DIAGNOSIFY: AI - ASSISTED HUMAN DISEASE CLASSIFICATION AND PREDICTION CONSULTATION SYSTEM USING DEEP LEARNING TECHNIQUES

Submitted by

HARISH.B **(111421205016)**

PUNUGOTI KOWSHIK **(111421205041)**

PRAVEEN KUMAR B.S **(111421205043)**

In partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

PRATHYUSHA ENGINEERING COLLEGE

(An Autonomous Institution)

THIRUVALLUR – 602 025

ANNA UNIVERSITY :: CHENNAI 600 025

APRIL 2025

BONAFIDE CERTIFICATE

Certified that this project report **"DIAGNOSIFY: AI - ASSISTED HUMAN DISEASE CLASSIFICATION AND PREDICTION CONSULTATION SYSTEM USING DEEP LEARNING TECHNIQUES"** is the Bonafide work of the **" HARISH.B (111421205016), PUNUGOTI KOWSHIK (111421205041), PRAVEEN KUMAR B.S (111421205043) "** who carried out the project work under my supervision.

SIGNATURE
Dr.M.GOPIKRISHNAN , Ph.D,
PROFESSOR,
HEAD OF THE DEPARTMENT,
Department of IT,
Prathyusha Engineering college,
Thiruvallur- 602 025.

SIGNATURE
Mr.A.SUBBARAYUDU,M.Tech,
ASSISTANT PROFESSOR,
SUPERVISOR,
Department of IT,
Prathyusha Engineering college,
Thiruvallur- 602 025.

Place: Thiruvallur

Date:

Submitted for the Project Viva-Voce held on at
Prathyusha Engineering College, Thiruvallur- 602 025.

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGEMENT

Our sincere thanks to **Shri. P.RAJARAO**, Chairman, Prathyusha Engineering College for facilitating us to do this project.

We are grateful to our Vice Chairman, **Shri. P. CHARAN TEJA** & our CEO **Smt. P. PRATHYUSHA**, for being a source of Inspiration.

We are sincerely thanking **Dr.P.L.N.RAMESH, Ph.D.,** Dean, Prathyusha Engineering College for encouraging our endeavors for this project.

We are sincerely thanking **Dr.R.S.KUMAR**, **Ph.D.,** Principal, Prathyusha Engineering College for encouraging our endeavors for this project.

We are sincerely thanking **Dr.M.GOPIKRISHNAN**, **Ph.D.,** Professor, Head of Department,for supporting us in all stages of project conduction.

We are grateful to our project coordinator **Ms.K.SHILPA, M.E,** Assistant professor for her valuable suggestions and guidance.

We are grateful to our internal guide **MR.A.SUBBARAYUDU, M.Tech,** Assistant Professor for her valuable suggestions and guidance for the successful completion of this project.

We also wish to extend our sincere thanks to all the committee members for their constant support throughout the review.

We wish to express our sincere gratitude to **PARENTS AND FRIENDS** for valuable help, co-operation, and encouragement during this project work.

Last but not least, we wish to express our sincere thanks to the entire teaching faculty and non-teaching staff for their support.

ABSTRACT

Nowadays, humans face various diseases due to the current environmental condition and their living habits. The identification and prediction of such diseases at their earlier stages are much important, so as to prevent the extremity of it. It is difficult for doctors to manually identify the diseases accurately most of the time.

There are a lot of procedures for the treatment of multiple diseases across the world. Machine Learning is an emerging approach that helps in prediction, diagnosis of a disease. **Disease Detection** is done to detect potential health disorders or diseases in people who do not have any symptoms of disease. Early detection and lifestyle adjustments, as well as surveillance, are the goals in order to lower the risk of disease or to find it early enough to treat it effectively.

Our Project Named "***DISEASE DETECTION AND CONSULTATION USING DJANGO AND MACHINE LEARNING***" is designed for early detection and reduce the risk of disease, or to detect it early enough to treat it most effectively. This Project aims to present the functionality and accuracy of different machine learning algorithms to detect the disease a patient is suffering from based on the information or the symptoms he/she enter into the system and provides the accurate results based on that information, if the patient is not much serious and the user just wants to know the type of disease, he/she has been through. It is a system which provides the user the tips and tricks to maintain the health system of the user and it provides a way to find out the disease using this prediction. Now a day's health industry plays major role in curing the diseases of the patients so this is also some kind of help for the health industry to tell the user and also it is useful for the user in case he/she doesn't want to go to the hospital or any other clinics, so just by entering the symptoms and all other useful information the user can get to know the disease he/she is suffering from and the health industry can also get benefit from this system by just asking the symptoms from the user and entering in the system and in just few seconds they can tell the exact and up to some

extent the accurate diseases. Our proposed system mainly focusses on the development of a system where patients can feed symptoms into this tool as an input, which will be process and go through the machine learning module and then predict the possible disease to the patient. Finally, a doctor will be suggested by the system to cure the predicted disease of the patient. This dataset would then be analyzed using different machine learning algorithms to deliver results with maximum accuracy. This Disease Prediction Using Machine Learning is completely done with the help of Machine Learning and Python Programming language with Django framework for it and also using the dataset that is available previously by the hospitals using that we will predict the disease.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 GENERAL

"A disease is a condition that deteriorates the normal functioning of the cells, tissues, and organs". Diseases are often thought of as medical conditions that are characterized by their signs and symptoms. The disease can also be defined as:

"Any dangerous divergence from a functional or normal state of an entity". (Fig 1.1) When a person is inflicted with a disease, he exhibits a few symptoms and signs that range from normal to severe depending upon the medical condition. Hence, in order to identify different diseases, the normalcy of an entity needs to be studied and understood as a clear demarcation between disease and disease-free is not always apparent.

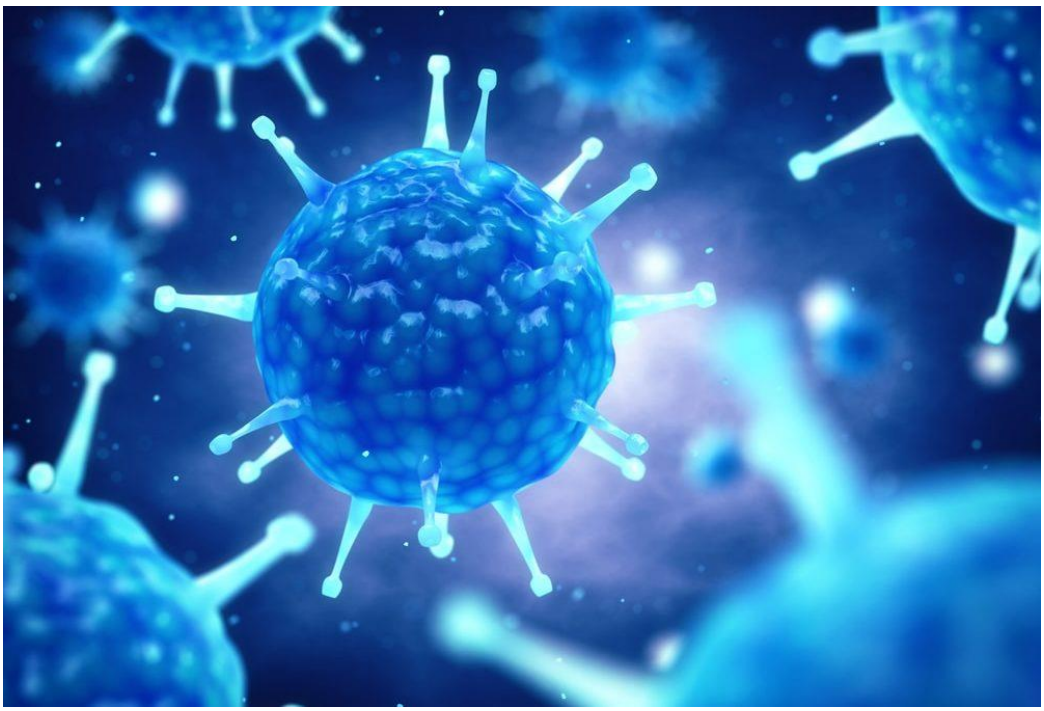

Fig 1.1 Disease : A disorder of Structure

The diseases are usually caused by many factors rather than a single cause. When we have a disease, we eventually show some signs, such as headaches, cough, cold, weakness. These

signs are referred to as “symptoms.” In almost all diseases, symptoms are shown immediately after having been struck by the disease. However, it varies depending upon the seriousness of the disease. Today, there are various ways to classify diseases.

Classification of Diseases:

Classification of Diseases		
Type	**Explanation**	**Example**
Anatomic Classification	This type refers to the affected organ or tissue	Heart disease
Topographical Classification	Further classified into types such as vascular disease, chest disease, gastrointestinal disease, and abdominal diseases. These are then handled by specializations in medicine that follow these topographical classifications	An ENT specialist (Ear-Nose-Throat) A Gastroenterology specialist etc.
Physiological Classification	This type includes diseases that affect a process or a function (such as metabolism, digestion or respiration)	Diabetes

Pathological Classification	This type considers the nature of the disease. For instance, cancer is associated with uncontrolled cell growth, and there are variations or types in the disease.	Neoplastic diseases (uncontrolled cell growth that are characteristic of cancer) Inflammatory diseases (autoimmunity)
Epidemiological Classification	This classification refers to the rate of occurrence, distribution and the control of the disease in a population.	Epidemic diseases such as plague and Influenza pandemic of 1918–1919

Types of Diseases:

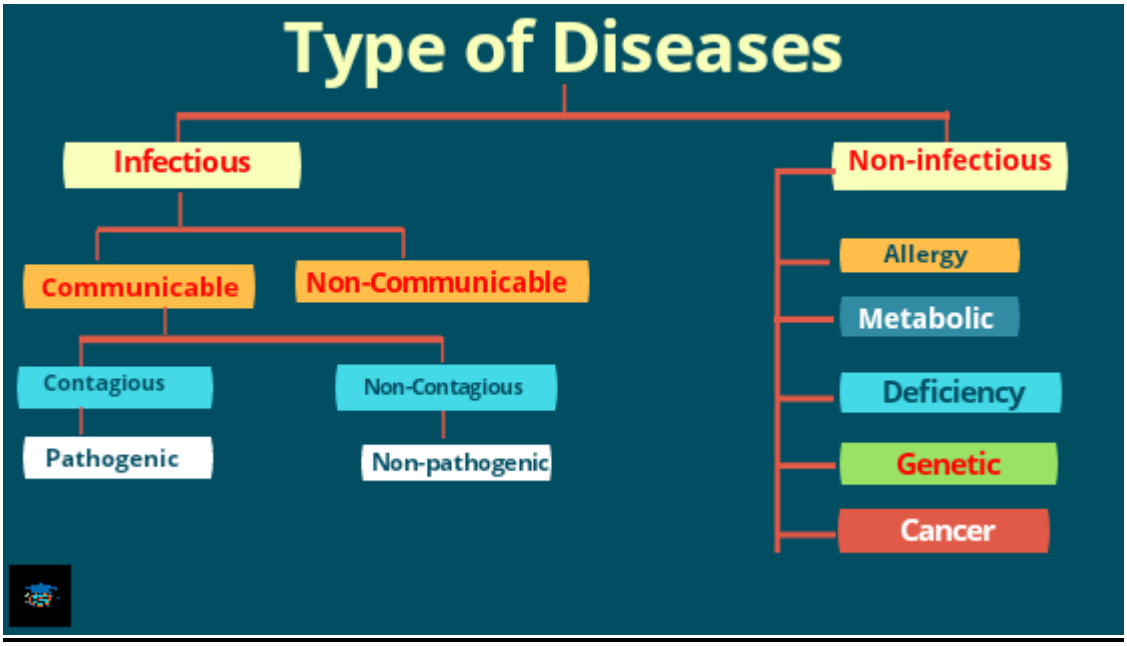

Fig 1.2 Types of Diseases

Diseases can be of two types

- Infectious diseases
- Non-infectious diseases

Infectious Diseases

Diseases that spread from one person to another are called communicable diseases. They are usually caused by microorganisms called pathogens (fungi, rickettsia, bacteria, viruses, protozoans, worms). When an infected person discharges bodily fluids, pathogens may exit the host and infected a new person (sneezing, coughing etc). Examples include Cholera, chickenpox, malaria etc.

Non-infectious Diseases

These diseases are caused by pathogens, but other factors such as age, nutritional deficiency, gender of an individual, and lifestyle also influence the disease. Examples include hypertension, diabetes, and cancer. They do not spread to others, and they restrain within a person who has contracted them. Alzheimer's, asthma, cataract and heart diseases are other non-infectious diseases.

Degenerative Diseases

They are mainly caused by the malfunctioning of vital organs in the body due to the deterioration of cells over time. Diseases such as osteoporosis show characteristics of degenerative diseases in the form of increased bone weakness. This increases the risk of bone fractures.

When degeneration happens to the cells of the central nervous system, such as neurons, the condition is termed as a neurodegenerative disorder. Alzheimer's is a prominent example of this disorder. Degenerative diseases are usually caused by ageing and bodywear. Others are caused by lifestyle choices, and some are hereditary.

Allergies

An allergic reaction arises when the body becomes hypersensitive to certain foreign substances called allergens. This usually happens when the immune system reacts abnormally to any seemingly harmless substances. Common allergens include dust, pollen, animal dander, mites, feathers, latex and also certain food products like nuts and gluten. Peanuts and other nuts have the capability to cause severe allergic reactions that may induce life-threatening conditions such as difficulty in breathing, tissues swelling up and blocking the airways and anaphylaxis shock.

Other common and less life-threatening symptoms include coughing, sneezing, running nose, itchy and red eyes, skin rashes. One of the best examples of this allergic reaction is asthma. Sometimes, bee stings and ant bites also trigger allergies. Consumption of shellfish and certain medication can induce allergic reactions.

Asthma is a chronic disease, mainly affects the bronchi and bronchioles of the lungs. One of the factors responsible for this is airborne allergens such as pollens or dust. Symptoms include difficulty in breathing, wheezing, and cough.

Deficiency Diseases

They occur due to the deficiencies of hormones, minerals, nutrients, and vitamins. For example, diabetes occurs due to an inability to produce or utilize insulin, goiter is mainly caused by iodine deficiency, kwashiorkor is caused by a lack of proteins in the diet. Vitamin B1 deficiency causes beriberi.

- **Goiters**

  It is an abnormal enlargement of the thyroid gland by blocking the oesophagus or other organs of the chest and neck. This causes difficulty in breathing and eating.

- **Blood Diseases**

  Blood contains the plasma, white blood cells, platelets, and red blood cells. When any of these components are affected, it can lead to blood disorders. For instance, the red blood cells are destroyed when a person contracts the sickle cell disease. The red blood cells are distorted into the shape of a sickle (hence, the name) and it loses its ability to carry oxygen. Consequently, this disease is characterized by symptoms similar to chronic anemia, such as shortness of breath and tiredness.

  Other diseases such as eosinophilic disorders, leukemia, myeloma (cancer of plasma cells in bone marrow), Sickle Cell Anemia, Aplastic Anemia, Hemochromatosis and Von Miller and Disease (blood-clotting disorder) fall under this classification.

  General Symptoms: Pale skin, swelling of lymph nodes, fever, bleeding, bruising, skin rashes, etc.

Disease-Causing Agents

We have seen the classification of different entities based on various characteristics. For simplification, we classify organisms to group them together and study about them as a class. Similarly, diseases are caused by different microorganisms and can be classified as diseases caused by bacteria, fungi, viruses etc. Some diseases are also caused by multicellular organisms such as worms.

Listed below are few diseases and the disease-causing agents

List of Diseases	
Disease	**Causative Agent**
Plague	Pasteurella pestis
Cholera	Vibrio comma (Vibrio cholera)

Tetanus	Clostridium tetani
Anthrax	Bacillus anthracis
Whooping cough	Bordetella pertussis
Human papillomavirus infection	Human papillomavirus
Acquired Immune Deficiency Syndrome (AIDS)	Human Immunodeficiency Virus (HIV)
Hepatitis	Hepatitis A, Hepatitis B, Hepatitis C, Hepatitis D, Hepatitis E viruses
Chickenpox	Varicella zoster virus (VZV)
Meningoencephalitis	Naegleria fowleri (amoeba)

1.2 OVERVIEW OF THE PROJECT

This Project aims to present the functionality and accuracy of different machine learning algorithms to detect the disease a patient is suffering from based on the information or the symptoms he/she enters into the system and provides the accurate results based on that information. The purpose of making this project is to predict the accurate disease of the patient using all their general information's and also the symptoms. Using this information, there we will compare with our previous datasets of the patients and predict the disease of the patient he/she has been through. If this Prediction is done at the early stages of the disease with the help of this project and all other necessary measures the disease can be cured and in general this prediction system can also be very useful in the health industry. If the health industry adopts this project then the work of the doctors can be reduced, and they can easily predict the disease of the patient. The general purpose of this Disease prediction is to provide predictions for the various and generally occurring diseases that when unchecked and sometimes ignored can turn into fatal disease and cause a lot of problems to the patient and as well as their family members. This system will predict the most possible disease based on the symptoms. The

health industry is information yet knowledge poor and this industry is a very vast industry which has a lot of work to be done. So, with the help of all those algorithms, techniques, and methodologies we have done this project which will help the people who are in need.

1.3 LITERATURE SURVEY

Numerical Research Work has been carried out for the prediction of diseases based on the symptoms shown by an individual using machine learning algorithms. Monto et al. designed a statistical model to predict whether a patient had influenza or not. They included 3744 unvaccinated adults and adolescent patients of influenza who had fever and at least 2 other symptoms of influenza. Out of 3744, 2470 were confirmed to have influenza by the laboratory.

Based on this data, their model gave an accuracy of 79 %. Sreevalli et al. used the random forest machine-learning algorithm to predict the disease based on the symptoms. The system resulted in low time consumption and minimal cost for the prediction of diseases. The algorithm resulted in an accuracy of 84.2 %.

Various tools were developed by Langbehn et al. to detect Alzheimer's disease. Data for 29 adults were used for the training purpose of the ML algorithm. They had developed classification models to detect reliable absolute changes in the scores with the help of SmoteBOOST and wRACOG algorithms. A variety of ML techniques such as artificial neural networks (ANNs), bayesian networks (BNs), support vector machines (SVMs) and decision trees (DTs) have been widely applied in cancer research for the development of predictive models, resulting in effective and accurate decision making. Karayilan et al. proposed a heart disease prediction system that uses the artificial neural network backpropagation algorithm. 13 clinical features were used as input for the neural network and then the neural network was

trained with the backpropagation algorithm to predict absence or presence of heart disease with an accuracy of 95 %.

Various machine learning algorithms were streamlined for the effective prediction of a chronic disease outbreak by Chen et al. The data collected for the training purpose was incomplete. To overcome this, a latent factor model was used. A new convolutional neural network-based multimodal disease risk prediction (CNN-MDRP) was structured. The algorithm reached an accuracy of around 94.8 %. Chae et al. used 4 different deep learning models namely deep neural networks (DNN), long short-term memory (LSTM), ordinary least squares (OLS), an autoregressive integrated moving average (ARIMA) for monitoring 80 infectious diseases in 6 groups. Of all the models used, DNN and LSTM models had better performance. The DNN model performed better in terms of average performance and the LSTM model gave close predictions when occurrences were large.

Haq et al. used a database that contained information about patients having any heart disease. They extracted features using three selection algorithms which are relief, minimum redundancy, and maximum relevance (mRMR), and least absolute shrinkage and selection operator which was cross-verified by the K-fold method. The extracted features were sent to 6 different machine learning algorithms and then it was classified based on the presence or absence of heart disease. An effective heart disease prediction system was developed by Mohan et al.

They achieved an accuracy level of 88.4 % through the prediction model for heart disease with the hybrid random forest with a linear model (HRFLM). Maniruzzaman et al. classified the diabetes disease using ML algorithms. Logistic regression (LR) was used to identify the risk factors for diabetes disease. The overall accuracy of the ML-based system was 90.62 %.

Method	*Model Used*	*Maximum Accuracy*
Mir et al.	Naive Bayes, SVM Random Forest and Simple CART	79.13
Vijayarani et al.	SVM	79.66
Mohan et al.	HRFLM	88.4
Sriram et al.	Random forest	90.26
Our proposed method	Decision Tree, Random Forest, Gradient Boosting, K-nearest Neighbors, GNB	94.78

1.4 PROBLEM STATEMENT

Now a day's in Health Industry there are various problems related to machines or devices which will give wrong or unaccepted results, so to avoid those results and get the correct and desired results we are building a program or project which will give the accurate predictions based on information provided by the user and also based on the datasets that are available in that machine. The health industry is information, yet knowledge poor and this industry is a very vast industry which has a lot of work to be done. So, with the help of all those algorithms, techniques, and methodologies we have done this project which will help the people who are in need.

So, the problem here is that many people goes to hospitals or clinic to know how their health is and how much they are improving in the given days, but they have to travel to get to know there answers and sometimes the patients may or may not get the results based on various factors such as doctor might be on leave or some whether problem so he might not have come to the hospital and many more reasons will be there so to avoid all those reasons and confusion we are making a project which will help all those person's and all the patients who are in need to know the condition of their health, and at sometimes if the person has been observing few symptoms and he/she is not sure about the disease he/she is encountered with so this will lead to various diseases in future. So, to avoid that and get to know the disease in early stages of the symptoms, this disease prediction will help a lot to the various people ranging from children to teenagers to adults and also the senior citizens.

