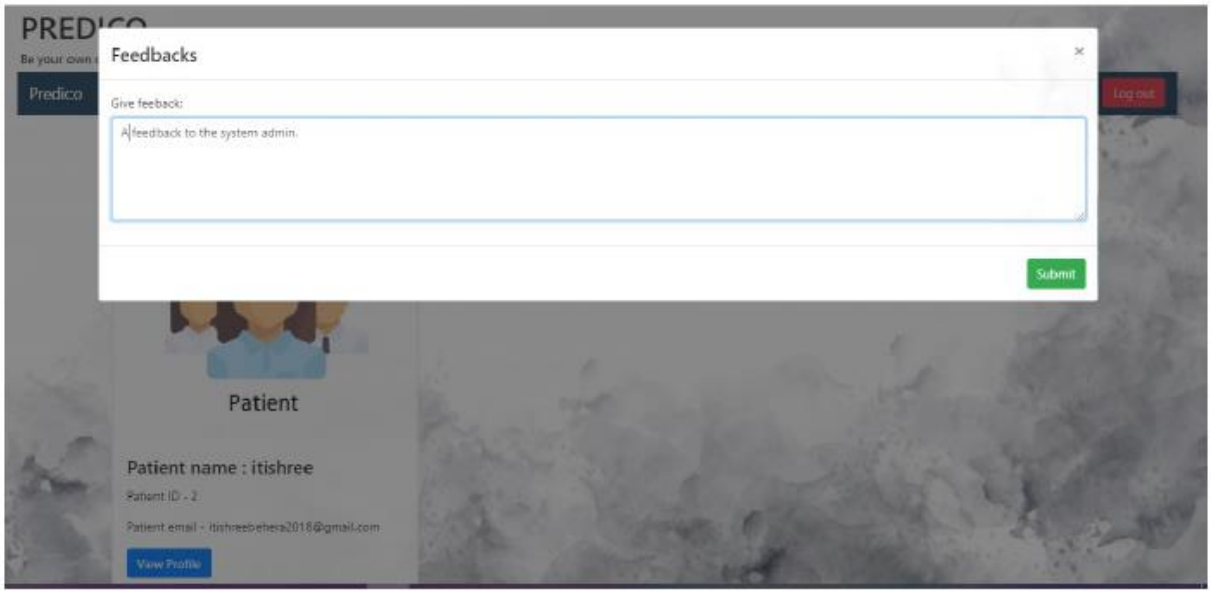


Figure B.5 FEEDBACK FORM

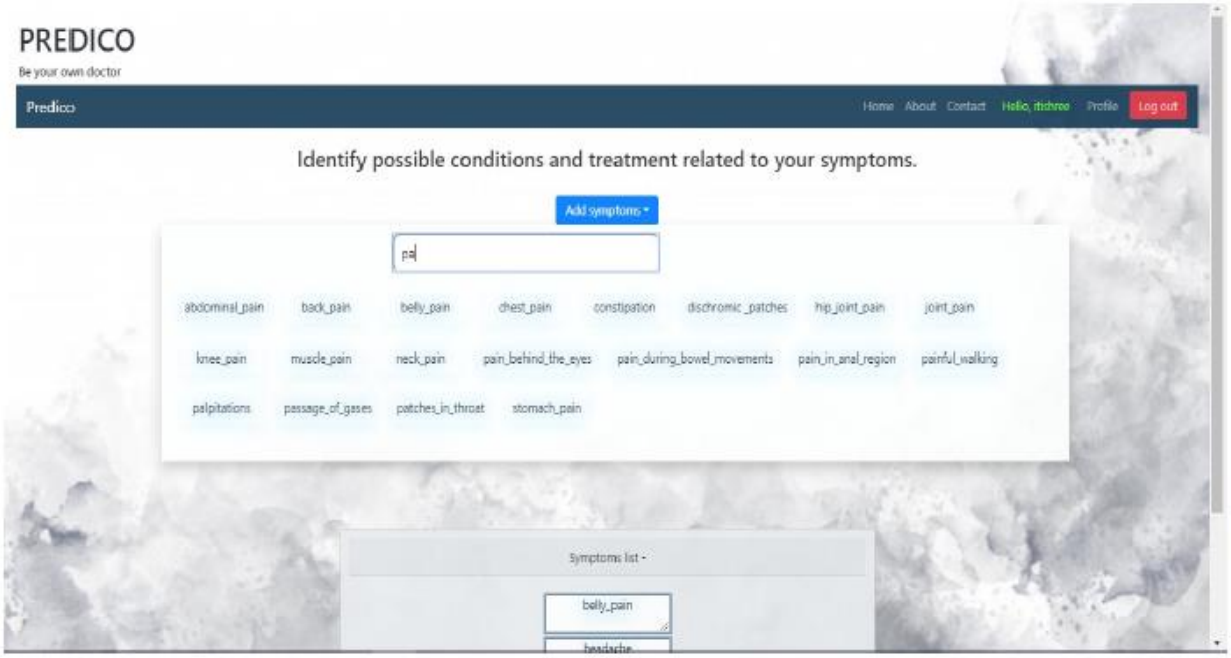