1.5 SCOPE OF STUDY

The idea behind this project was to come up with an automated system that can discover and extract hidden knowledge associated with the diseases from a historical(diseases-symptoms) database according to the rule set of the respective algorithms. The healthcare and medical sector are more in need of data mining today. When certain data mining methods are used in the right way, valuable information can be extracted from large databases and that can help the medical practitioner to make early decisions and improve health services. Diseases and health related problems like malaria, dengue, Impetigo, Diabetes, Migraine, Jaundice, Chickenpox etc., cause significant effect on one's health and sometimes might also lead to death if ignored. The healthcare industry can make an effective decision making by "mining" the huge database they possess i.e., by extracting the hidden patterns and relationships in the database. Data mining algorithms like Decision Tree, Random Forest and Naïve Bayes algorithms can give a remedy to this situation. The techniques of machine learning have been successfully employed

in assorted applications including Disease prediction. The aim of developing a classifier system using machine learning algorithms is to immensely help to solve the health-related issues by assisting the physicians to predict and diagnose diseases at an early stage.

The spirit is to use the classification in order to assist the physician. Smart data processing is emerging as a requirement for effective and robust diseases to be found by society. Detection of patients providing the necessary treatment as soon as possible within the shortest possible period. This identification has been achieved in recent decades through the method of identifying exciting patterns in databases. A comprehensive overview of intelligent data analysis tools in the medical sector is illustrated in this project.

CHAPTER 2

SYSTEM ANALYSIS

2.1 GENERAL

Prediction using traditional methods and models involves various risk factors and it consists of various measures of algorithms such as datasets, programs and much more to add on. High-risk and Low-risk patient classification is done on the basis of the tests that are done in groups. But these models are only valuable in clinical situations and not in big industry sectors. So, to include the disease predictions in various health related industries, we have used the concepts of machine learning and supervised learning methods to build the predictions system.

2.2 PRELIMINARY INVESTIGATION

After doing the research and comparison of all the algorithms and theorems of machine learning we have come to conclusion that all those algorithms such as Decision Tree, KNN, Naïve Bayes, Regression and Random Forest Algorithm all are important in building a disease prediction system which predicts the disease of the patients from which he/she is suffering from and to do this we have used various machine learning techniques to make predictions of the diseases and after doing that we come to the conclusion that it can predict up to 93% accuracy rate after doing the experimentation and verifying the results. The information of patient statistics, results, and disease history is recorded in EHR, which enables us to identify the potential data centric solution, which reduces the cost of medical case studies. Existing system can predict the disease but not the subtype of the disease and it fails to predict the condition of the people, the predictions of disease have been indefinite and non-specific.

2.3 FEASIBILITY STUDY

2.3.1 TECHNICAL FEASIBILITY

The feasibility of creating a disease detector in hardware utilizing various Machine learning algorithms is investigated in this section. We discovered that Kaggle's dataset may be used to detect disease from a set of given symptoms and disease test data, and that our detection methods are resistant to slight modifications in input symptoms. As a result, we can detect many changes within a disease family after evaluating a limited collection of variations on Windows Platform.

We propose methods and techniques for

(1) collection of fine-grained runtime data without slowing down applications,

(2) secure execution of Machine Learning algorithms to detect at runtime the type of disease and

(3) Providing Consultation to the Patient via chat application.

Furthermore, our suggested hardware changes allow the disease detector to run securely behind the system software, paving the way for providing consultation to the patient that is more fast, reliable, and accessible than other typical offline consultation.

2.3.2 ECONOMICAL FEASIBILITY

This study is being carried out to determine the system's economic impact on the organization. With the emergence of various online consultation facilities providing cheap access to sign-in and chat-consultation as a service, there is a new platform in which patients can directly consult the related doctor about their medical condition after logging-in to their account.

Thus, Machine Learning, when integrated with the secure and maintainable Django Framework, has the potential to advance state-of-the-art online Disease detection and Consultation.

2.3.3 OPERATIONAL FEASIBILITY

Operational feasibility is a measure of how successfully a proposed system solves issues and capitalizes on possibilities discovered during scope definition, as well as how well it meets the criteria determined during the requirements analysis phase of system development. The project "Disease Detection Using Django and Machine Learning " solves the problem of identifying the type of disease a person is suffering from based upon the symptoms that he/she gives as input to the system and furthermore providing online consultation facility to the patient once the diagnosis is done.

2.4 SOFTWARE AND HARDWARE SPECIFICATION

The project involved analyzing the design of a few applications so as to make the application more users friendly. To do so, it was really important to keep the navigations from one screen to the other well-ordered and at the same time reducing the amount of typing the user needs to do. In order to make the application more accessible, the browser version had to be chosen so that it is compatible with most of the Browsers.

Software Requirements:

For developing the application, following are the Software Requirements:

Machine Learning:

- Python 3.10
- VS Code (version 1.62.3) as IDE

- Anaconda Environment (Jupyter Package)

Project Implementation:

Operating System Supported:

- Windows 8
- Windows 10
- Windows 11

Web Browser:

- Any Browser, especially Google Chrome (Version 96.0.4664.45)

Hardware Requirements:

- Processor : Intel i3 2120 2nd Generation 3.3 Ghz
- RAM: 8 GB
- System Type : 64-bit/32-bit Operating System, x64/x86 based Processor
- Minimum Space on Hard Disk: 1 GB

2.5 DATA FLOW DIAGRAM

In the below figure (Fig 2.1), the Data Flow Diagram for Disease Detection and consultation has been shown.

The data flow diagram of the project disease prediction using machine learning consists of all the various aspects a normal flow diagram requires.

This data flow diagram shows how from starting the model flows from one step to another, like user enters into the system then enters all the information's and all other general information along with the symptoms that goes into the system, compares with the prediction

model and if true is predicts the appropriate results otherwise it shows the details where the user if gone wrong while entering the information.

Disease prediction using machine learning predicts the presence of the disease for the user based on various symptoms and the information the user gives such as sugar level, hemoglobin level and many more such general information through the symptoms. The flow of the system disease prediction using machine learning consist of various datasets through which we will compare the symptoms of the user and predicts it, then the datasets are transformed into the smaller sets and from there it gets classified based on the classification algorithms later on the classified data is then processed into the machine learning technologies through which the data gets processed and goes in to the disease prediction model using all the inputs from the user that is mentioned above.

Then after the user enters the above information and overall processed data combines and compares in the prediction model of the system and finally predicts the disease. An architecture diagram is a graphical representation of a set of concepts that are part of an architecture, including their principles, elements and components. The diagram explains about the system software in perception of overview of the system.

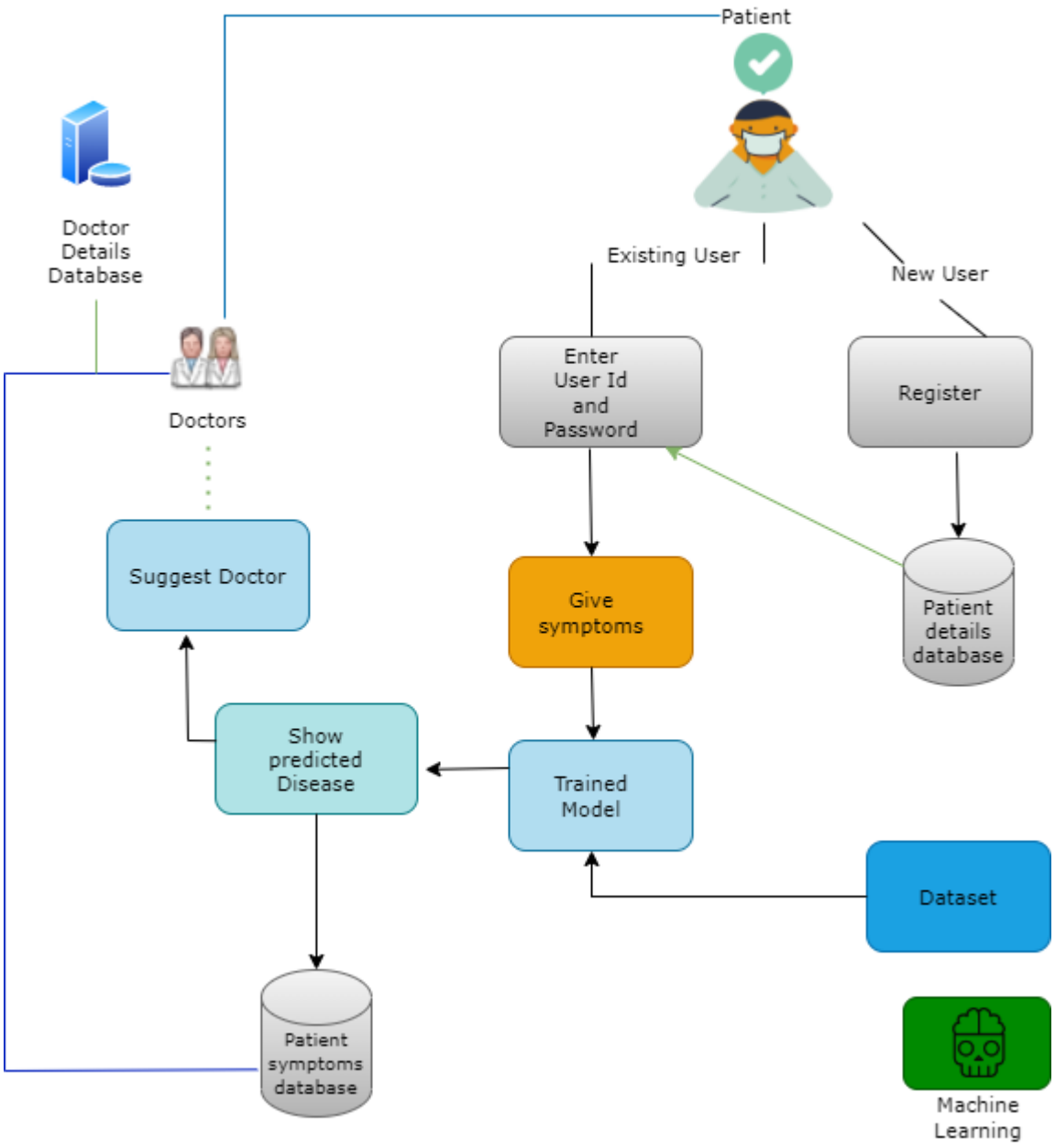

Fig 2.1 Data Flow Diagram : Disease Detection and Consultation

CHAPTER 3

SYSTEM DESIGN

3.1 DESIGN METHODOLOGY

We propose a real-time role-based access web-Interface that would first enable the user to Register/login as a doctor or a patient, then further on incase of Patient would provide with the choice to input his/her symptoms and would predict what disease the patient is suffering from and give a facility for online consultation to the suggested doctor for that disease.

In case of a Doctor, the user would login and get access to the consultation History and can view the patient's profile and provide real time consultation.

3.1.1 MACHINE LEARNING :

Detection of disease from a set of given input symptoms is done through Machine Learning. 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 contains 4920 rows of records of the patients with their symptoms (132 types of different symptoms) and their corresponding disease (41 classes of general disease). The system is trained to predict the diseases using five algorithms

- Decision Trees Classifier
- Random Forest Classifier
- Gradient Boosting Classifier
- K-nearest Classifier
- Gaussian Naive Bayes Classifier

Once the system is trained with the training set using the mentioned algorithms a rule set is formed and when the user the symptoms are given as an input to the model, those symptoms are processed according to the rule set developed, thus making classifications, and predicting the most likely disease.

3.1.2 UI DESIGN:

Starting with the Home page, there would be five template HTML files to be used for the website – admin, consultation, Doctor, patient, and sign-in page. After the successful login of the user, sign-in is loaded where the user gets to sign-in as either Doctor, Patient or Admin. After successfully logging in into the concerned role, the user gets more options to explore, in case of patient - getting consultation facility and incase of Doctor - viewing profile of patient and consultation History menu. It is an interactive web page with responsive buttons and links and various resources such as Google font, Bootstrap, Owl Carousel, Magnific popup, CSS and Awesome Icon. The web application also takes into use static files such as javascript and CSS, supporting the display of a web page. Usually, the web server is configured to serve them for us, but during the development, these files are served from the static folder in your package or next to your module and it will be available at /static on the application. A special endpoint 'static' is used to generate URL for static files.

3.1.3 DJANGO AS FRAMEWORK:

The Interface is built upon Django which is a high-level Python web framework that encourages rapid development and clean, pragmatic design. In Django, every web app you want to create is called a project; and a project is a sum of applications. An application is a set of code files relying on the MVT pattern. As example , let's say we want to build a website, the website is our project and the forum, news, contact engine are applications. This structure makes it easier to move an application between projects since every application is independent. Every web-app in Django has the following structure -

```
myproject/
   manage.py
   myproject/
      __init__.py
      settings.py
      urls.py
      wsgi.py
```

Fig 3.1 Django Project Structure

We have the following web apps in the system besides the main app:

- accounts : This manages the register/login of the user.
- chats: This web app handles the chat application used during the consultation between patient and the Doctor.
- disease prediction: This web app handles the password validation and authentication.

After having the patient input the symptoms, we ran our machine learning model to predict what kind of disease the patient is suffering from and from thereon provide consultation facility to the patient.

3.1.4 PostgreSQL as Database Storage:

We save the login details of patients and Doctors using the PostgreSQL, which is an advanced, enterprise class open-source relational database that supports both SQL (relational) and JSON (non-relational) querying. We first installed pgAdmin4 then clicking onto the server, we created our database Predico. Then. going on to settings.py we get the database config part as,

```
DATABASES = {
   'default': {
      'ENGINE': 'django.db.backends.postgresql',
      'NAME': 'predico',
      'USER': 'postgres',
```

```
        'PASSWORD': 'tiger',
        'HOST': 'localhost'
    }
}
```

3.2 USER INTERFACE DESIGN

The corresponding figure (Fig 3.1) is the snapshot Home page where users will get the option to sign-in/sign-up and get information related to Doctors sign up and proceed further with the detection process and then to the Consultation.

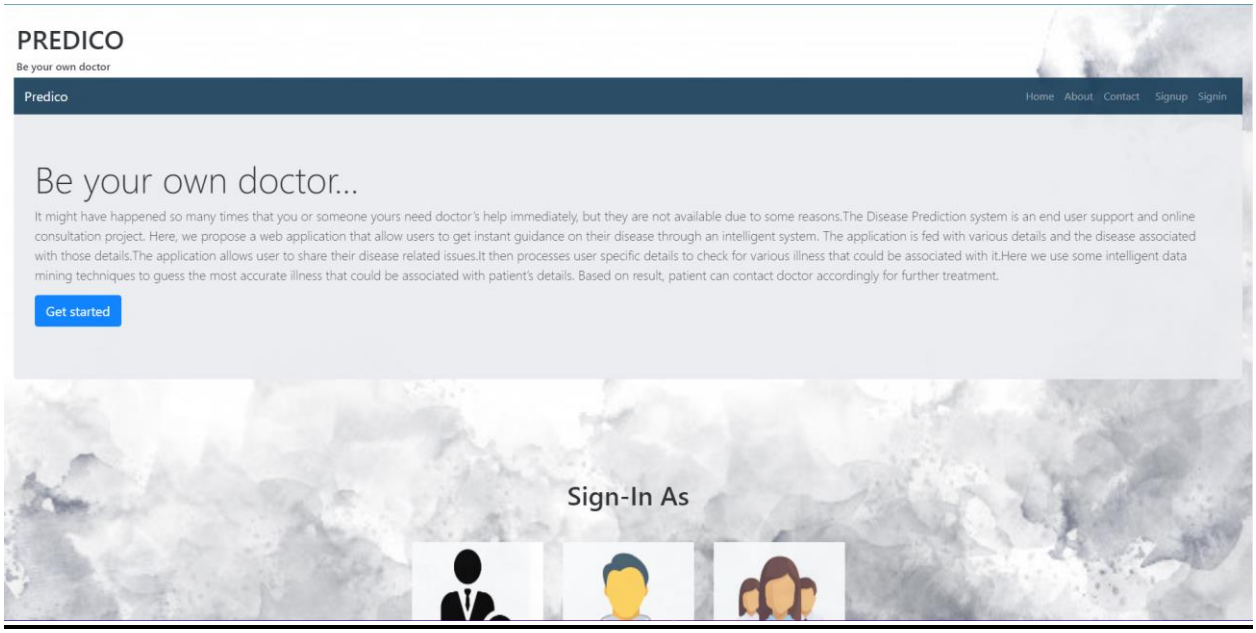

Fig 3.2 Home Page Displayed to User

The below displayed figure (Fig 3.3) is of the Sign-Up page. This will allow the Doctor/Patient to sign-up using his/her details.

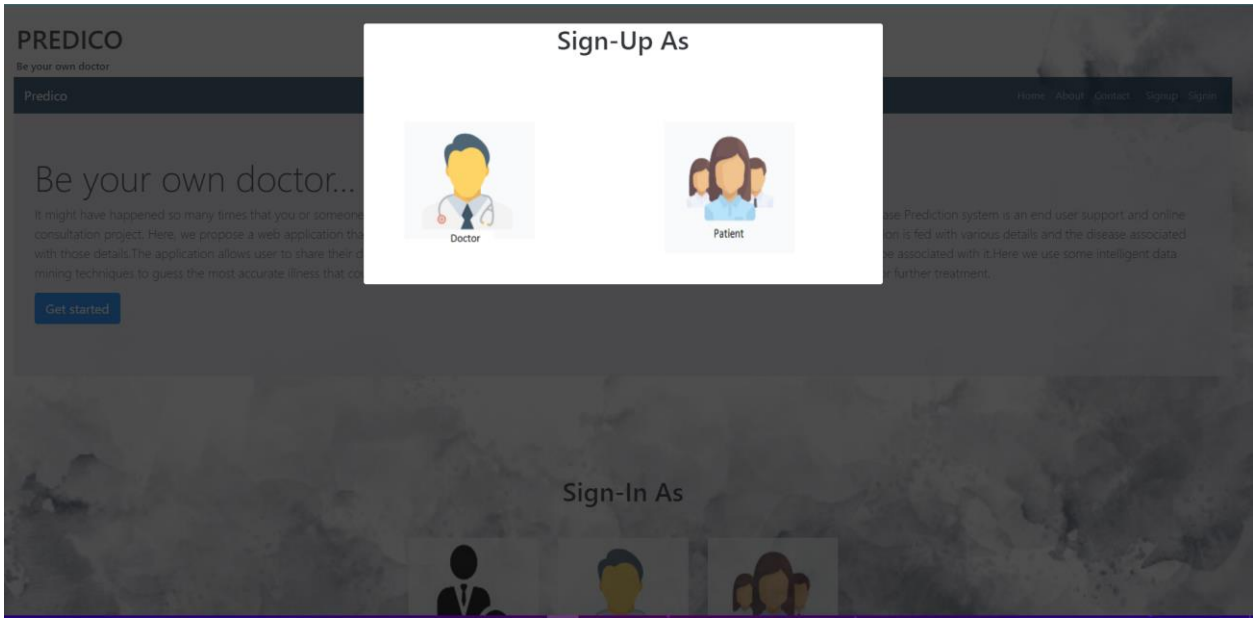

Fig 3.3 Sign Up Page Displayed to User

The below displayed figure (Fig 3.4) is of the Sign-in page. This will allow the Doctor/Patient to sign-in using his/her details.

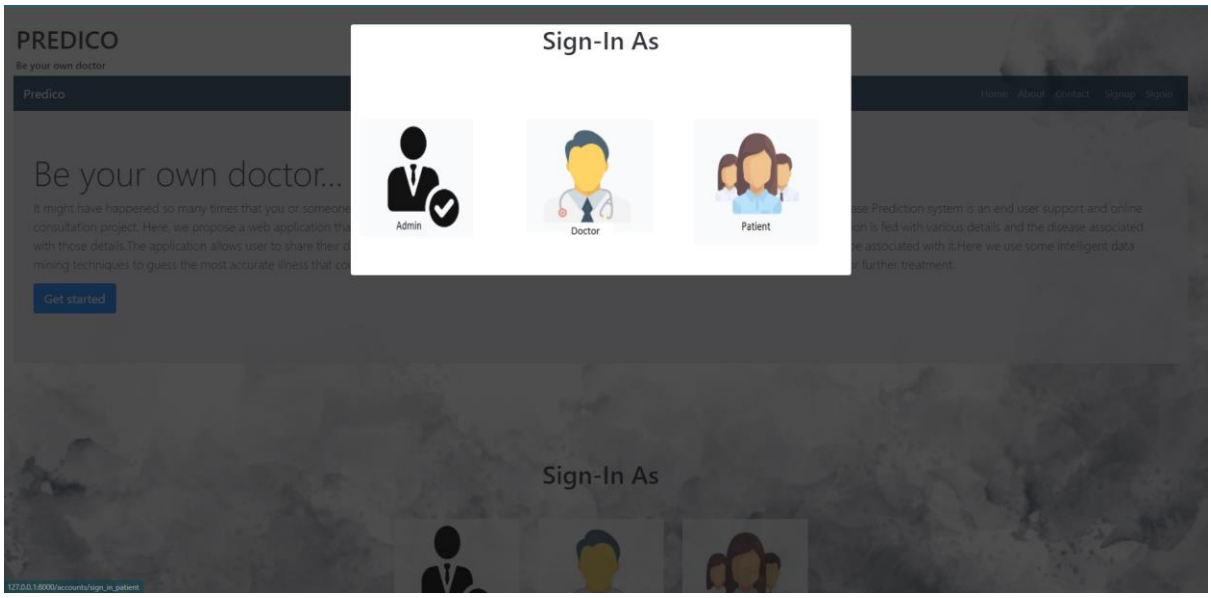

Fig 3.4 Sign-in Page Displayed to User

The below figure (Fig 3.5) is the interface shown to the user to login as Doctor/ Patient in order to get access to the account.