Figure B.6 CHECKING DISEASES

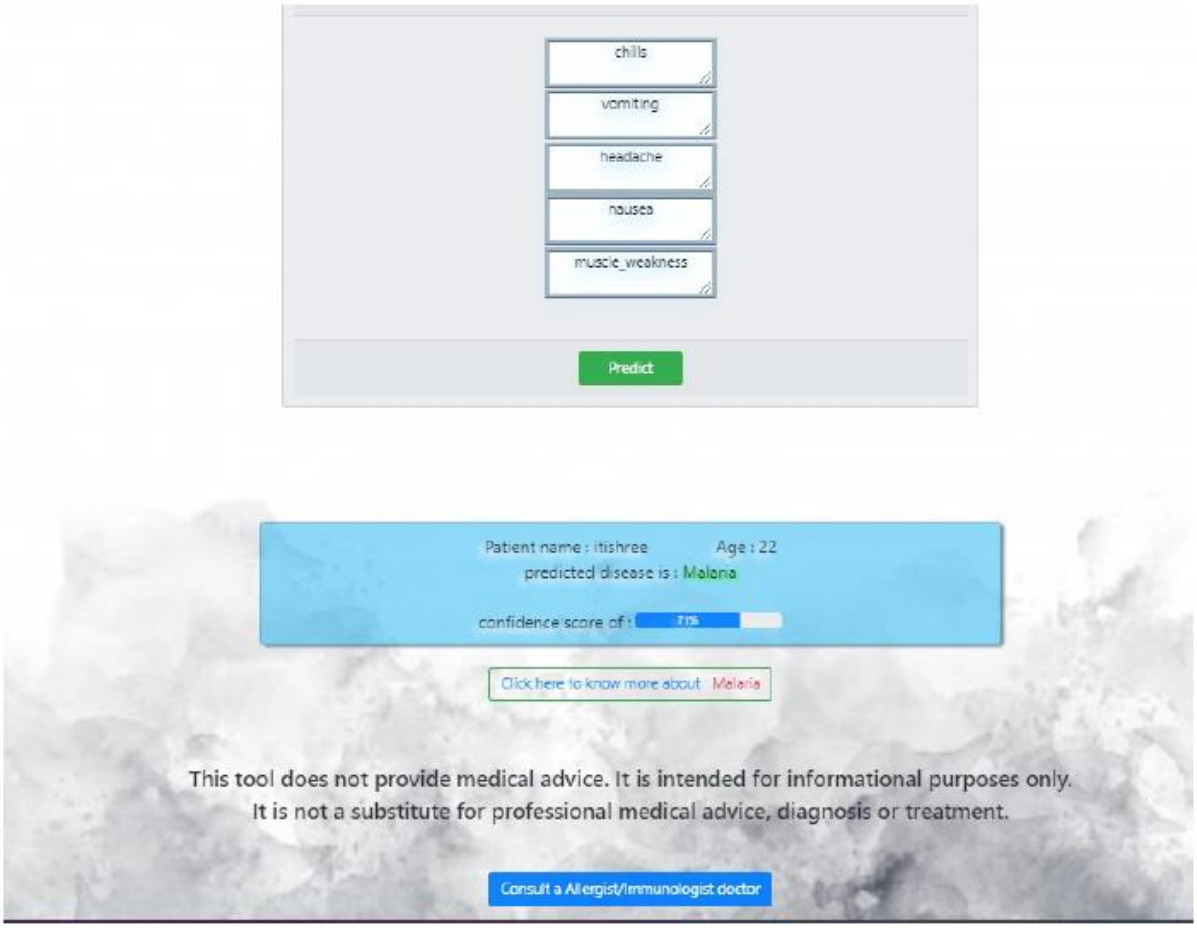


Figure B.7 PREDICTION