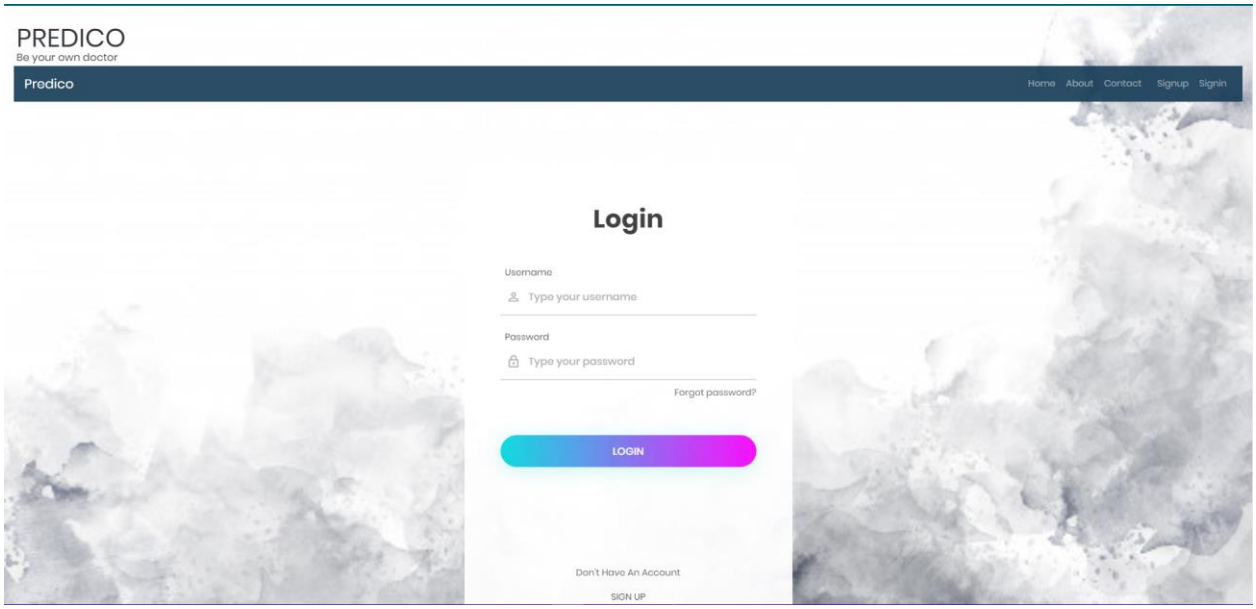

Fig 3.5 Login Page Displayed to User

The below figure (Fig 3.6) is the Patient's profile page wherein the patient could check disease, view his/her consultation history or give any feedback related to the system.

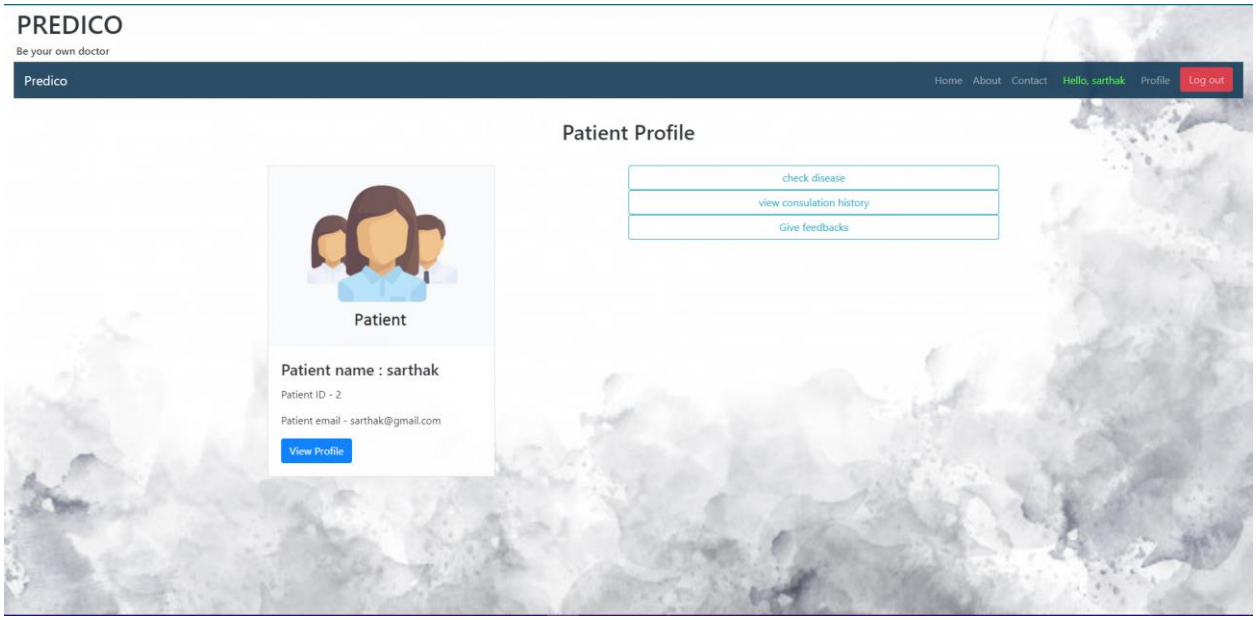

Fig 3.6 Patient Profile Page

The corresponding figure (Fig 3.7) is designed for adding symptoms as input to the system so that the Disease Predictor could identify the type of disease a patient is suffering from.

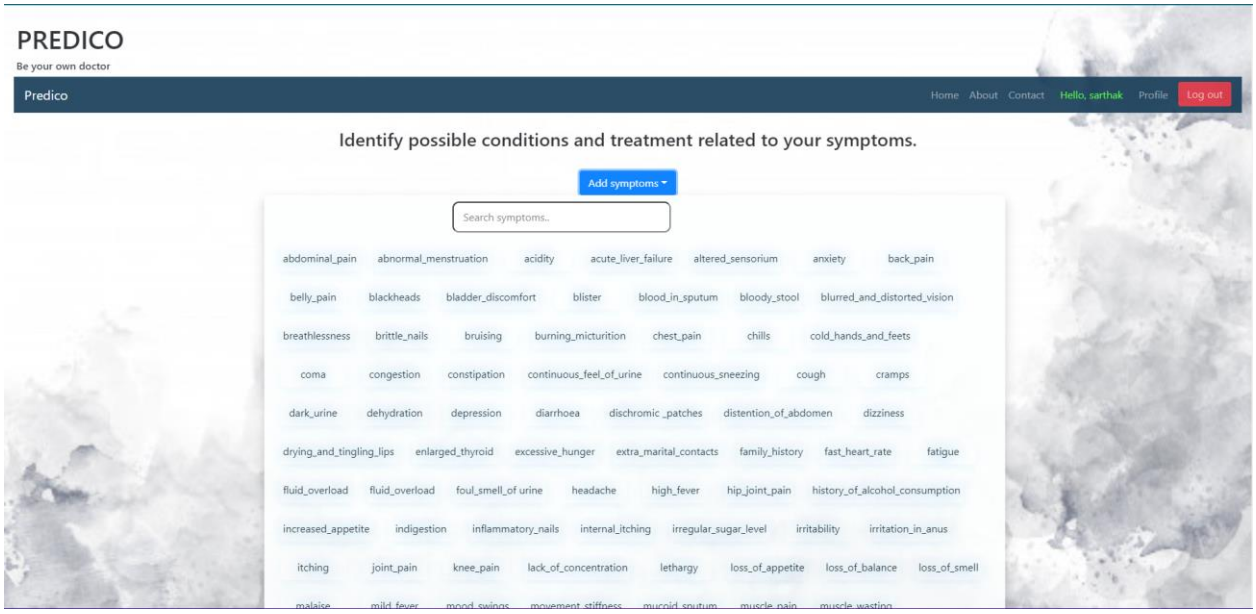

Fig 3.7 Symptoms Input page

Below figure (Fig 3.8) is a snapshot of the prediction results page that shows what disease the patient is suffering from and furthermore recommends a suitable doctor for that.

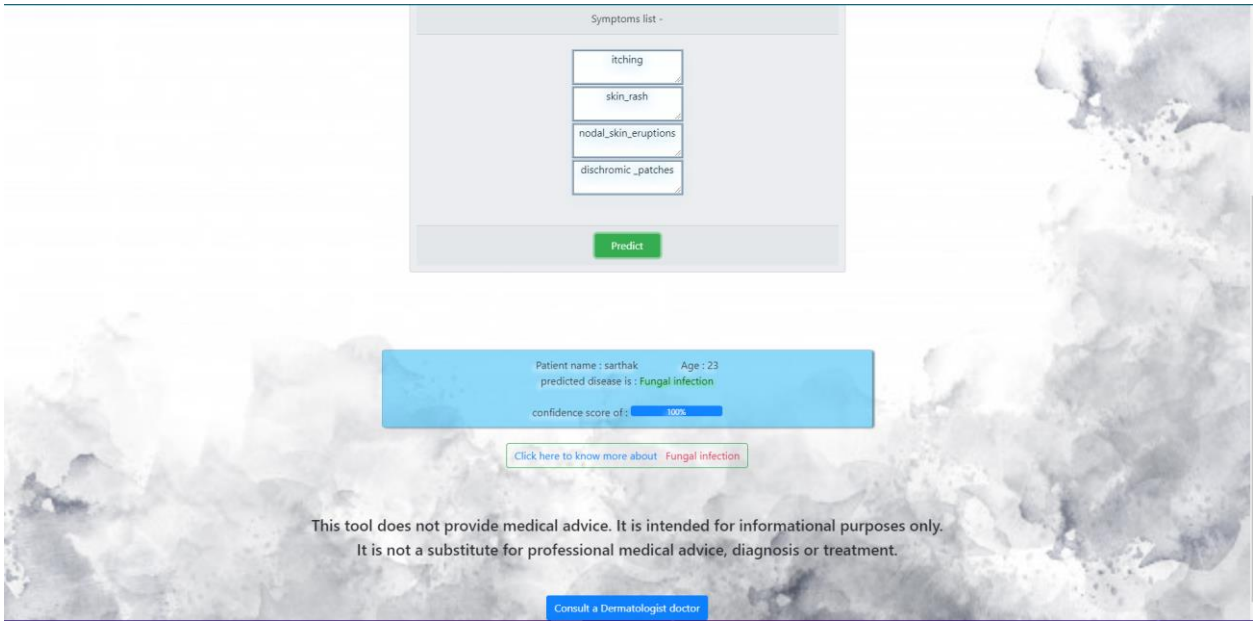

Fig 3.8 Prediction result page

Below figure (Fig 3.9) is a snapshot of the Doctor consultation page that shows all the available doctors and provides options to the patient to choose from.

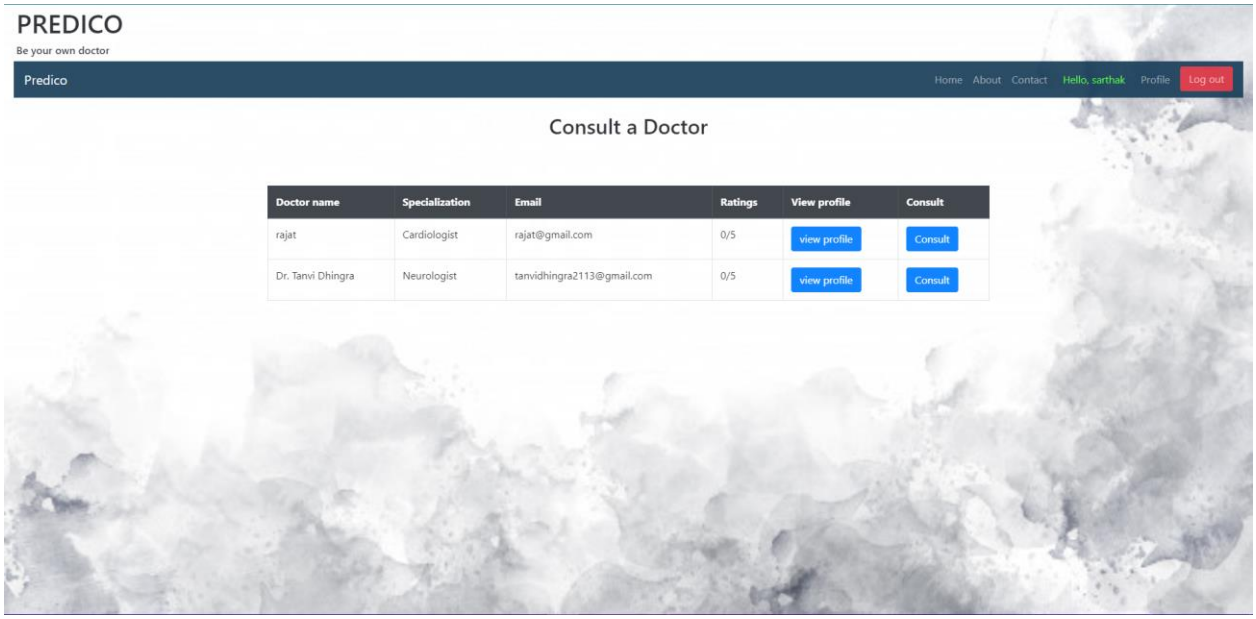

Fig 3.9 Doctor Consultation Page

The below figure (Fig 3.10) is the interface shown to the user to login as Doctor/ Patient in order to get access to the account.

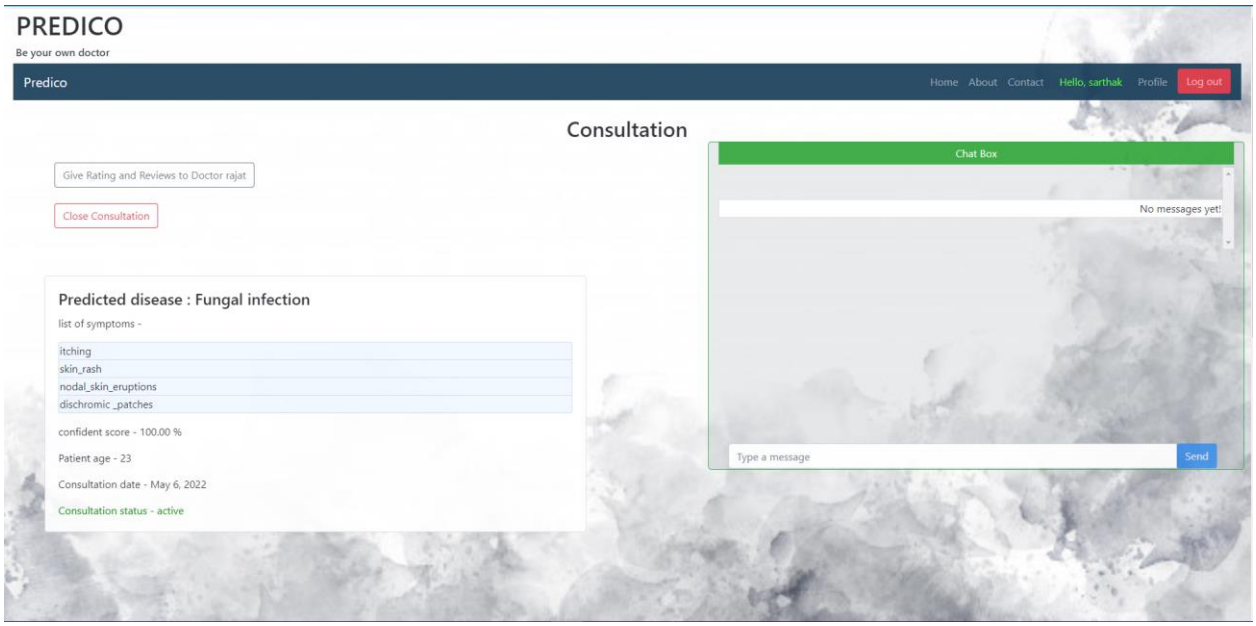

Fig 3.10 Consultation chat Page

CHAPTER 4

TESTING

4.1 TESTING TECHNIQUES AND TESTING STRATEGIES

As a part of testing the software, it is important to train the Machine Learning Model with the most important features of the dataset. For this, the most important 95 features are observed from the Disease Dataset.csv file from a total of 132 features using Feature Selection Techniques. The same is shown in the below figure (4.1)

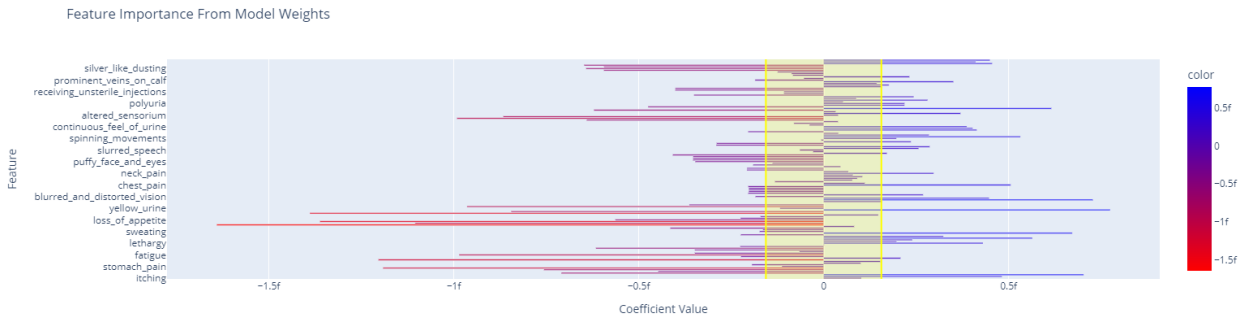

Fig 4.1 Feature importance from Model Weights

```
Index(['itching', 'stomach_pain', 'ulcers_on_tongue', 'muscle_wasting',
       'anxiety', 'restlessness', 'headache', 'constipation', 'yellow_urine',
       'fluid_overload', 'weakness_in_limbs', 'fast_heart_rate',
       'pain_during_bowel_movements', 'pain_in_anal_region', 'bloody_stool',
       'irritation_in_anus', 'dizziness', 'obesity', 'swollen_blood_vessels',
       'drying_and_tingling_lips', 'slurred_speech', 'movement_stiffness',
       'weakness_of_one_body_side', 'passage_of_gases', 'internal_itching',
       'toxic_look_(typhos)', 'altered_sensorium', 'belly_pain',
       'family_history', 'rusty_sputum', 'receiving_blood_transfusion',
       'receiving_unsterile_injections', 'fluid_overload.1', 'palpitations',
       'pus_filled_pimples', 'blackheads', 'scurring'],
```

Fig 4.2 Feature Selection performed on Disease Data.csv

4.2 DEBUGGING AND CODE IMPROVEMENT

After successful training the model with important features, so the most accurate results are obtained and displayed to the user, we test our model for the accuracies of 5 different machine learning algorithms namely :

- Decision Trees
- Random Forest
- Gradient Boosting
- K-nearest
- Gaussian Naive Bayes

The same is shown in the below figure (Fig 4.2)

```
#Algorithm comparison
algorithms = {
        "DecisionTree": tree.DecisionTreeClassifier(max_depth=100),
        "RandomForest": ske.RandomForestClassifier(n_estimators=10),
        "GradientBoosting": ske.GradientBoostingClassifier(n_estimators=50),
        "K-Nearest": KNeighborsClassifier(n_neighbors=5),
        "GNB": GaussianNB()
    }
```

Fig 4.3 Algorithms Comparison for Accurate Results

Looping over the dictionary iterable of algorithms for testing out each algorithm. It is shown in (Fig 4.3)

```
results = {}
print("\nNow testing algorithms")
for algo in algorithms:
    clf = algorithms[algo]
    clf.fit(X_train, y_train)
    score = clf.score(X_test, y_test)
    print("%s : %f %%" % (algo, score*100))
    results[algo] = score
```

Fig 4.4 Testing out each Machine Learning Algorithm

The accuracy results obtained by various algorithms is shown in the below output snippet in Fig 4.5

```
Now testing algorithms
DecisionTree : 97.112156 %
RandomForest : 97.112156 %
GradientBoosting : 97.112156 %
K-Nearest : 97.112156 %
GNB : 97.112156 %
```

Fig 4.5 Accuracy Results obtained with each Machine Learning Algorithm

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set X_test, y_test. Note that the word “experiment” is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally. Here is a flowchart of typical cross validation workflow in model training. The best parameters can be determined by grid search techniques. The same is shown below (Fig 4.5)

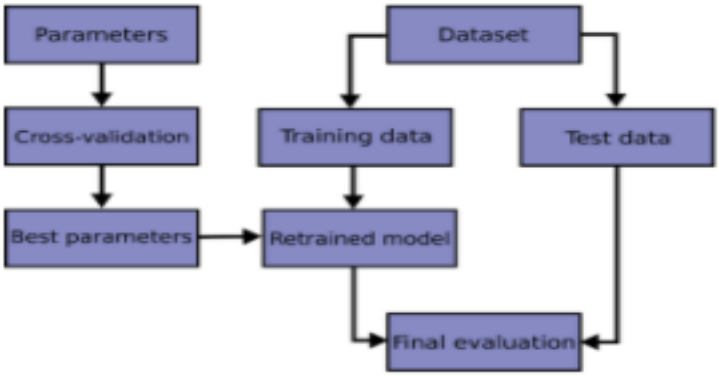

Fig 4.6 Cross Validation

Thus, it is clear that after successful debugging and code improvement the following results are obtained by the code (Fig 4.6)

```
Algorithm with highest accuracy on train/test is DecisionTree with a 97.112156 % success
```

Fig 4.7 Final Results obtained after Successful Code Run

A confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning. Below obtained is the same for our model.

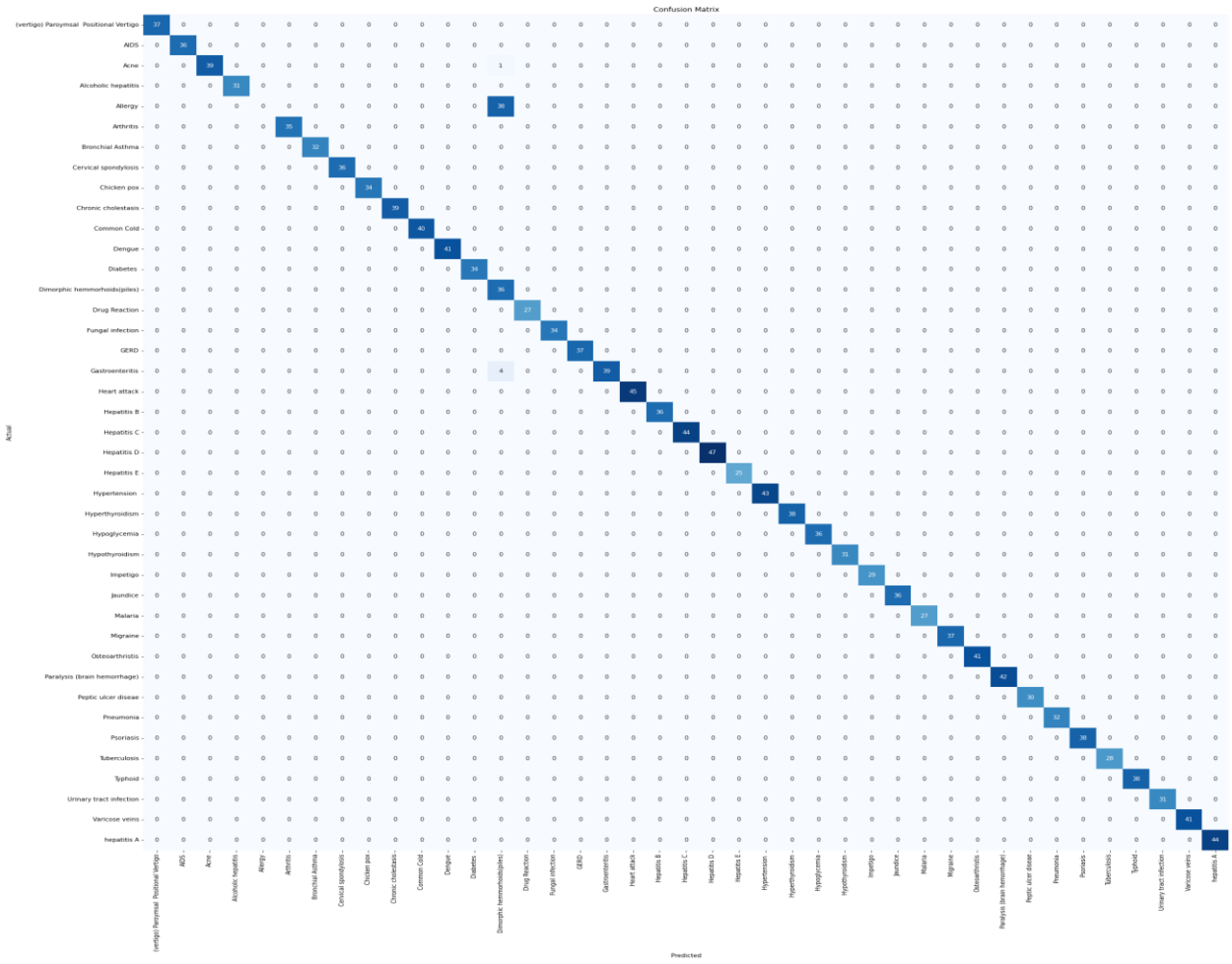

Fig 4.8 Plotting Confusion Matrix

Hence, the final chosen algorithm named Decision Tree's results would be displayed to the user and would help him get the kind of disease the patient is suffering from..

CHAPTER 5

IMPLEMENTATION

5.1 SYSTEM IMPLEMENTATION

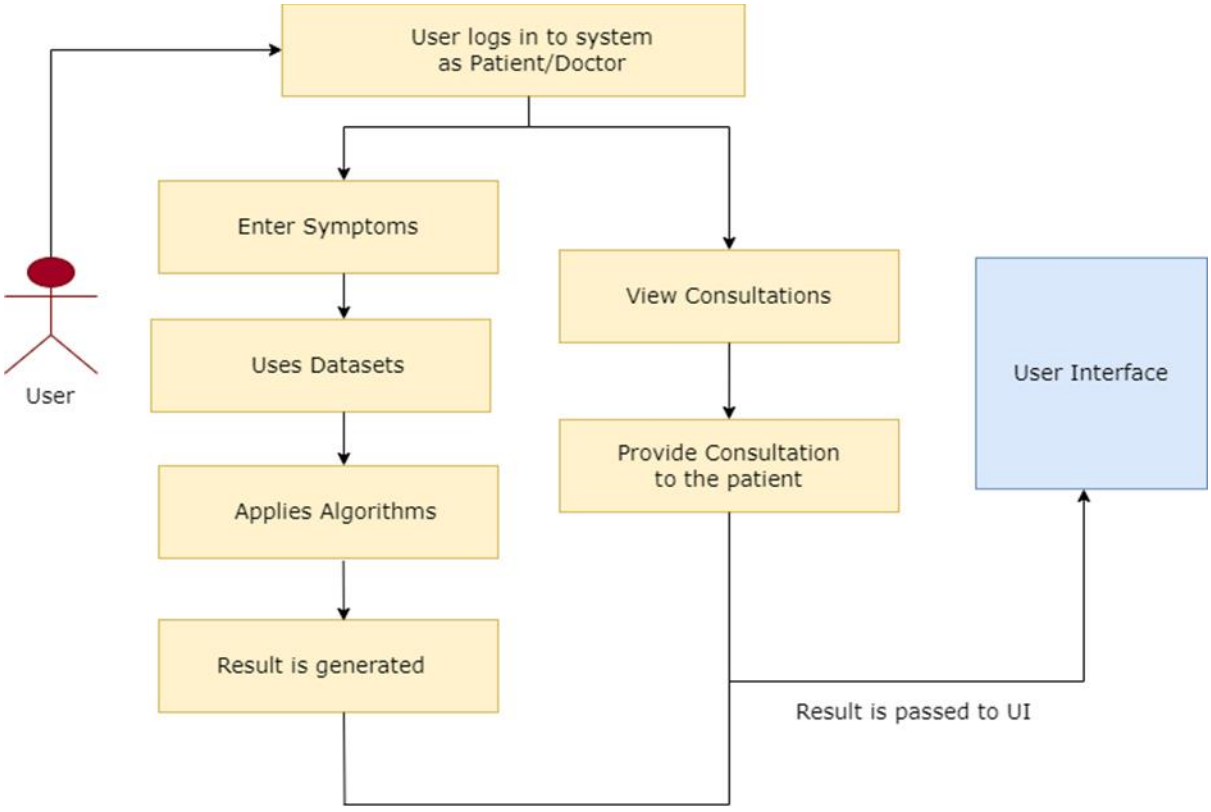

Fig 5.1 System Implementation Flow Chart

5.1.1 TECHNOLOGIES TO BE INCORPORATED AND WORKED UPON:

5.1.1.1MACHINE LEARNING (DETECTING SOFTWARE TO BE MALICIOUS OR LEGITIMATE):

Machine Learning is a category of algorithms that allow software applications to predict much better results without being specifically programmed. The basic premise of machine learning is to build algorithms that receive input data and use statistical analysis to predict output data while output data is updated like many input data become valid. The processes involved in machine learning are similar to the processes of data mining and predictive modeling. Both require searching for certain patterns by date and adjusting program actions accordingly. Many people are also familiar with machine learning from internet shopping and the advertisements that are shown to them depending on what they are buying. This is because referral engines use machine learning to customize ads that are delivered online in near real time. In addition to personalized marketing, other well-known cases in which machine learning is used are fraud detection, spam filtering, threat detection of countries in the network, maintenance, predictability, and building the flow of news.

Machine Learning Algorithms

Machine Learning algorithms (Fig 5.1) are the programs that can learn the hidden patterns from the data, predict the output, and improve the performance from experiences on their own. Different algorithms can be used in machine learning for different tasks, such as simple linear regression that can be used **for prediction problem**s like **stock market prediction,** and **the KNN algorithm can be used for classification problems**.

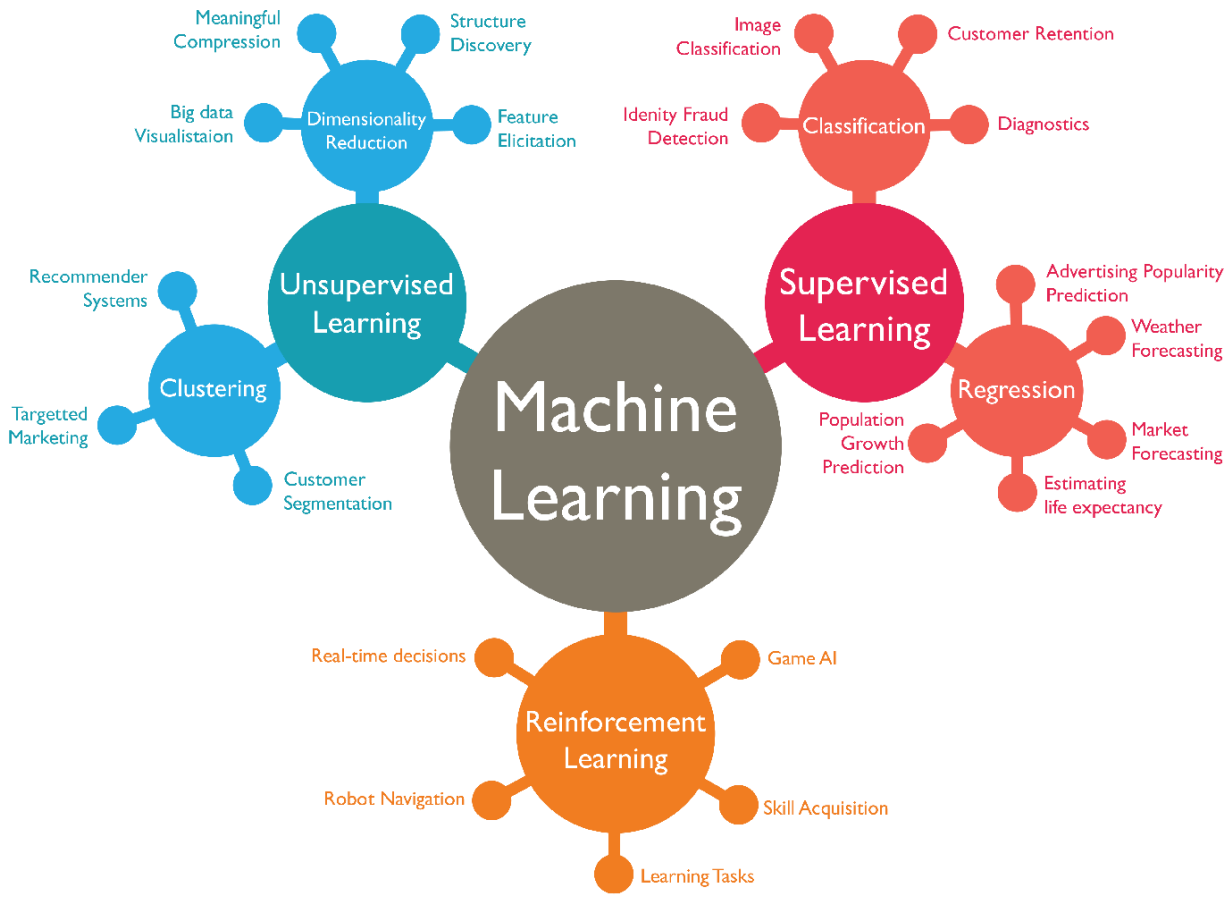

Fig 5.2 Machine Learning and Its Types

Types of Machine Learning Algorithms

Machine Learning Algorithm can be broadly classified into three types (Fig 5.2):

1. **Supervised Learning Algorithms**
2. **Unsupervised Learning Algorithms**
3. **Reinforcement Learning algorithm**

The below diagram illustrates the different ML algorithm, along with the categories:

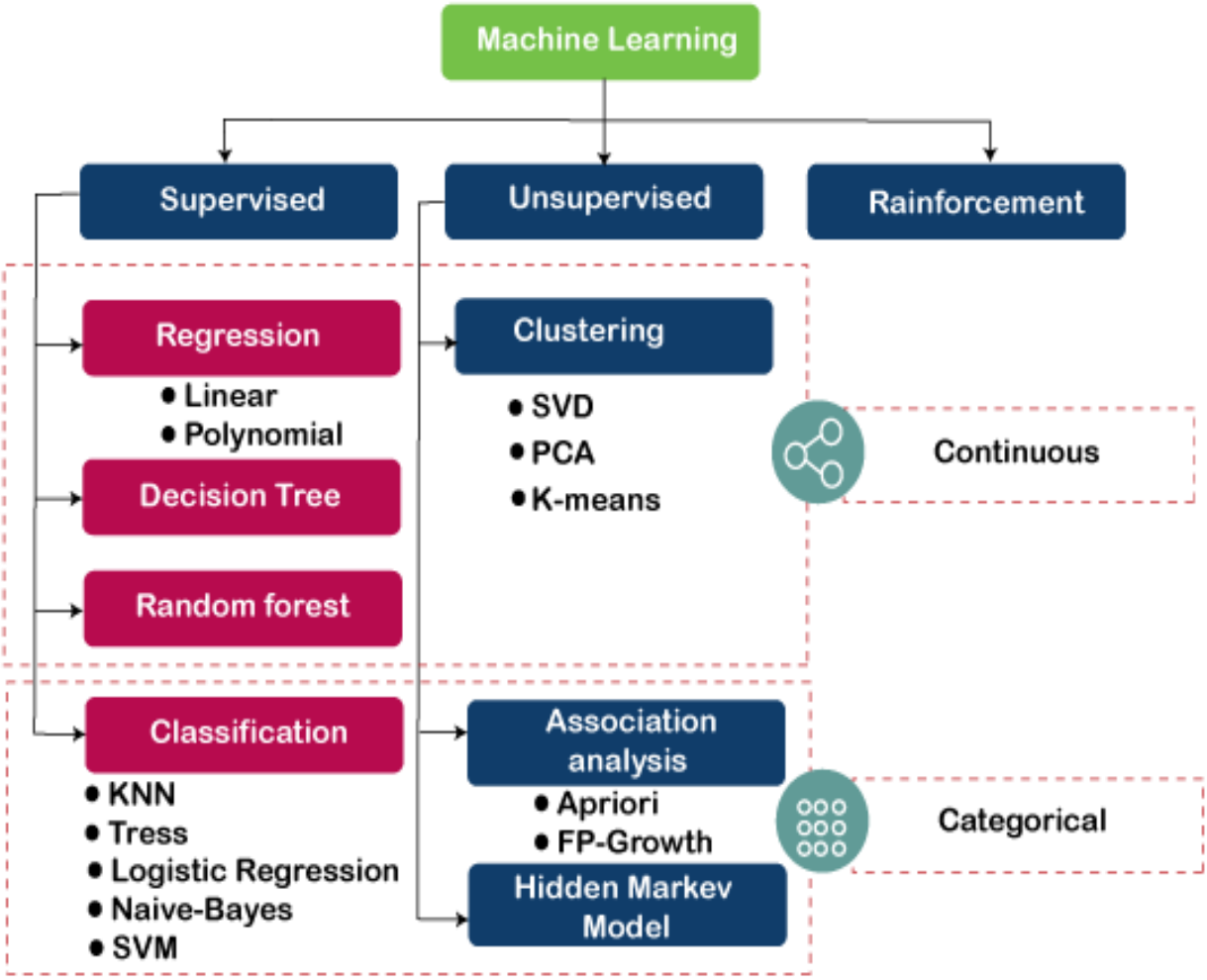

Fig 5.3 Classification of Machine Learning Algorithms

1) Supervised Learning Algorithm

Supervised learning (Fig 5.3) is a type of Machine learning in which the machine needs external supervision to learn. The supervised learning models are trained using the labeled dataset. Once the training and processing are done, the model is tested by providing sample test data to check whether it predicts the correct output.

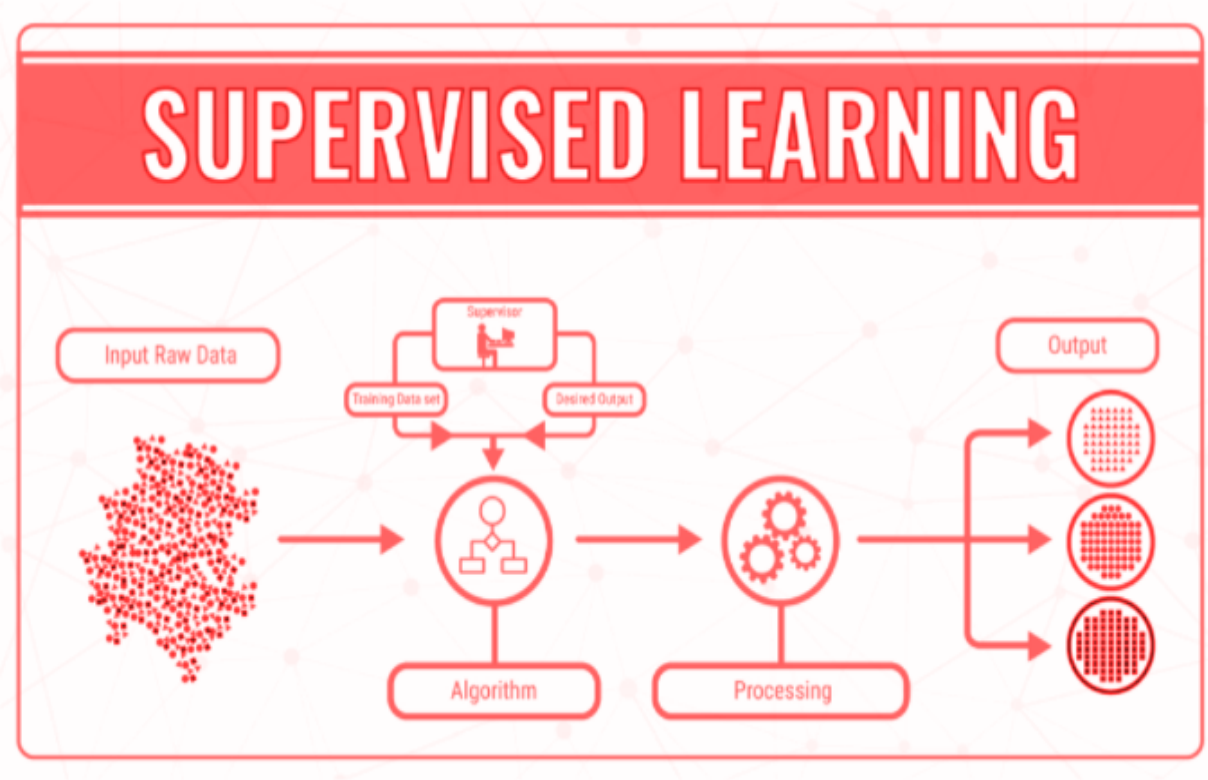

Fig 5.4 Supervised Learning Working

The goal of supervised learning is to map input data with the output data. Supervised learning is based on supervision, and it is the same as when a student learns things under the teacher's supervision. The example of supervised learning is **spam filtering**.

Supervised learning can be divided further into two categories of problem:

- Classification
- Regression

Examples of some popular supervised learning algorithms are Simple Linear regression, Decision Tree, Logistic Regression, KNN algorithm, etc.

2) Unsupervised Learning Algorithm

It is a type of machine learning in which the machine does not need any external supervision to learn from the data, hence called unsupervised learning. The unsupervised models can be

trained using the unlabelled dataset that is not classified, nor categorized, and the algorithm needs to act on that data without any supervision.(Fig 5.4)

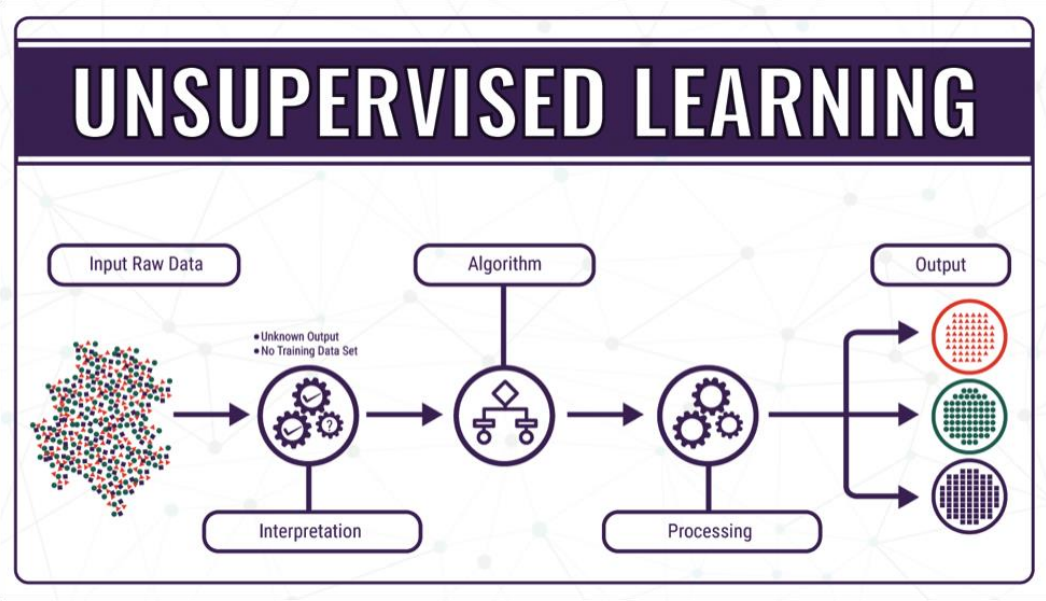

Fig 5.5 Unsupervised Learning Working

In unsupervised learning, the model doesn't have a predefined output, and it tries to find useful insights from the huge amount of data. These are used to solve the Association and Clustering problems**. Hence further, it can be classified into two types:**

- Clustering
- Association

Examples of some Unsupervised learning algorithms are **K-means Clustering, Apriori Algorithm, Eclat, etc.**

3) Reinforcement Learning

In Reinforcement learning (Fig 5.5), an agent interacts with its environment by producing actions, and learn with the help of feedback. The feedback is given to the agent in the form of rewards, such as for each good action, he gets a positive reward, and for each bad action, he gets a negative reward. There is no supervision provided to the agent.