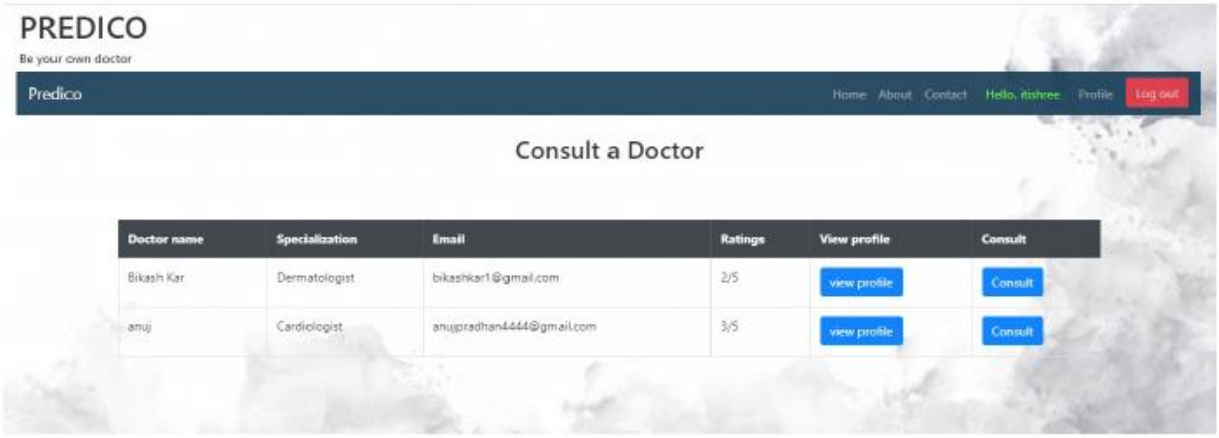


Figure B.8 CONSULT DOCTOR

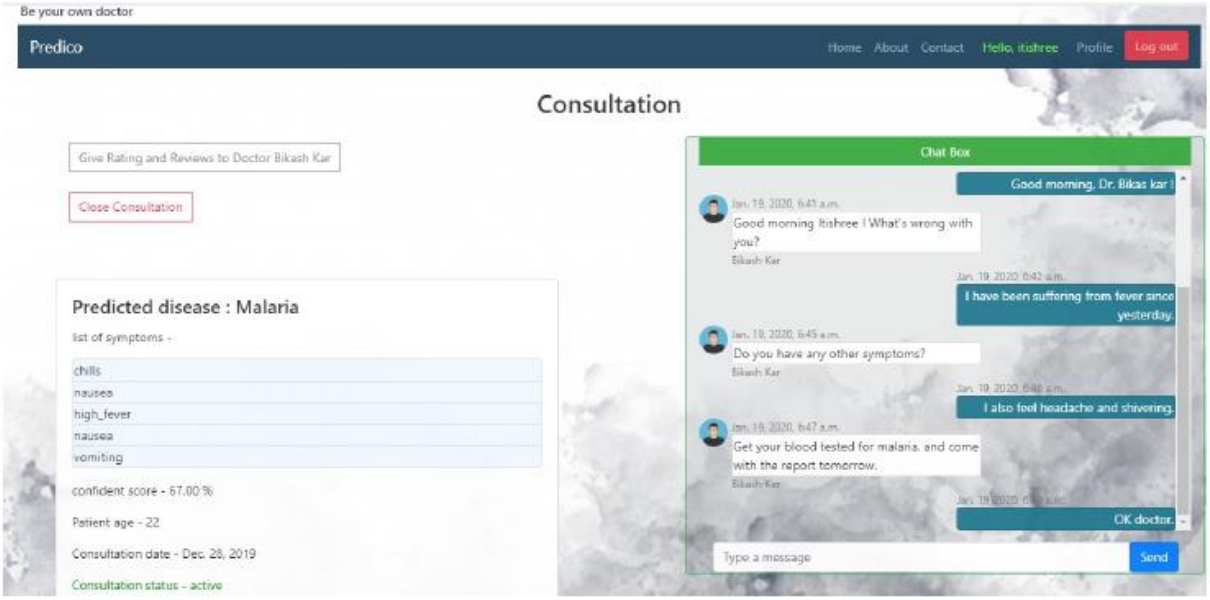