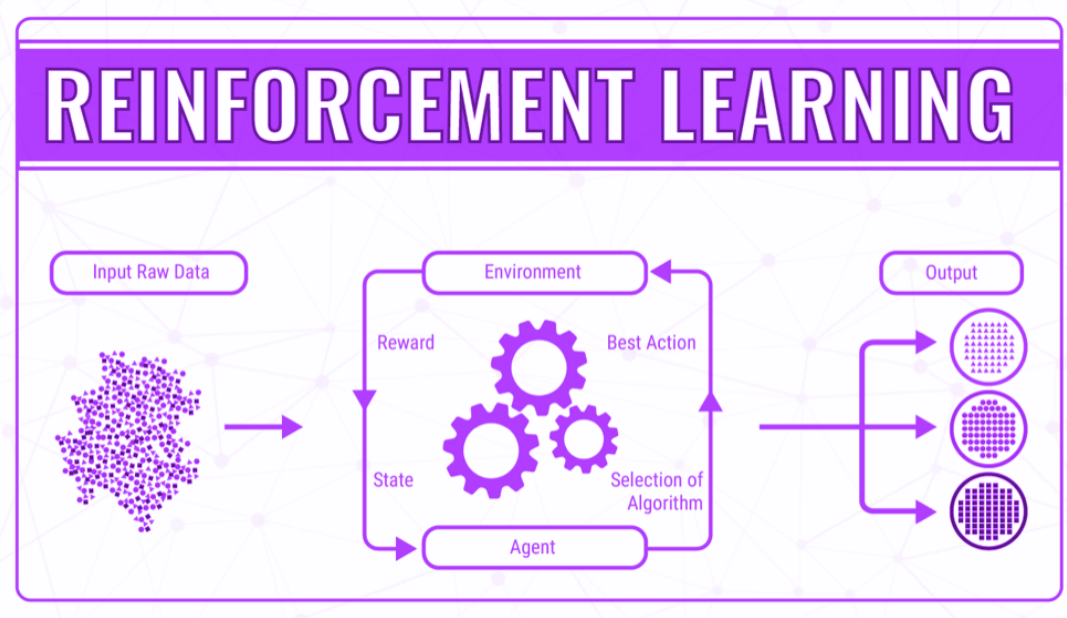

Fig 5.6 Reinforcement Learning Working

The Q-Learning algorithm is used in reinforcement learning.

FIVE ALGORITHMS USED IN THIS PROJECT :

- **Random Forest :**

Random forest (Fig 5.6) is an ensemble of many decision trees. Random forests are built using a method called bagging in which decision trees are used as parallel estimators. If used for a classification problem, the result is based on majority vote of the results received from each decision tree. For regression, the prediction of a leaf node is the mean value of the target values in that leaf. If we use the same or very similar trees, the overall result will not be much different than the result of a single decision tree.

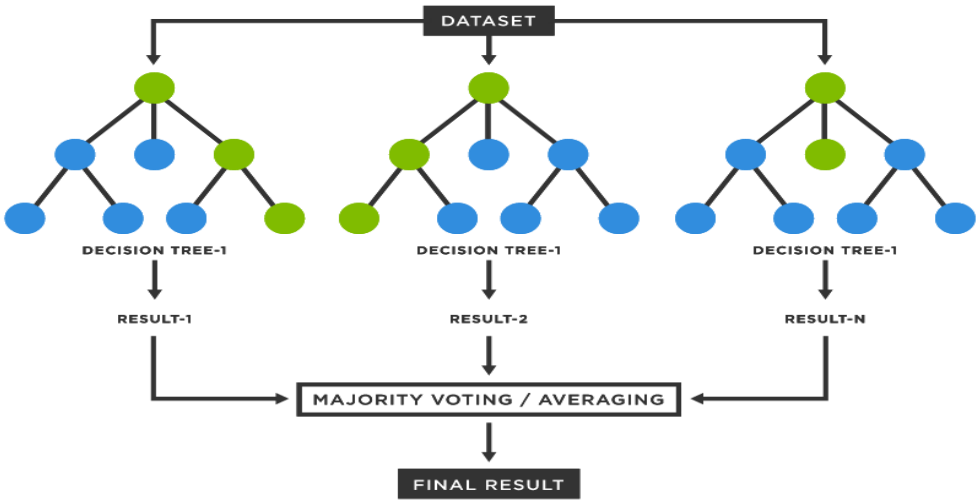

Fig 5.7 Random Forest Working

Random forests achieve to have uncorrelated decision trees by bootstrapping and feature randomness.

Bootstrapping (Fig 5.7) is randomly selecting samples from training data with replacement. They are called bootstrap samples.

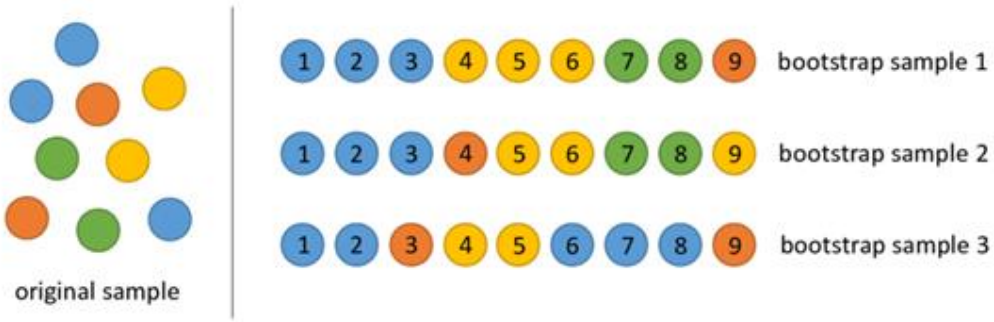

Fig 5.8 Bootstrap samples in Random Forest

Feature randomness (Fig 5.8) is achieved by selecting features randomly for each decision tree in a random forest. The number of features used for each tree in a random forest can be controlled with max_features parameter.

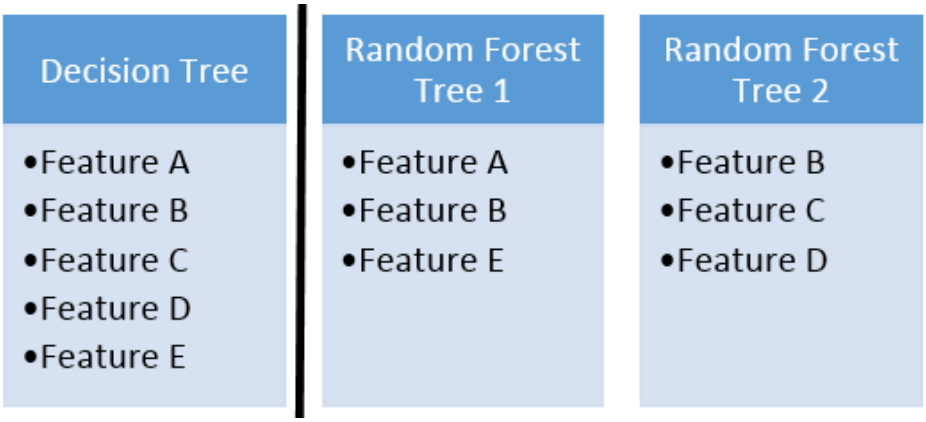

Fig 5.9 Feature randomness in Random Forest

Random forest(Fig 5.9) is a highly accurate model on many different problems and does not require normalization or scaling. However, it is not a good choice for high-dimensional data sets (i.e. text classification) compared to fast linear models (i.e. Naive Bayes).

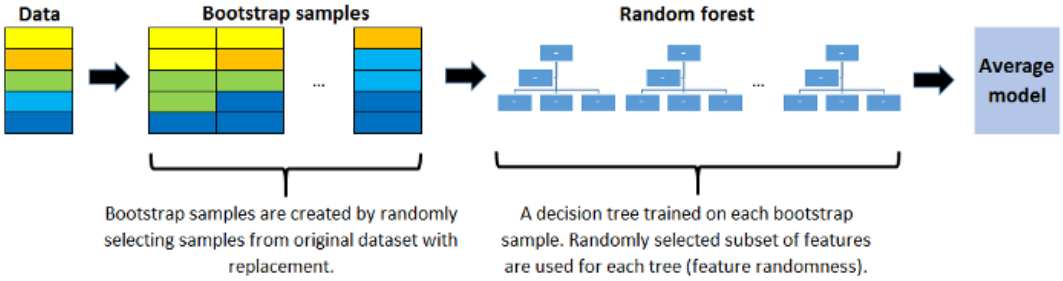

Fig 5.10 Steps in Random Forest Algorithm

- **Gaussian Naive Bayes**

Naive Bayes is a supervised learning algorithm used for classification tasks. Hence, it is also called Naive Bayes Classifier. Naive Bayes assumes that features are independent of each other and there is no correlation between features. However, this is not the case in real life. This naive assumption of features being uncorrelated is the reason why this algorithm is called "naive". The intuition behind naive bayes algorithm is the bayes' theorem: (Fig 5.10)

$$p(A|B) = \frac{p(A).p(B|A)}{p(B)} \qquad (Bayes'\ Theorem)$$

Fig 5.11 Bayes' Theorem

p(A|B): Probability of event A given event B has already occurred

p(B|A): Probability of event B given event A has already occurred

p(A): Probability of event A

p(B): Probability of event B

Naive bayes classifier calculates the probability of a class given a set of feature values (i.e. p(yi | x1, x2 , … , xn)). Input this into Bayes' theorem (Fig 5.11):

$$\mathbf{p(yi \mid x1, x2, \ldots, xn)} = \frac{\mathbf{p(x1, x2, \ldots, xn \mid yi).p(yi)}}{\mathbf{p(x1, x2, \ldots, xn)}}$$

Fig 5.12 Input for Bayes' Theorem

p(x1, x2 , … , xn | yi) means the probability of a specific combination of features (an observation / row in a dataset) given a class label. We need extremely large datasets to have an estimate on the probability distribution for all different combinations of feature values. To overcome this issue, the naive bayes algorithm assumes that all features are independent of each other. Furthermore, denominator (p(x1,x2, … , xn)) can be removed to simplify the equation because it only normalizes the value of conditional probability of a class given an observation (p(yi | x1,x2, … , xn)).

The probability of a class (p(yi)) is very simple to calculate (Fig 5.12):

$$p(yi) = \frac{\text{number of observations with class yi}}{\text{number of all observations}}$$

Fig 5.13 Probability of a Class

Under the assumption of features being independent, p(x1, x2 , … , xn | yi) can be written as:(Fig 5.13)

$$\mathbf{p(x1, x2, \dots, xn \mid yi) = p(x1 \mid yi) \, . p(x2 \mid yi) . \dots . p(xn \mid yi)}$$

Fig 5.14 Conditional Probability Formula

The conditional probability for a single feature given the class label (i.e. p(x1 | yi)) can be more easily estimated from the data. The algorithm needs to store probability distributions of features for each class independently. For example, if there are 5 classes and 10 features, 50 different probability distributions need to be stored.

Adding all these up, it became an easy task for naive bayes algorithm to calculate the probability to observe a class given values of features (p(yi | x1, x2 , … , xn))

The assumption that all features are independent makes naive bayes algorithm very fast compared to complicated algorithms. In some cases, speed is preferred over higher accuracy. On the other hand, the same assumption makes naive Bayes algorithms less accurate than complicated algorithms.

- **Adaptive Boosting**

AdaBoost is implemented by combining several weak learners into a single strong learner. The weak learners in AdaBoost take into account a single input feature and draw out a single split decision tree called the decision stump. Each observation is weighed equally while drawing out the first decision stump.The results from the first decision stump are analyzed and if any observations are wrongfully classified, they are assigned higher weights. Post this, a new decision stump is drawn by considering the observations with higher weights as more significant. Again, if any observations are misclassified, they’re given higher weight and this process continues until all the observations fall into the right class. Adaboost can be used for both classification and regression-based problems, however, it is more commonly used for classification purpose.(5.14)

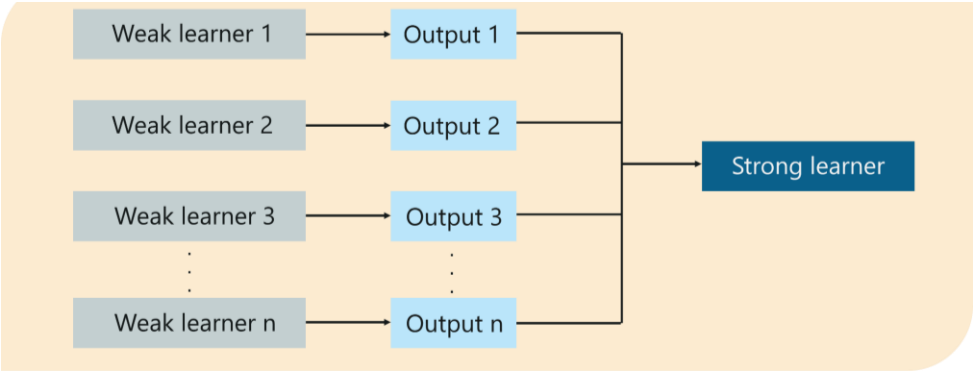

Fig 5.15 Working of Boosting Algorithms (Adaptive Boosting, Gradient Boosting)

- **Gradient Boosting**

Gradient Boosting is also based on sequential ensemble learning. Here the base learners are generated sequentially in such a way that the present base learner is always more effective than the previous one, i.e., the overall model improves sequentially with each iteration.Gradient Boosting method tries to optimize the loss function of the previous learner by adding a new model that adds weak learners in order to reduce the loss function. The main idea here is to overcome the errors in the previous learner's predictions. This type of boosting has three main components:

1. Loss function that needs to be ameliorated.
2. Weak learner for computing predictions and forming strong learners.
3. An Additive Model that will regularize the loss function.

Like AdaBoost, Gradient Boosting can also be used for both classification and regression problems.

- **Decision Trees**

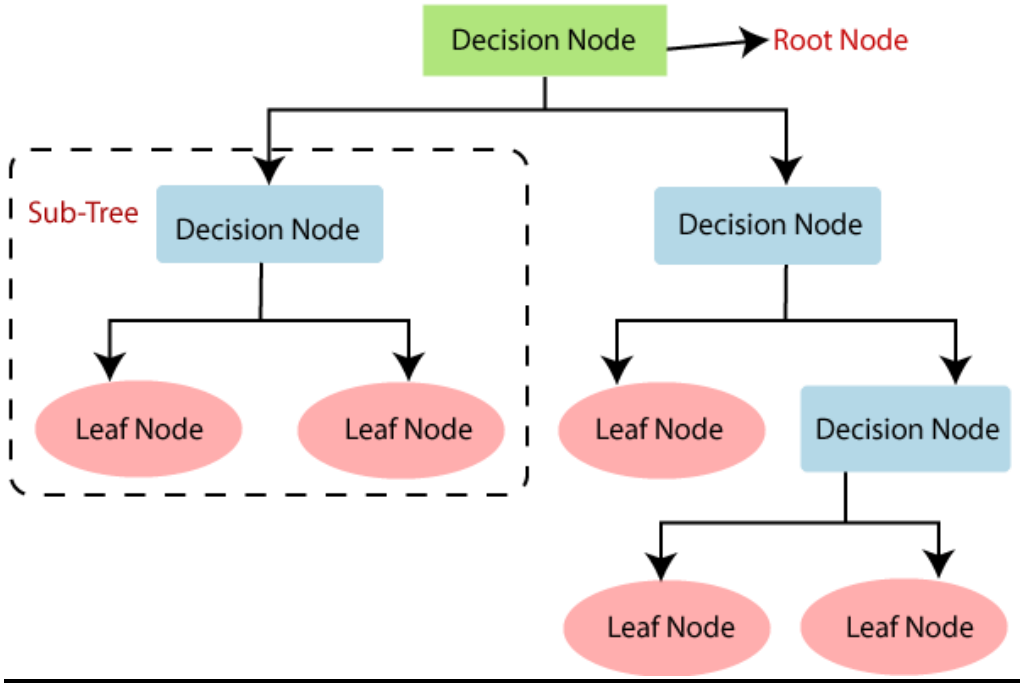

Fig 5.16 Decision Tree Structure

A decision tree (Fig 5.15) is a map of the possible outcomes of a series of related choices. It allows an individual or organization to weigh possible actions against one another based on their costs, probabilities, and benefits.

- **K - Nearest Neighbor**

The K Nearest Neighbor algorithm falls under the Supervised Learning category and is used for classification (most commonly) and regression. It is a versatile algorithm also used for imputing missing values and resampling datasets. As the name (K Nearest Neighbor) suggests it considers K Nearest Neighbors (Data points) to predict the class or continuous value for the new Datapoint.

The algorithm's learning is:

- Instance-based learning: Here we do not learn weights from training data to predict output (as in model-based algorithms) but use entire training instances to predict output for unseen data.

- Lazy Learning: Model is not learned using training data prior and the learning process is postponed to a time when prediction is requested on the new instance.
- Non -Parametric: In KNN, there is no predefined form of the mapping function.

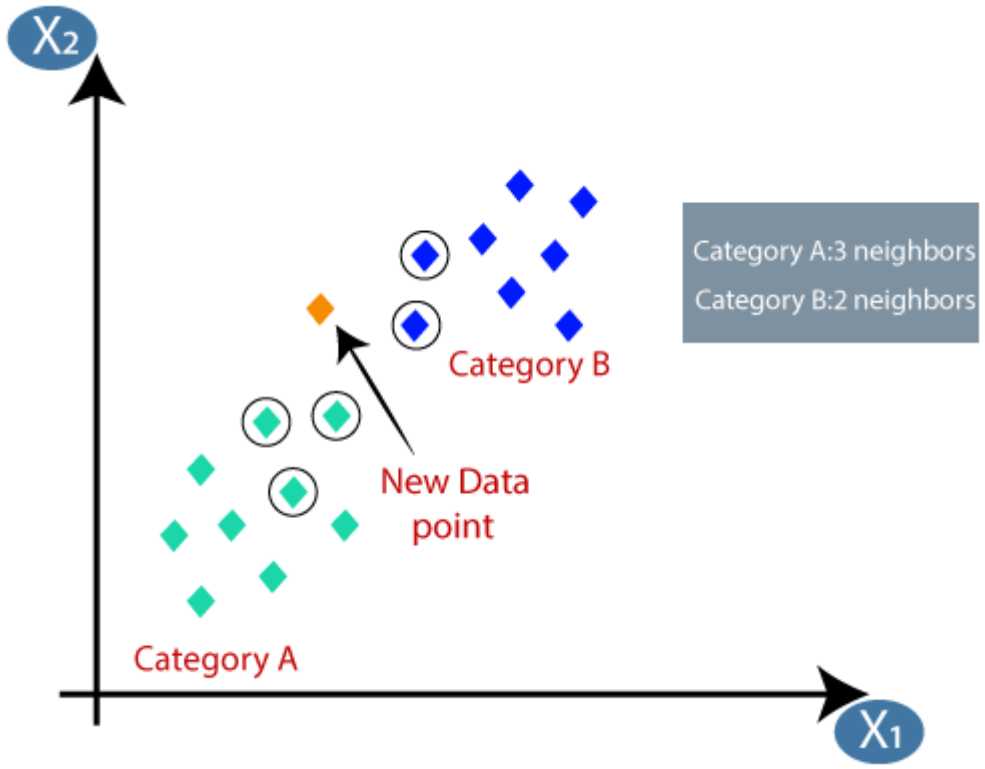

Fig 5.17 KNN Algorithm

To select the K that's right for your data, we run the KNN algorithm several times with different values of K and choose the K that reduces the number of errors we encounter while maintaining the algorithm's ability to accurately make predictions when it's given data it hasn't seen before. Here are some things to keep in mind:

1. As we decrease the value of K to 1, our predictions become less stable. Just think for a minute, imagine K=1 and we have a query point surrounded by several reds and one green (I'm thinking about the top left corner of the colored plot above), but the green is the single nearest neighbor. Reasonably, we would think the query point is most likely red, but because K=1, KNN incorrectly predicts that the query point is green.

2. Inversely, as we increase the value of K, our predictions become more stable due to majority voting / averaging, and thus, more likely to make more accurate predictions (up to a certain point). Eventually, we begin to witness an increasing number of errors. It is at this point we know we have pushed the value of K too far.
3. In cases where we are taking a majority vote (e.g., picking the mode in a classification problem) among labels, we usually make K an odd number to have a tiebreaker.

5.1.1.2 WEB DEVELOPMENT (FRONTEND AND BACKEND):

Web development refers to the building, creating, and maintaining of websites. It includes aspects such as web design, web publishing, web programming, and database management. It is the creation of an application that works over the internet i.e., websites.

The word Web Development is made up of two words, that is:

- **Web:** It refers to websites, web pages or anything that works over the internet.
- **Development:** Building the application from scratch.

Web Development can be classified into two ways:

- **Frontend Development (Fig 5.18)**
- **Backend Development (Fig 5.19)**

Frontend Development: The part of a website that the user interacts directly is termed as front end. It is also referred to as the ‘client side’ of the application.