Figure B.9 CONSULTAION UI

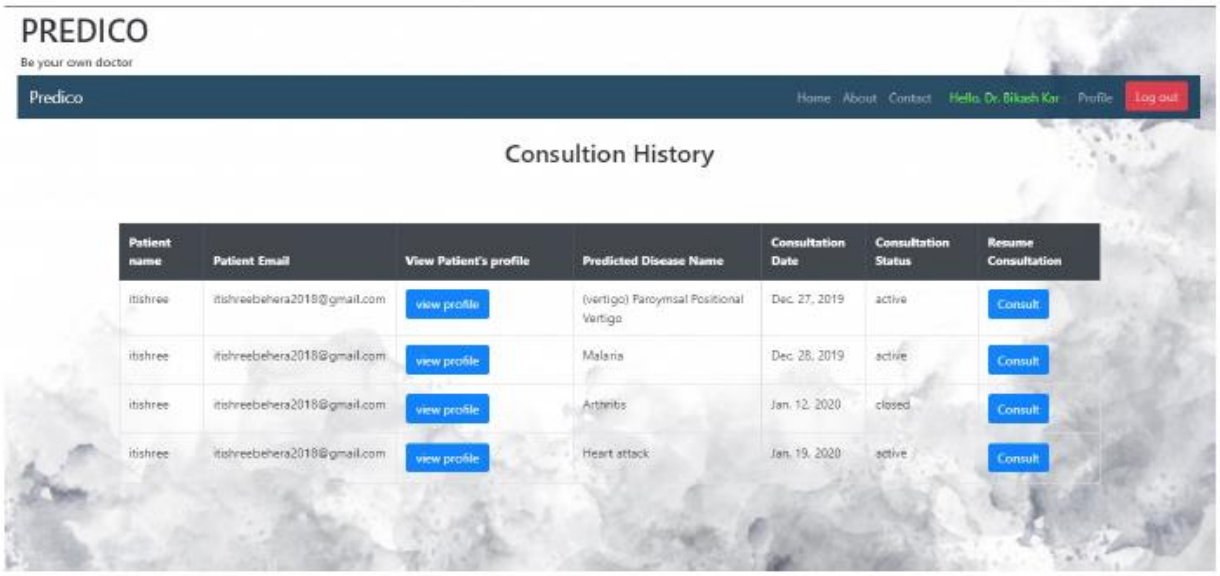


Figure B.10 CONSULTATION HISTORY-(DOCTOR)