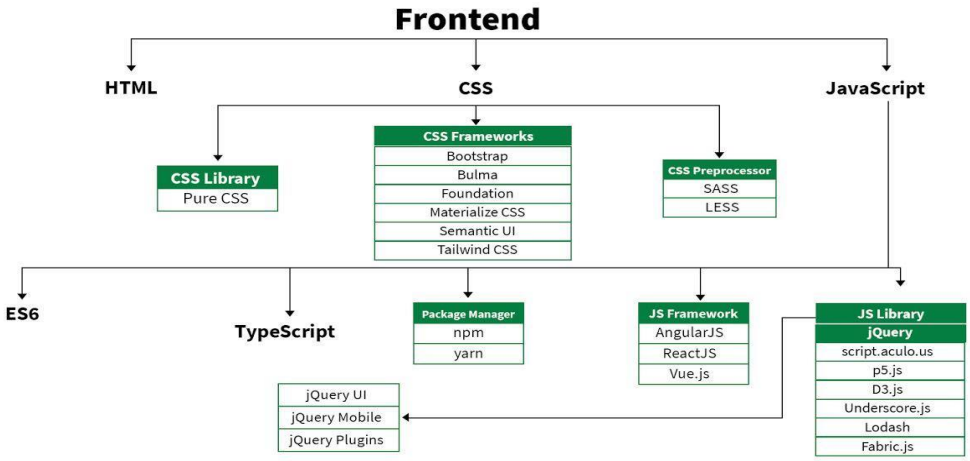

Fig 5.18 Frontend 'Client side of application'

Backend Development: Backend is the server side of a website. It is the part of the website that users cannot see and interact with. It is the portion of software that does not come in direct contact with the users.

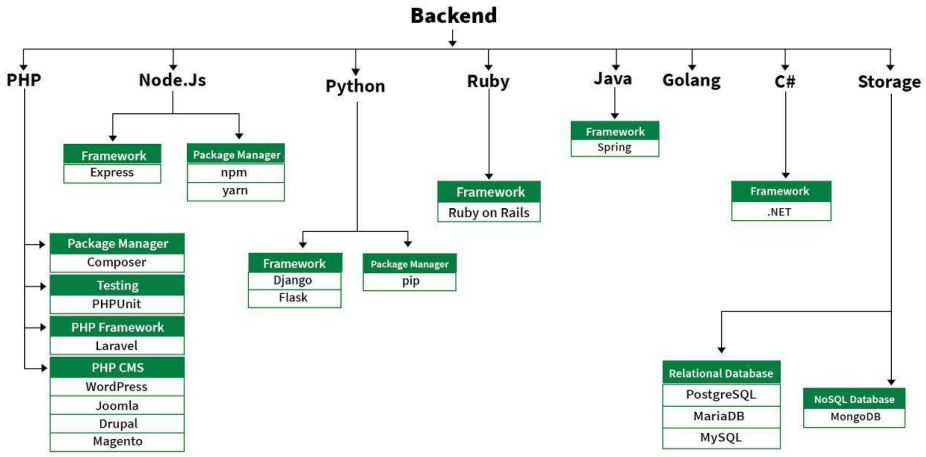

Fig 5.19 Backend 'Server side of application'

5.1.1.3 DATABASE (POSTGRESQL)

PostgreSQL is a powerful, open-source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for

reliability, data integrity, and correctness. PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows.

Below are the features of PostgreSQL:

- It is written in C programming language.
- PostgreSQL is cross-platform and runs on various operating systems such as Microsoft Windows, UNIX, FreeBSD, Mac OS X, Solaris, HP-UX, LINUX, and so on.
- PostgreSQL is also pronounced as Post-gress-Q-L, which is developed by the PostgreSQL Global Development Group (a worldwide team of volunteers); any organization or other private entity does not control it.
- PostgreSQL will offer us the facility to add custom functions with the help of various programming languages such as Java, C, and C++, etc.

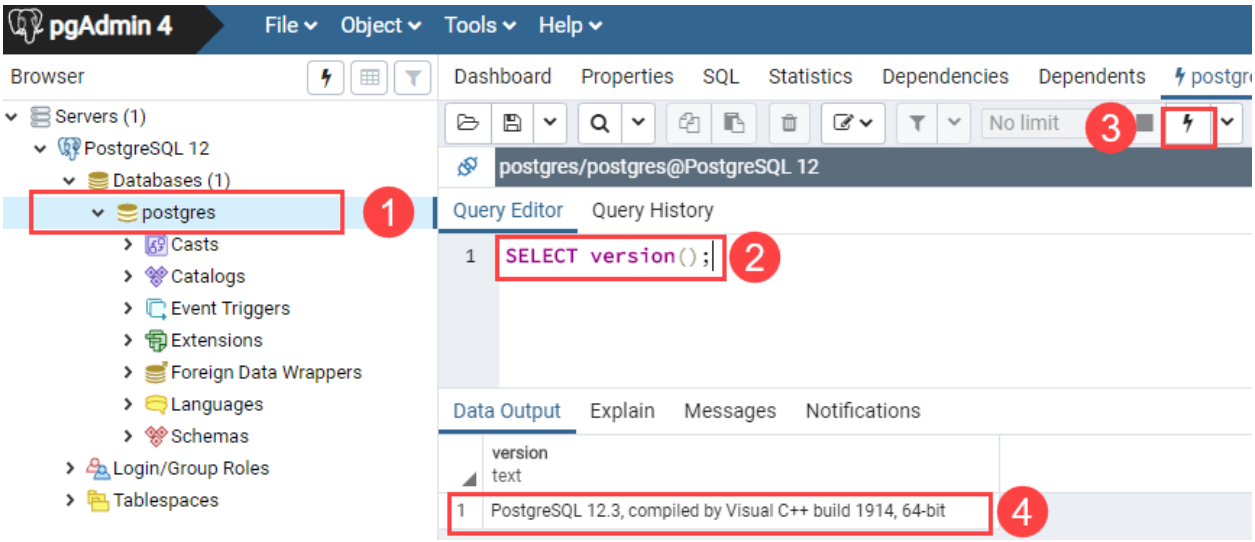

Fig 5.20 PostgreSQL Interface

How PostgreSQL Works

PostgreSQL uses a client-server model where the client and the server can reside on different hosts in a networked environment. The server program manages the database files, accepts connections to the database from client applications. It can handle multiple concurrent connections from clients by "forking" a new process for each connection. It executes database requests from clients and sends the results back to the clients. Remote clients can connect over the network or internet to the server.

Valid client programs include text-oriented tools that ship with PostgreSQL, a graphical tool, or applications developed using other programming languages.

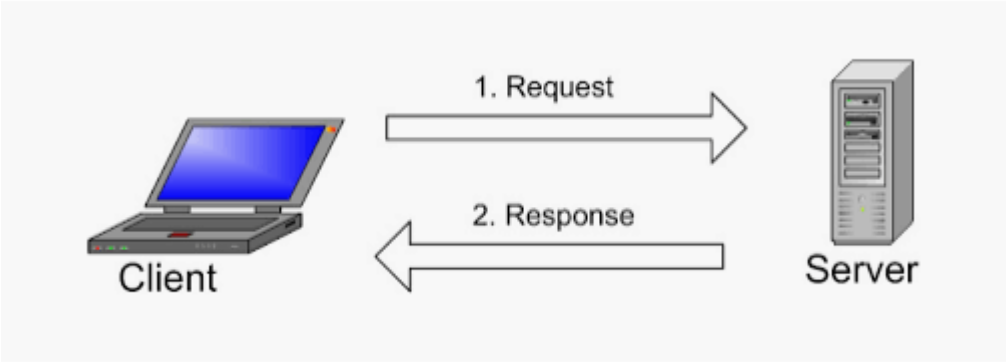

Fig 5.21 Working of PostgreSQL

5.1.2.1 FOR MACHINE LEARNING:

- Python

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently whereas other languages use punctuation, and it has fewer syntactic constructions than other languages.

- **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

Python Features

Python's features include −

- **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.

- **Scikit Learn**

Scikit-learn (Sklearn) is the most useful and robust library for machine learning in Python. It provides a selection of efficient tools for machine learning and statistical modelling including classification, regression, clustering, and dimensionality reduction via a consistent interface in Python. This library, which is largely written in Python, is built upon NumPy, SciPy and Matplotlib.

5.1.2.2 FOR FRONTEND AND BACKEND INTERFACE:

- **Django - Web Development Framework**

  Django is a web development framework that assists in building and maintaining quality web applications. Django helps eliminate repetitive tasks making the development process an easy and time saving experience.

  The Project Structure

  The “myproject” folder is just our project container, it actually contains two elements –

  - manage.py − This file is kind of your project local django-admin for interacting with your project via command line (start the development server, sync db...). To get a full list of command accessible via manage.py you can use the code −

    $ python manage.py help

  - The “myproject” subfolder − This folder is the actual python package of your project. It contains four files −
    - __init__.py − Just for python, treat this folder as a package.
    - settings.py − As the name indicates, your project settings.
    - urls.py − All links of your project and the function to call. A kind of ToC of your project.
    - wsgi.py − If you need to deploy your project over WSGI.

- **HTML**

  Hypertext Markup Language (HTML) is the standard markup language for documents designed to be displayed in a web browser.

- **CSS**

  Cascading Style Sheets (CSS) is a style sheet language used for describing the presentation of a document written in a markup language like HTML. CSS is a cornerstone technology of the World Wide Web, alongside HTML and JavaScript.

  CSS is designed to enable the separation of presentation and content, including layout, colors, and fonts.

- **BOOTSTRAP - Open Source CSS Framework**

  Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components.

- **jQuery - JavaScript Library**

  jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript.

- **Jinja**

  Jinja2 is a modern and designer-friendly templating language for Python, modelled after Django’s templates. It is fast, widely used and secure with the optional sandboxed template execution environment.

- **Werkzeug**

  Werkzeug is a comprehensive WSGI web application library. It began as a simple collection of various utilities for WSGI applications and has become one of the most advanced WSGI utility libraries.

5.1.2.3 FOR DATABASE STORAGE:

- **PostgreSQL**

  PostgreSQL is a powerful, open-source object-relational database system. It has more than 15 years of active development and a proven architecture that has earned it a strong reputation for reliability, data integrity, and correctness. PostgreSQL runs on all major operating systems, including Linux, UNIX (AIX, BSD, HP-UX, SGI IRIX, Mac OS X, Solaris, Tru64), and Windows.

  PostgreSQL can be integrated with Python using psycopg2 module. psycopg2 is a PostgreSQL database adapter for the Python programming language. psycopg2 was written with the aim of being very small and fast, and stable as a rock.

  We can use pip command to install it as follows:

  psycopg2-binary==2.8.6

- **Procedural Language Support**

  PostgreSQL supports four standard procedural languages, which allows the users to write their own code in any of the languages and it can be executed by PostgreSQL database server. These procedural languages are - PL/pgSQL, PL/Tcl, PL/Perl and PL/Python. Besides, other non-standard procedural languages like PL/PHP, PL/V8, PL/Ruby, PL/Java, etc., are also supported.

- **Connecting to Database**

The following Python code shows how to connect to an existing database. If the database does not exist, then it will be created and finally a database object will be returned.

```
import psycopg2
conn = psycopg2.connect(database="testdb", user = "postgres", password =
"pass123", host = "127.0.0.1", port = "5432")
print "Opened database successfully"
```

5.2 SOFTWARE IMPLEMENTATION

5.2.1 PYTHON SCRIPTS

There will be an ipython file named Disease prediction that would be used for building disease prediction models from the given dataset.

5.2.1.1 IMPORTING MODULES

Using Pip to install all the packages required for the project.

py -m pip install [options] <requirement specifier> [package-index-options]

Then importing modules into files (Fig. 5.21)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix
pd.set_option('max_columns', None)
```

Fig 5.22 importing modules

5.2.1.2 HANDLING DATA

Our dataset includes two files, one is training dataset that has 4920 rows and 132 columns and the other one being testing dataset containing 41 rows and 132 columns separated by'|' respectively.

Refer to Figure 5.22 and Fig. 5.23

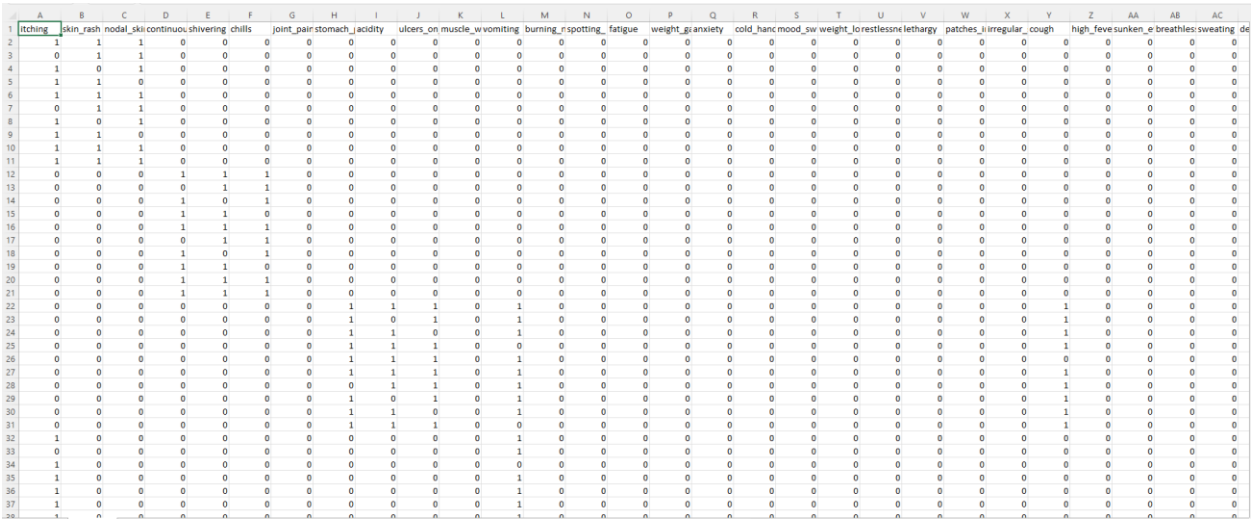

Fig 5.23 Training.csv

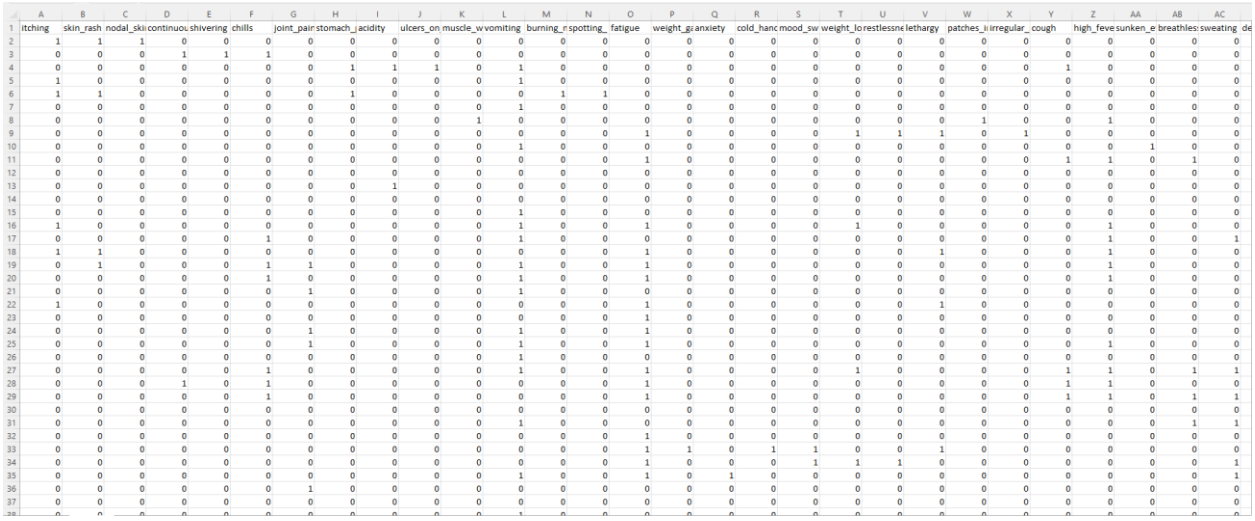

Fig 5.24 Testing.csv

To train the data in supervised learning we first use the pandas module to load the .csv file into our VScode IDE. (Fig 5.24)

```
[2]  train = pd.read_csv('Training.csv')
     test = pd.read_csv('Testing.csv')
     ✓ 0.7s
```

Fig 5.25 Loading Dataset

As Less than 1% for test is too small, we did our own train-test split,

```
data = pd.concat([train, test])
✓ 0.7s
```

Fig 5.26 Concatenating data

We found out that fatigue and vomiting are the two most common and most generic symptoms in this dataset and probably won't be an unique/significant predictor for an illness.

```
data_X = {'Symptoms': [], 'Prognosis': [], 'length': []}
table = pd.DataFrame(data_X)
table = table.astype({"Symptoms": str, "Prognosis": object, 'length': int})
i = 0

for symp in sorted(data.columns.tolist()[:-1]):
    prognosis = data[data[symp] == 1].prognosis.unique().tolist()
    table = table.append({'Symptoms': symp}, {'Prognosis': prognosis}, {'length':len(prognosis)})
    table.at[i,'Prognosis'] = prognosis
    table.at[i, 'length'] = len(prognosis)
    i += 1

table.sort_values(by='length', ascending=False).head(10)
```

Fig 5.27 Common symptoms

5.2.1.3 FEATURE SELECTION

We first find the measure of how each important each feature is (regardless of class) and storing them in a variable ‘coefficients’ and then create an importance threshold by creating first a new plotly figure (Bar chart in this case) , giving it a horizontal orientation and include the coefficients on the x-axis and on the y-axis we include the column names of the data.

The Bar chart is shown in Fig. 5.27 below

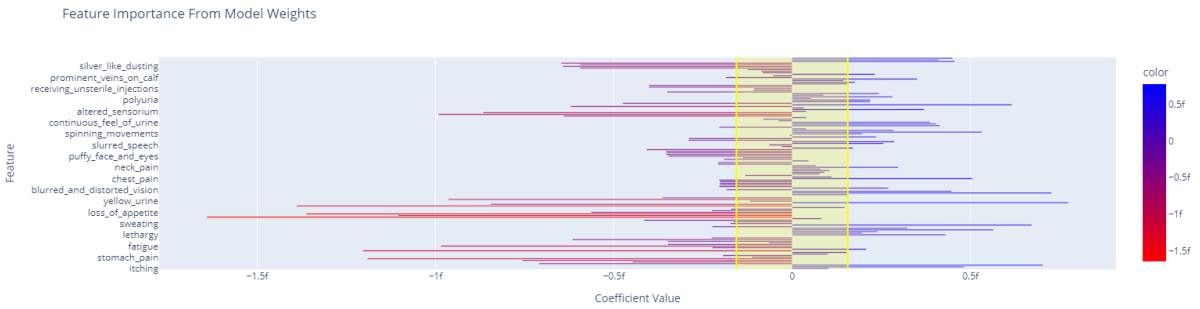

Fig 5.28 Bar-chart for Feature importance

From the graph, under the yellow section are the features that we will remove, we then get the low importance features by taking the absolute value of coefficients and checking if those values are below the threshold value that was calculated before.

```
low_importance_features = X_train.columns[np.abs(coefficients) < importance_threshold]
low_importance_features
```

Fig 5.29 Finding low-importance features

Low importance features:

'continuous_sneezing', 'shivering', 'chills', 'stomach_pain', 'spotting_ urination', 'weight_gain', 'cold_hands_and_feets', 'mood_swings', 'weight_loss', 'sunken_eyes', 'dehydration', 'indigestion', 'back_pain', 'constipation', 'diarrhoea', 'fluid_overload', 'swelling_of_stomach', 'pain_during_bowel_movements', 'pain_in_anal_region', 'bloody_stool', 'irritation_in_anus', 'puffy_face_and_eyes', 'enlarged_thyroid', 'brittle_nails', 'swollen_extremeties', 'movement_stiffness', 'toxic_look_(typhos)', 'belly_pain', 'abnormal_menstruation', 'watering_from_eyes', 'family_history', 'distention_of_abdomen', 'history_of_alcohol_consumption', 'fluid_overload.1', 'pus_filled_pimples', 'blackheads', 'scurring'

5.2.1.4 TRAINING ON REDUCED DATA

We then find out the accuracy using the reduced data i.e data without the low importance features

```
reduced_data = data.drop(low_importance_features, axis=1).copy()

X_train, X_test, y_train, y_test = preprocess_inputs(reduced_data)
```

Fig 5.30 Getting reduced dataset

	itching	skin_rash	nodal_skin_eruptions	joint_pain	acidity	ulcers_on_tongue	muscle_wasting	vomiting	burning_micturition	fatigue	anxiety	restlessness	lethargy	patches_in_throat	irregular_sugar_level	c
1237	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	
1565	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	
1238	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	
1577	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
1987	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
2895	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	
2763	1	1	0	0	0	0	0	0	0	1	0	0	1	0	0	
905	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
3980	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	
235	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	

3472 rows × 95 columns

Fig 5.31 Reduced Dataset

5.2.1.5 SAVING THE TRAINED MODEL

Once we build our model using the algorithm “Decision Tree”, we then use the joblib to save our model as ‘trained_model’ into the project directory.

```
import joblib as jb

jb.dump(clf,'trained_model')
```

Fig 5.32 Saving trained model

Joblib works especially well with NumPy arrays which are used by sklearn so depending on the classifier type you use you might Have performance and size benefits using

joblib.Otherwise pickle does work correctly so saving a trained classifier and loading it again will produce the same results no matter which of the serialization libraries we use.

5.2.2 DJANGO SCRIPTS

5.2.2.1 main_app

First writing all the necessary import files, we then run our main app for creating Web Interface

```
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.http import JsonResponse
from datetime import date

from django.contrib import messages
from django.contrib.auth.models import User , auth
from .models import patient , doctor , diseaseinfo , consultation ,rating_review
from chats.models import Chat,Feedback
```

Fig 5.33 Django imports

Loading the trained model so that it could be used to further prediction in the application:

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

Fig 5.34 Loading Machine Learning 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')

def admin_ui(request):

    if request.method == 'GET':
      if request.user.is_authenticated:
        auser = request.user
        Feedbackobj = Feedback.objects.all()
        return render(request,'admin/admin_ui/admin_ui.html' , {"auser":auser,"Feedback":Feedbackobj})
      else :
        return redirect('home')
    if request.method == 'POST':
      return render(request,'patient/patient_ui/profile.html')
```

```python
def patient_ui(request):
    if request.method == 'GET':
      if request.user.is_authenticated:
        patientusername = request.session['patientusername']
        puser = User.objects.get(username=patientusername)
        return render(request,'patient/patient_ui/profile.html' , {"puser":puser})
      else :
        return redirect('home')
    if request.method == 'POST':
      return render(request,'patient/patient_ui/profile.html')

def pviewprofile(request, patientusername):
    if request.method == 'GET':
          puser = User.objects.get(username=patientusername)
          return render(request,'patient/view_profile/view_profile.html', {"puser":puser})
```

```python
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']

  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']
```

```
alphabaticsymptomslist = sorted(symptomslist)

if request.method == 'GET':
    return render(request,'patient/checkdisease/checkdisease.html', {"list2":alphabaticsymptomslist})

elif request.method == 'POST':
    ## access you data by playing around with the request.POST object
    inputno = int(request.POST["noofsym"])
    print(inputno)
    if (inputno == 0 ) :
        return JsonResponse({'predicteddisease': "none",'confidencescore': 0 })
    else :
      psymptoms = []
      psymptoms = request.POST.getlist("symptoms[]")
      print(psymptoms)
      """      #main code start from here...
      """
      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]

      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)


    #saving to database.....................

    patient = puser.patient
    diseasename = predicted_disease
    no_of_symp = inputno
    symptomsname = psymptoms
    confidence = confidencescore

    diseaseinfo_new = diseaseinfo(patient=patient,diseasename=diseasename,no_of_symp=no_of_symp,symptomsname=symptomsname,confidence=confidence,consultdoctor=consultdoctor)
    diseaseinfo_new.save()


    request.session['diseaseinfo_id'] = diseaseinfo_new.id

    print("disease record saved sucessfully.............................")

    return JsonResponse({'predicteddisease': predicted_disease ,'confidencescore':confidencescore , "consultdoctor": consultdoctor})
```

```
def pconsultation_history(request):

    if request.method == 'GET':
      patientusername = request.session['patientusername']
      puser = User.objects.get(username=patientusername)
      patient_obj = puser.patient
      consultationnew = consultation.objects.filter(patient = patient_obj)
      return render(request,'patient/consultation_history/consultation_history.html',{"consultation":consultationnew})

def dconsultation_history(request):

    if request.method == 'GET':
      doctorusername = request.session['doctorusername']
      duser = User.objects.get(username=doctorusername)
      doctor_obj = duser.doctor
      consultationnew = consultation.objects.filter(doctor = doctor_obj)
      return render(request,'doctor/consultation_history/consultation_history.html',{"consultation":consultationnew})

def doctor_ui(request):
    if request.method == 'GET':
      doctorid = request.session['doctorusername']
      duser = User.objects.get(username=doctorid)
      return render(request,'doctor/doctor_ui/profile.html',{"duser":duser})

def dviewprofile(request, doctorusername):

    if request.method == 'GET':


        duser = User.objects.get(username=doctorusername)
        r = rating_review.objects.filter(doctor=duser.doctor)

        return render(request,'doctor/view_profile/view_profile.html', {"duser":duser, "rate":r} )

def  consult_a_doctor(request):

    if request.method == 'GET':
        doctortype = request.session['doctortype']
        print(doctortype)
        dobj = doctor.objects.all()

        return render(request,'patient/consult_a_doctor/consult_a_doctor.html',{"dobj":dobj})
```

```
def  make_consultation(request, doctorusername):

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

        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)

def  consultationview(request,consultation_id):

    if request.method == 'GET':


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

      return render(request,'consultation/consultation.html', {"consultation":consultation_obj })
def rate_review(request,consultation_id):
    if request.method == "POST":

            consultation_obj = consultation.objects.get(id=consultation_id)
            patient = consultation_obj.patient
            doctor1 = consultation_obj.doctor
            rating = request.POST.get('rating')
            review = request.POST.get('review')

            rating_obj = rating_review(patient=patient,doctor=doctor1,rating=rating,review=review)
            rating_obj.save()

            rate = int(rating_obj.rating_is)
            doctor.objects.filter(pk=doctor1).update(rating=rate)


            return redirect('consultationview',consultation_id)

def close_consultation(request,consultation_id):
    if request.method == "POST":

            consultation.objects.filter(pk=consultation_id).update(status="closed")

            return redirect('home')
```

Fig 5.35 main_app

Chatting System is designed by the following code:

```
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)
        if msg != '':
            c.save()
            print("msg saved"+ msg )
            return JsonResponse({ 'msg': msg })
```

```python
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})
```

Fig 5.36 Chatting system

5.2.2.2 accounts

```python
from django.shortcuts import render, redirect
from django.http import HttpResponse
from django.http import HttpResponseRedirect, JsonResponse
from django.views.decorators.csrf import csrf_exempt
from django.contrib import messages
from django.contrib.auth.models import User , auth
from main_app.models import patient , doctor
from datetime import datetime
```

```python
def logout(request):
    auth.logout(request)
    request.session.pop('patientid', None)
    request.session.pop('doctorid', None)
    request.session.pop('adminid', None)
    return render(request,'homepage/index.html')

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')
```

```
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['mc

        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')
```

```
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 :
      return render(request,'patient/signin_page/index.html')
```

```
def savepdata(request,patientusername):

  if request.method == 'POST':
    name =  request.POST['name']
    dob =  request.POST['dob']
    gender =  request.POST['gender']
    address =  request.POST['address']
    mobile_no = request.POST['mobile_no']
    print(dob)
    dobdate = datetime.strptime(dob,'%Y-%m-%d')

    puser = User.objects.get(username=patientusername)

    patient.objects.filter(pk=puser.patient).update(name=name,dob=dobdate,gender=gender,address=address,mobile_no=mobile_no)

    return redirect('pviewprofile',patientusername)
```

```python
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['mo

            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')
            password1 =  request.POST.get('password1')
```

```python
            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_re
                    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')
```

```python
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')
            else :
                messages.info(request,'invalid credentials')
                return redirect('sign_in_doctor')
    else :
        return render(request,'doctor/signin_page/index.html')
```

Fig 5.37 accounts webapp

5.2.2 SETTING POSTGRESQL DATABASE

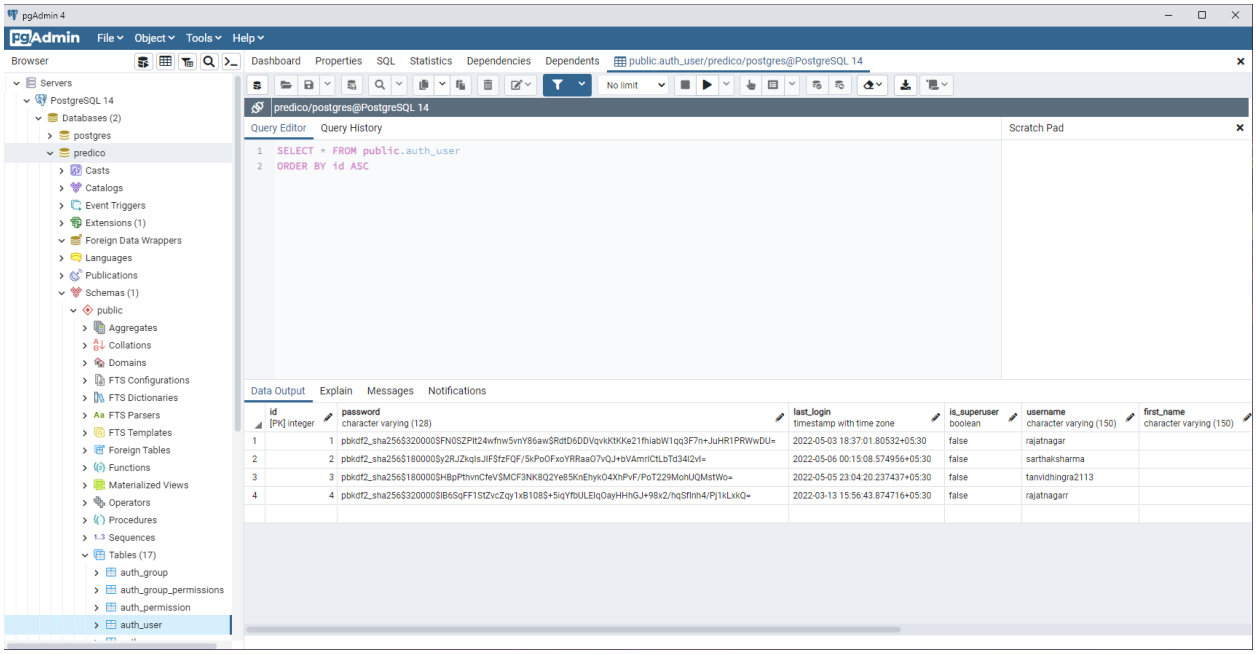

Fig 5.38 Setting up PostgreSQL

```
import os

# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.2/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'v3v5vfsn0xxjtmb=eoawoiw$5br4g0r&jy_l39995h_93l+-z5'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

ALLOWED_HOSTS = []


# Application definition

INSTALLED_APPS = [
    'chats.apps.ChatsConfig',
    'accounts.apps.AccountsConfig',
    'main_app.apps.MainAppConfig',
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'disease_prediction.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,'templates')],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'disease_prediction.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.2/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'predico',
        'USER': 'postgres',
        'PASSWORD': 'tiger',
        'HOST': 'localhost'
    }
}
```

```
AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.2/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True

USE_TZ = True


# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/

STATIC_URL = '/static/'
STATICFILES_DIRS = [

    os.path.join(BASE_DIR,'templates')
]
```

Fig 5.39 Installing PostgreSQL in web-app

5.2.3 TEMPLATES

There are six template files used in the project

- admin
- consultation

- doctor
- homepage
- patient
- signin_page

5.2.3.1 admin

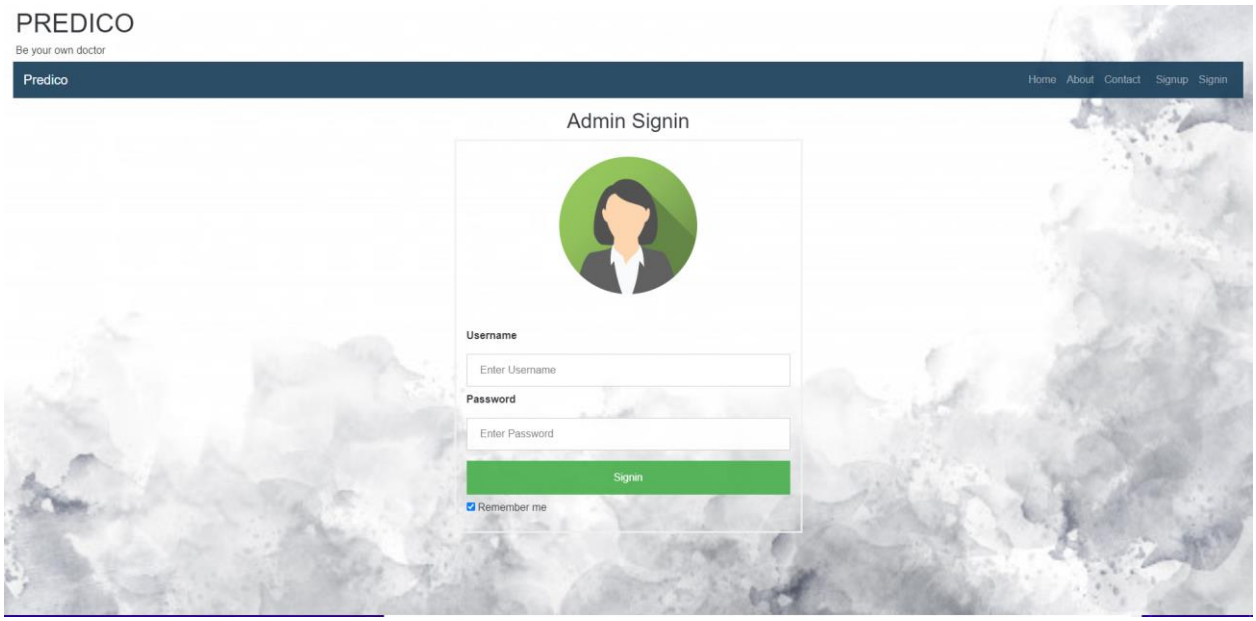

Fig 5.40 admin design page

```
{% extends "basic.html" %}
{% load static %}

{% block head %}
<style>
  #box {
    padding-left: 15%;
    padding-right: 15%;
  }
</style>
{% endblock %}
{% block body %}
<br>
<div id="box" class="container mt-5 mb-5">
  <h2>Admin - {{auser.username}}</h2>
  <div class="col">
    <div class="row">
      <a id="links" class="btn btn-outline-info btn-block" href="{% url 'admin:index' %}">Manage user's data</a><br>
    </div>
    <div class="row">
      <button class="btn btn-outline-info btn-block" data-toggle="modal" data-target="#myModal2">
        View user's feedback
      </button>
    </div>
  </div>
```

```
<!-- The Modal -->
<div class="modal fade" id="myModal2">
  <div class="modal-dialog modal-xl">
    <div class="modal-content">
      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Feedback's</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>
      <!-- Modal body -->
      <div class="modal-body">
        {% for i in Feedback %}

        <div class="row" style="border-bottom: 1px solid #ccc;">

          <div class="col">
            <p class=""><mark>Date created</mark>: {{i.created}}</p>
          </div>
          <div class="col">
            <p class=""><mark>Feedback</mark>: {{i.feedback}}</p>
          </div>
          <div class="col">
            <p class=""><mark>Sender</mark>: {{i.sender.username}}</p>
          </div>
        </div>
        <br>
        {% endfor %}
      </div>
    </div>
  </div>
</div>

{% endblock %}
```

Fig 5.41 admin Interface file

5.2.3.2 consultation

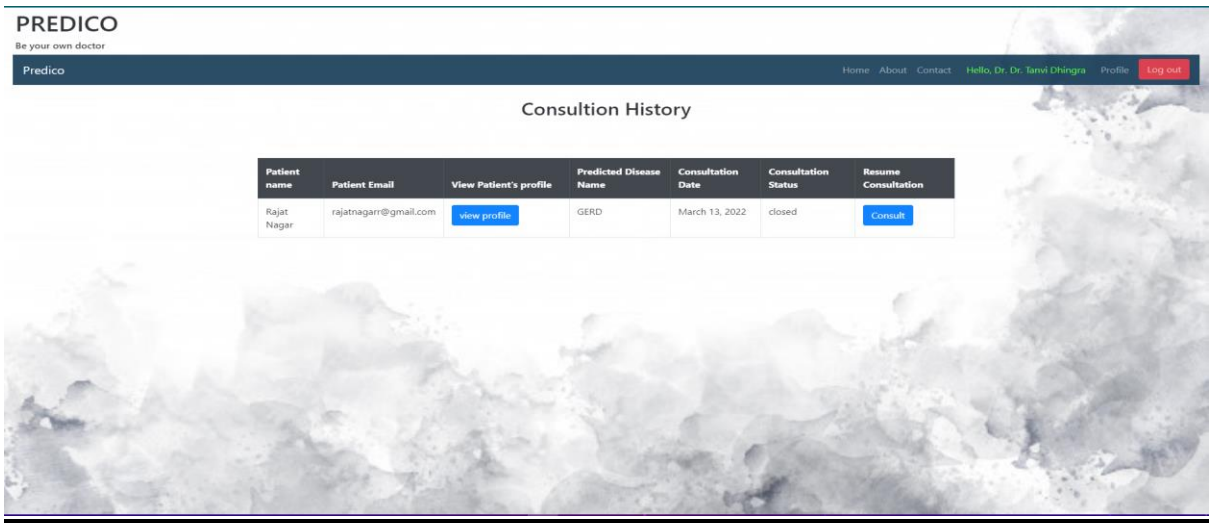

Fig 5.42 Consultation design page

```
{% load static %}
{% for obj in chat %}
{% if obj.sender == request.user %}
<div class="outgoing_msgs">
  <div class="sent_msg">
    <span class="time_date"> {{ obj.created }} </span>
    <li class="text-right list-group-item">{{ obj.message }} </li>
  </div>
</div>

{% else %}
<div class="incoming_msg">
  <div class="incoming_msg_img"> <img src="https://ptetutorials.com/images/user-profile.png" alt="Sender">
  </div>
  <div class="received_msg">
    <div class="received_withd_msg">
      <span class="time_date"> {{ obj.created }} </span>
```

```
      <li class="text-left list-group-item">{{ obj.message }}</li>

      <span class="time_date"> {{obj.sender}}</span>


    </div>
  </div>
</div>

{% endif %}
{% empty %}
<br><br>
<li class="text-right list-group-item">No messages yet!</li>
<br><br>

{% endfor %}
```

Fig 5.43 Consultation interface file

5.2.3.3 doctor

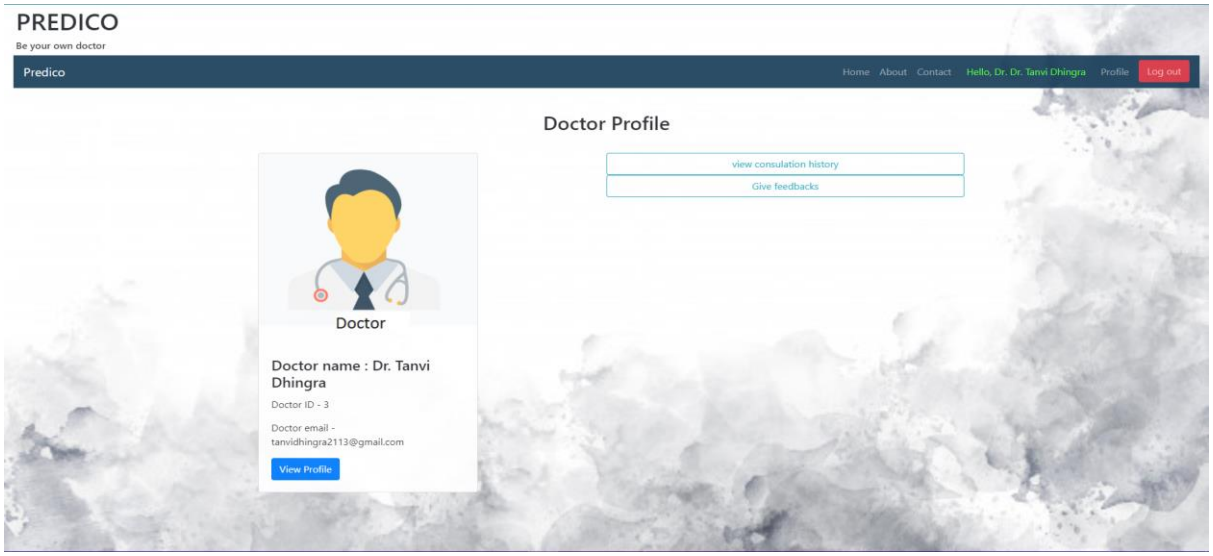

Fig 5.44 doctor design page

```
{% extends "basic.html" %}
{% load static %}
{% block head %}
{% endblock %}
{% block body %}

<br>
<div class="container mt-3 mb-3">
  <center>
    <h2>Doctor Profile</h2>
  </center><br>

  <div class="row">
    <div class="col">
      <div class="card" style="width:350px">
        <img class="card-img-top" src="{% static 'homepage/doctor.png' %}" alt="Card image" style="width:100%;">
        <div class="card-body">
          <h4 class="card-title">Doctor name : {{user.doctor.name}}</h4>
          <p class="card-text">Doctor ID - {{user.doctor.user_id}}</p>
          <p class="card-text">Doctor email - {{user.email}}</p>
          <a href="{% url 'dviewprofile' user.username %}" class="btn btn-primary">View Profile</a>
        </div>
      </div>
    </div>

    <div class="col">

      <div class="row">
        <a class="btn btn-outline-info btn-block" href="{% url 'dconsultation_history' %}">view consulation
          history</a><br>
      </div>


      <div class="row">
        <button class="btn btn-outline-info btn-block" data-toggle="modal" data-target="#myModal-feedback">Give
          feedbacks </button><br>
      </div>
```

```
    <!-- The Modal -->
    <div class="modal fade" id="myModal-feedback">
      <div class="modal-dialog modal-xl ">
        <div class="modal-content">

          <!-- Modal Header -->
          <div class="modal-header">
            <h4 class="modal-title">Feedbacks</h4>
            <button type="button" class="close" data-dismiss="modal">&times;</button>
          </div>
```

```
          <!-- Modal body -->
          <div class="modal-body">
            <form action="post_feedback" method="POST"> {% csrf_token %}
              <div class="form-group">
                <label for="comment">Give feeback:</label>
                <textarea class="form-control" rows="5" id="feedback" name="feedback"></textarea>
              </div>

            </div>

            <!-- Modal footer -->
            <div class="modal-footer">
              <button id="submit" type="submit" class="btn btn-success" data-dismiss="modal"
                style="color: white;">Submit</button>
            </div>
            </form>

          </div>
        </div>
      </div>



    </div>
  </div>
```

```
  $(document).ready(function () {
    $('#submit').click(function (event) {
      $.ajax({
        url: "{% url 'post_feedback' %}",
        type: "POST",
        data: {
          feedback: $('#feedback').val(),
          csrfmiddlewaretoken: $('input[name=csrfmiddlewaretoken]').val()
        },
        success: function (data) {
          alert(data);
        }
      });
    });
  });
</script>
{% endblock %}
```