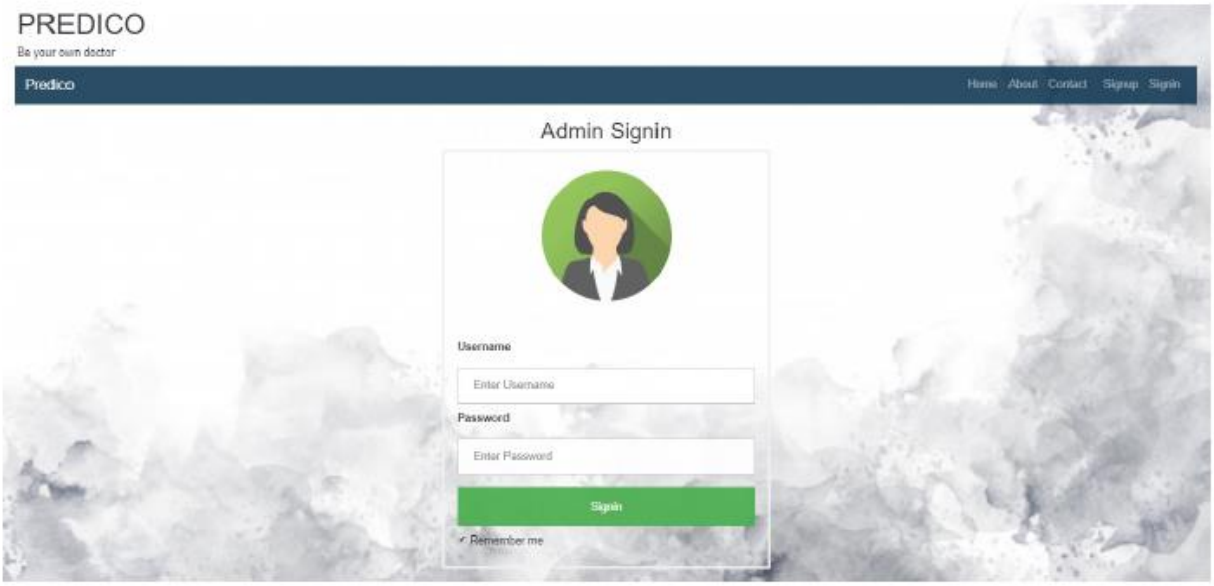


Figure B.11 ADMIN SIGN IN

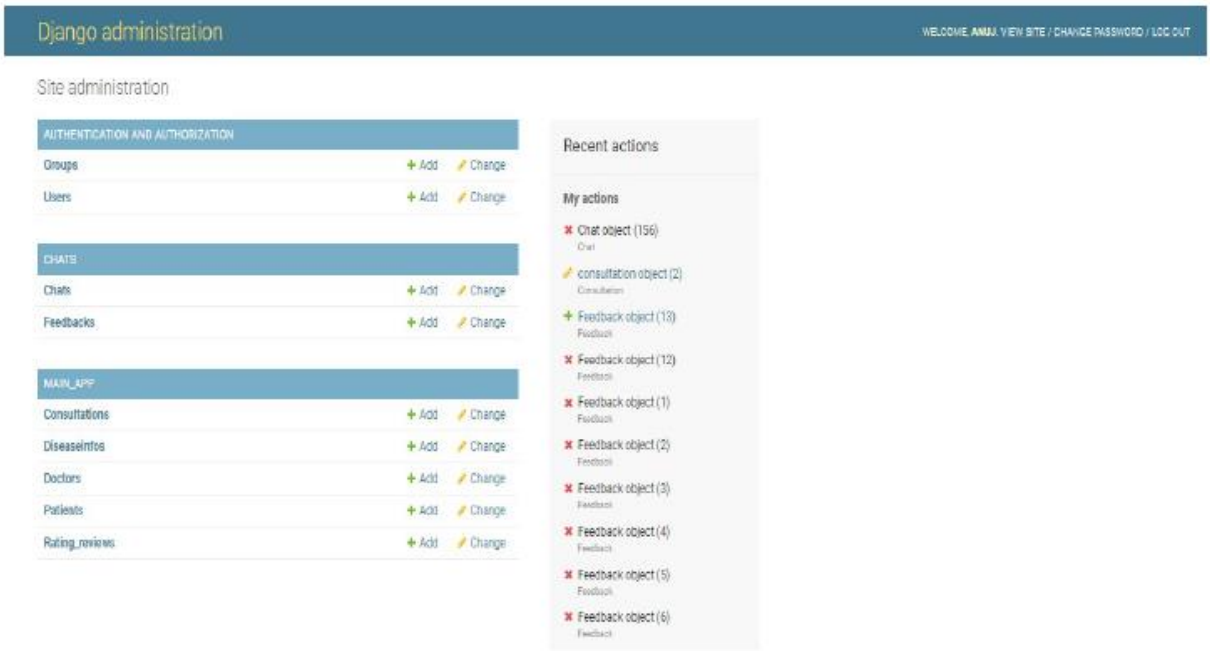


Figure B.12 ADMIN INTERFACE

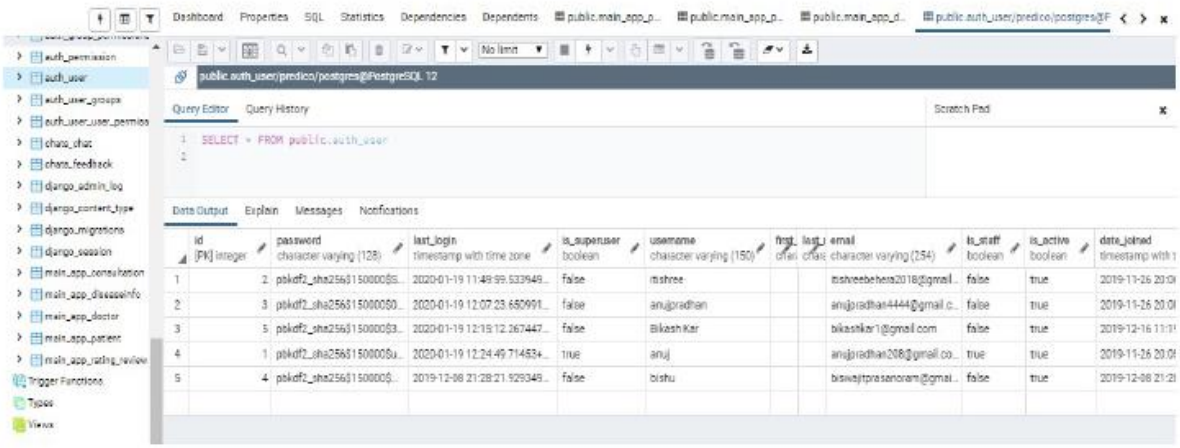


Figure B.13 DATABASE PREDICO USER TABLE

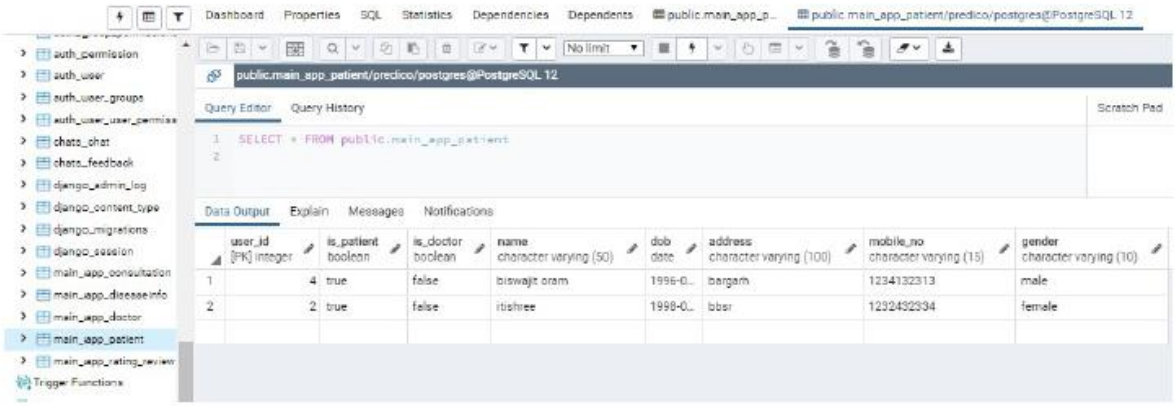
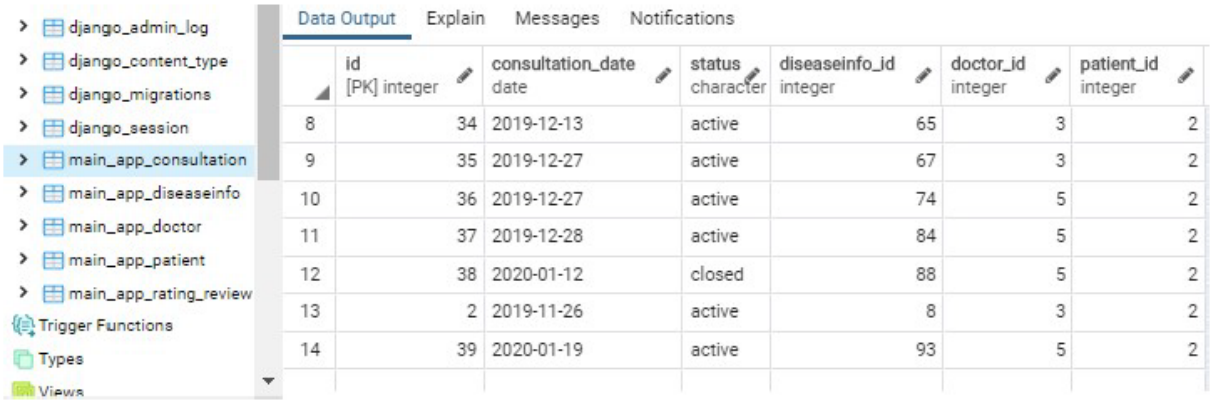


Figure B.14 PATIENT TABLE



	id [PK] integer	consultation_date date	status character	diseaseinfo_id integer	doctor_id integer	patient_id integer
8	34	2019-12-13	active	65	3	2
9	35	2019-12-27	active	67	3	2
10	36	2019-12-27	active	74	5	2
11	37	2019-12-28	active	84	5	2
12	38	2020-01-12	closed	88	5	2
13	2	2019-11-26	active	8	3	2
14	39	2020-01-19	active	93	5	2

Figure B.15 CONSULTATION TABLE