Fig 5.45 doctor interface file

5.2.3.4 homepage

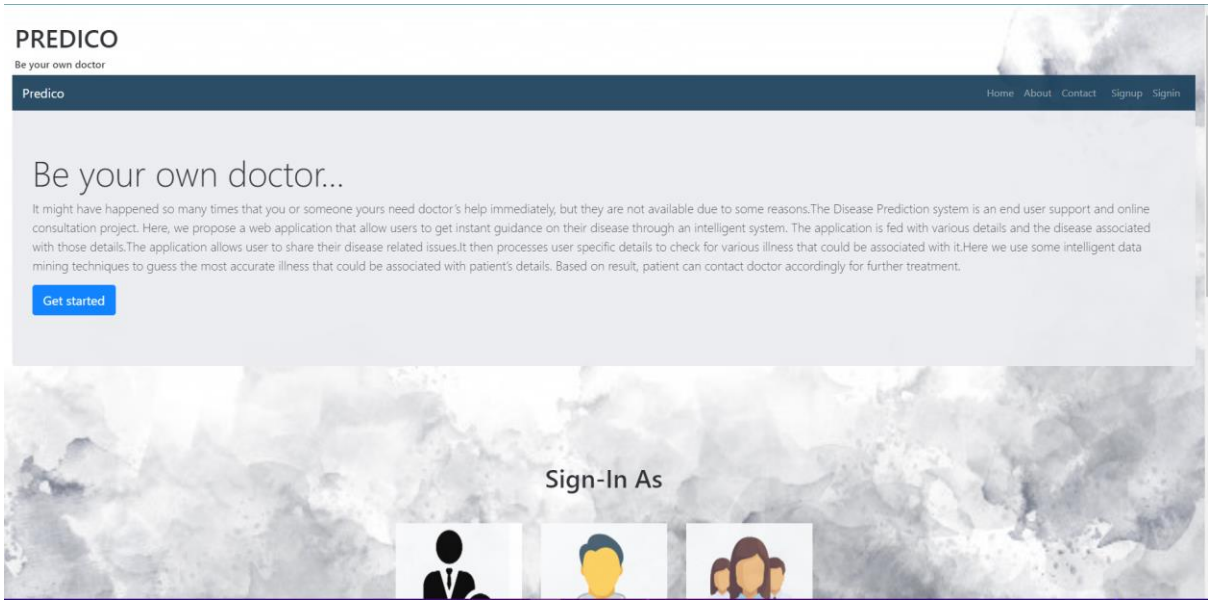

Fig 5.46 homepage design page

```
{% extends "basic.html" %}
{% load static %}

{% block head %}

<!--===============================================================================================-->
<link rel="icon" type="image/png" href="{% static 'signin_page/images/icons/favicon.ico' %}" />
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/bootstrap/css/bootstrap.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css"
    href="{% static 'signin_page/fonts/font-awesome-4.7.0/css/font-awesome.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css"
    href="{% static 'signin_page/fonts/iconic/css/material-design-iconic-font.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/animate/animate.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/css-hamburgers/hamburgers.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/animsition/css/animsition.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/select2/select2.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/daterangepicker/daterangepicker.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/css/util.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/css/main.css' %}">
<!--===============================================================================================-->

{% endblock %}

{% block body %}
```

```
<div class="limiter">
    <div class="container-login100">
        <div class="wrap-login100 p-l-55 p-r-55 p-t-65 p-b-54">
            <form class="login100-form validate-form" action="sign_in_doctor" method="POST">

                {% csrf_token %}

                <span class="login100-form-title p-b-49">
                    Login as Doctor
                </span>

                <div class="wrap-input100 validate-input m-b-23" data-validate="Username is reauired">
                    <span class="label-input100">Username</span>
                    <input class="input100" type="text" name="username" placeholder="Type your username" required>
                    <span class="focus-input100" data-symbol="&#xf206;"></span>
                </div>

                <div class="wrap-input100 validate-input" data-validate="Password is required">
                    <span class="label-input100">Password</span>
                    <input class="input100" type="password" name="password" placeholder="Type your password" required>
                    <span class="focus-input100" data-symbol="&#xf190;"></span>
                </div>

                <div class="text-right p-t-8 p-b-31">
                    <a href="#">
                        Forgot password?
                    </a>
                </div>

                <center>
                    <div>
                        {% for message in messages %}
                        <h3 style="color: red;">{{message}}</h3>
                        {% endfor %}
                    </div>
                </center>
                <br>
```

```
                <div class="container-login100-form-btn">
                    <div class="wrap-login100-form-btn">
                        <div class="login100-form-bgbtn"></div>
                        <button class="login100-form-btn" type="submit">
                            Login
                        </button>
                    </div>
                </div>


                <div class="flex-col-c p-t-155">
                    <span class="txt1 p-b-17">
                        Don't Have An Account
                    </span>

                    <a href="signup_doctor" class="txt2">
                        Sign Up
                    </a>
                </div>

            </form>
        </div>
    </div>
</div>
```

```
<div id="dropDownSelect1"></div>

<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/jquery/jquery-3.2.1.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/animsition/js/animsition.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/bootstrap/js/popper.js' %}"></script>
<script src="{% static 'signin_page/vendor/bootstrap/js/bootstrap.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/select2/select2.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/daterangepicker/moment.min.js' %}"></script>
<script src="{% static 'signin_page/vendor/daterangepicker/daterangepicker.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/countdowntime/countdowntime.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/js/main.js' %}"></script>


{% endblock %}
```

Fig 5.47 homepage interface file

5.2.3.5 patient

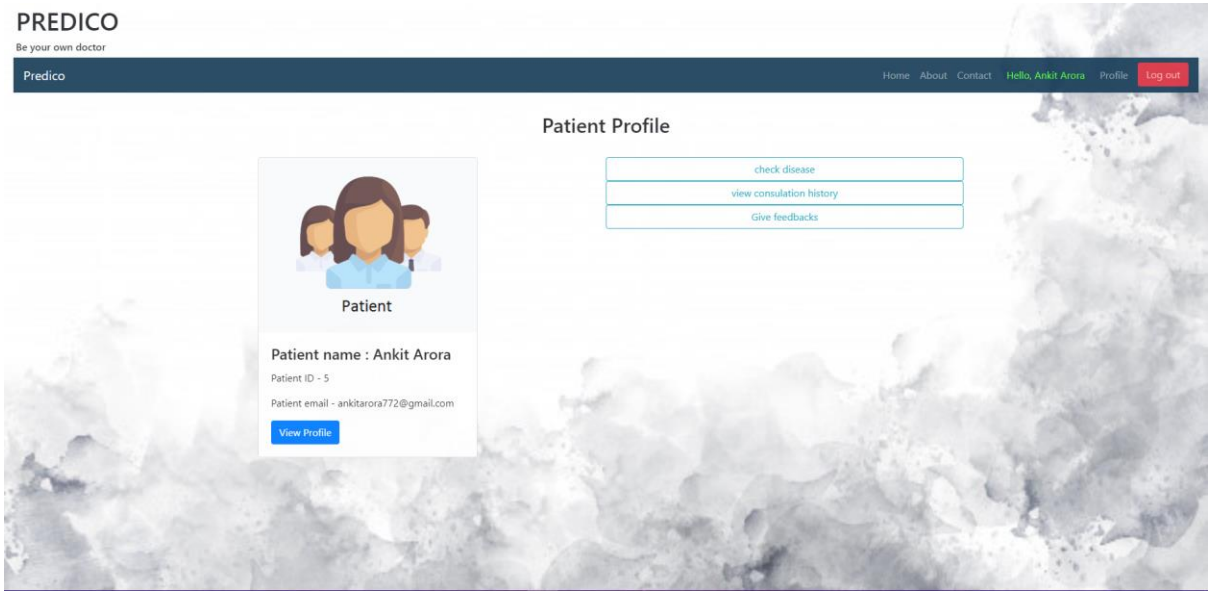

Fig 5.48 patient design page

```
{% extends "basic.html" %}
{% load static %}
{% block head %}
{% endblock %}
{% block body %}

<br>
<div class="container mt-2 mb-3">
  <center>
    <h2>Patient Profile</h2>
  </center><br>




  <div class="row">
    <div class="col">

      <div class="card" style="width:350px">
        <img class="card-img-top" src="{% static 'homepage/patient.png' %}" alt="Card image" style="width:100%;">
        <div class="card-body">
          <h4 class="card-title">Patient name : {{puser.patient.name}}</h4>
          <p class="card-text">Patient ID - {{puser.patient.user_id}}</p>
          <p class="card-text">Patient email - {{puser.email}}</p>
          <a href="{% url 'pviewprofile' puser.username %}" class="btn btn-primary">View Profile</a>
        </div>
      </div>

    </div>
```

```
    <div class="col">
      <div class="row">

        <a id="links" class="btn btn-outline-info btn-block" href="{% url 'checkdisease' %}">check disease </a><br>
      </div>

      <div class="row">
        <a class="btn btn-outline-info btn-block" href="{% url 'pconsultation_history' %}">view consulation
          history</a><br>
      </div>

      <div class="row">
        <button class="btn btn-outline-info btn-block" data-toggle="modal" data-target="#myModal-feedback">Give
          feedbacks </button><br>
      </div>
```

```
<!-- The Modal -->
<div class="modal fade" id="myModal-feedback">
  <div class="modal-dialog modal-xl ">
    <div class="modal-content">

      <!-- Modal Header -->
      <div class="modal-header">
        <h4 class="modal-title">Feedbacks</h4>
        <button type="button" class="close" data-dismiss="modal">&times;</button>
      </div>

      <!-- Modal body -->
      <div class="modal-body">
        <form action="post_feedback" method="POST"> {% csrf_token %}
          <div class="form-group">
            <label for="comment">Give feeback:</label>
            <textarea class="form-control" rows="5" id="feedback" name="feedback"></textarea>
          </div>

      </div>

      <!-- Modal footer -->
      <div class="modal-footer">
        <button id="submit" type="submit" class="btn btn-success" data-dismiss="modal"
          style="color: white;">Submit</button>
      </div>
      </form>

    </div>
  </div>
</div>
```

```
<script>
  $(document).ready(function () {
    $('#submit').click(function (event) {
      $.ajax({
        url: "{% url 'post_feedback' %}",
        type: "POST",
        data: {
          feedback: $('#feedback').val(),
          csrfmiddlewaretoken: $('input[name=csrfmiddlewaretoken]').val()
        },
        success: function (data) {
          alert(data);
        }
      });
    });
  });

</script>
{% endblock %}
```

Fig 5.49 Patient interface file

5.2.3.6 signin_page

Fig 5.50 signin_page design page

```
{% extends "basic.html" %}
{% load static %}
{% block head %}

<!--===============================================================================================-->
<link rel="icon" type="image/png" href="{% static 'signin_page/images/icons/favicon.ico' %}" />
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/bootstrap/css/bootstrap.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css"
    href="{% static 'signin_page/fonts/font-awesome-4.7.0/css/font-awesome.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css"
    href="{% static 'signin_page/fonts/iconic/css/material-design-iconic-font.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/animate/animate.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/css-hamburgers/hamburgers.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/animsition/css/animsition.min.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/select2/select2.min.css' %}">
<!--===============================================================================================-->
```

```
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/vendor/daterangepicker/daterangepicker.css' %}">
<!--===============================================================================================-->
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/css/util.css' %}">
<link rel="stylesheet" type="text/css" href="{% static 'signin_page/css/main.css' %}">
<!--===============================================================================================-->

{% endblock %}
{% block body %}



<div class="limiter">
    <div class="container-login100">
        <div class="wrap-login100 p-l-55 p-r-55 p-t-65 p-b-54">
            <form class="login100-form validate-form" action="sign_in_patient" method="POST">

                {% csrf_token %}

                <span class="login100-form-title p-b-49">
                    Login
                </span>

                <div class="wrap-input100 validate-input m-b-23" data-validate="Username is reauired">
                    <span class="label-input100">Username</span>
                    <input class="input100" type="text" name="username" placeholder="Type your username">
                    <span class="focus-input100" data-symbol="&#xf206;"></span>
                </div>

                <div class="wrap-input100 validate-input" data-validate="Password is required">
                    <span class="label-input100">Password</span>
                    <input class="input100" type="password" name="password" placeholder="Type your password">
                    <span class="focus-input100" data-symbol="&#xf190;"></span>
                </div>

                <div class="text-right p-t-8 p-b-31">
                    <a href="#">
                        Forgot password?
                    </a>
                </div>
```

```
            <center>
                <div>
                    {% for message in messages %}
                    <h3 style="color: red;">{{message}}</h3>
                    {% endfor %}
                </div>
            </center>
            <br>


            <div class="container-login100-form-btn">
                <div class="wrap-login100-form-btn">
                    <div class="login100-form-bgbtn"></div>
                    <button class="login100-form-btn" type="submit">
                        Login
                    </button>
                </div>
            </div>

            <div class="txt1 text-center p-t-54 p-b-20">
                <span>
                    Or Sign In Using
                </span>
            </div>

            <div class="flex-c-m">
                <a href="#" class="login100-social-item bg1">
                    <i class="fa fa-facebook"></i>
                </a>

                <a href="#" class="login100-social-item bg2">
                    <i class="fa fa-twitter"></i>
                </a>

                <a href="#" class="login100-social-item bg3">
                    <i class="fa fa-google"></i>
                </a>
            </div>
```

```
              <div class="flex-col-c p-t-155">
                  <span class="txt1 p-b-17">
                      Don't Have An Account
                  </span>

                  <a href="signup_patient" class="txt2">
                      Sign Up
                  </a>
              </div>

          </form>
      </div>
  </div>
</div>


<div id="dropDownSelect1"></div>

<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/jquery/jquery-3.2.1.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/animsition/js/animsition.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/bootstrap/js/popper.js' %}"></script>
<script src="{% static 'signin_page/vendor/bootstrap/js/bootstrap.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/select2/select2.min.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/daterangepicker/moment.min.js' %}"></script>
<script src="{% static 'signin_page/vendor/daterangepicker/daterangepicker.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/vendor/countdowntime/countdowntime.js' %}"></script>
<!--===============================================================================================-->
<script src="{% static 'signin_page/js/main.js' %}"></script>


{% endblock %}
```

Fig 5.51 signin_page interface file

5.2.4 DJANGO-ADMIN AND MANAGE.PY

django-admin is Django's command-line utility for administrative tasks. This document outlines all it can do.

In addition, manage.py is automatically created in each Django project. It does the same thing as django-admin but also sets the DJANGO_SETTINGS_MODULE environment variable so that it points to your project's settings.py file.

```
"""Django's command-line utility for administrative tasks."""
import os
import sys


def main():
    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'disease_prediction.settings')
    try:
        from django.core.management import execute_from_command_line
    except ImportError as exc:
        raise ImportError(
            "Couldn't import Django. Are you sure it's installed and "
            "available on your PYTHONPATH environment variable? Did you "
            "forget to activate a virtual environment?"
        ) from exc
    execute_from_command_line(sys.argv)


if __name__ == '__main__':
    main()
```

Fig 5.52 signin_page interface file

5.2.5 PREDICTION RESULTS

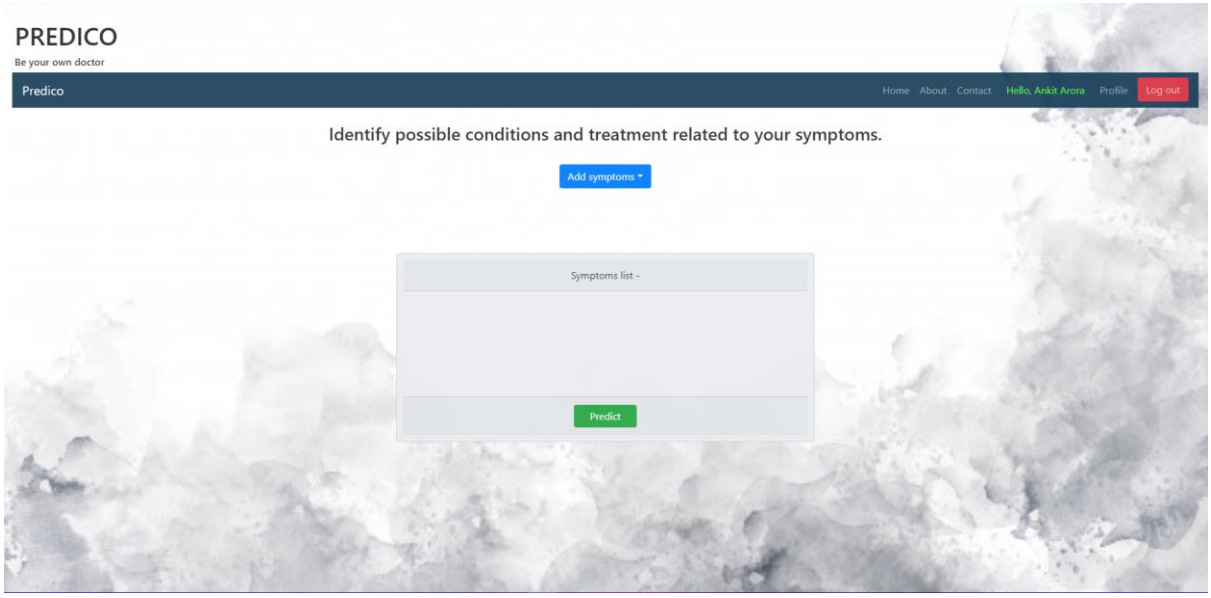

Fig 5.53 Adding symptoms design page

```
{% extends "basic.html" %}
{% load static %}


{% block head %}

<link rel="stylesheet" type="text/css" href="{% static 'patient/checkdisease/dps.css' %}">



<script>

  /* When the user clicks on the button,
toggle between hiding and showing the dropdown content */
  function Functionshow() {

    document.getElementById("searchbar").value = '';
    document.getElementById("myDropdown").classList.toggle("show");
    search_symptoms();

  }

  // Close the dropdown menu if the user clicks outside of it
  window.onclick = function (event) {
    if (!event.target.matches('.btn')) {
      if (!event.target.matches('.searchbardiv')) {
        if (!event.target.matches('.searchbar')) {

          var dropdowns = document.getElementsByClassName("drop-content");
          var i;
          for (i = 0; i < dropdowns.length; i++) {

            var openDropdown = dropdowns[i];
            if (openDropdown.classList.contains('show')) {
              openDropdown.classList.remove('show');
            }
          }
        }
      }
    }
  }
```

Fig 5.54 check disease interface file

Fig 5.55 Predicted Disease

Fig 5.56 Doctor consultation

5.3 SOFTWARE INSTALLATION

5.3.1 TECHNOLOGIES TO BE INCORPORATED AND WORKED UPON:

1. Machine Learning (Predicting kind of disease patient is suffering from)
2. Web Development (Frontend and Backend)

5.3.2 LANGUAGES AND FRAMEWORKS USED:

For Machine Learning :

- Python
- Scikit Learn

For Frontend and Backend Web Interface:

- Django - Python Open-source Web Framework
- HTML
- CSS
- BOOTSTRAP - Open-Source CSS Framework
- jQuery - JavaScript Library
- Jinja
- Werkzeug

For Database storage:

- PostgreSQL

CONCLUSION AND FUTURE SCOPE

So, Finally I conclude by saying that this project Disease prediction using machine learning is very much useful in everyone's day to day life and it is mainly more important for the healthcare sector, because they are the one that daily uses these systems to predict the diseases of the patients based on their general information and there symptoms that they are been through. Now a day's health industry plays major role in curing the diseases of the patients so this is also some kind of help for the health industry to tell the user and also it is useful for the user in case he/she doesn't want to go to the hospital or any other clinics, so just by entering the symptoms and all other useful information the user can get to know the disease he/she is suffering from and the health industry can also get benefit from this system by just asking the symptoms from the user and entering in the system and in just few seconds they can tell the exact and up to some extent the accurate diseases. If the health industry adopts this project then the work of the doctors can be reduced, and they can easily predict the disease of the patient. The Disease prediction is to provide predictions for the various and generally occurring diseases that when unchecked and sometimes ignored can turn into fatal disease and cause a lot of problems to the patient and as well as their family members.

In the near-future following enhancements could be done in this application to make it more productive for user at times of need:

- Facility for modifying user detail.
- More interactive user interface.
- Facilities for Backup creation.
- Can be done as a Web page.
- Can be done as a Mobile Application.
- More Details and Latest Diseases.

APPENDICES

- **HIV:** Human immunodeficiency virus
- **BMI:** Body Mass Index
- **KNN:** K-nearest Neighbor
- **RF:** Random Forest
- **GERD -** Gastroesophageal reflux disease
- **URL :** Uniform Resource Locator
- **CSV :** Comma Separated Values
- **DOM :** Document Object Model
- **DOS :** Disk Operating System
- **HTML :** Hyper Text Markup Language
- **CSS :** Cascading Style Sheets
- **HTTPS :** Hypertext Transfer Protocol Secure
- **OS:** Operating System
- **WSGI:** Web Server Gateway Interface

REFERENCES

[1] Sayantan Saha, Argha Roy Chowdhuri et,al “Web Based Disease Detection System”,IJERT, ISSN:22780181,Vol.2 Issue 4, April-2013

[2] M. Chen, Y. Hao, K. Hwang, L. Wang, and L.Wang,“Disease prediction by machine learning over big data from healthcare communities”, ,” IEEE Access, vol. 5, no. 1, pp. 8869–8879, 2017.

[3] Palli Suryachandra, Prof.Venkata Subba Reddy,“Comparison of Machine Learning algorithms For Breast Cancer”, IEEE.

[4] Mr Chintan Shah,Dr. Anjali Jivani, “Comparison Of Data Mining Classification Algorithms for Breast Cancer Prediction”, IEEE-31661

[5] Balasubramanian, Satyabhama, and Balaji Subramanian. "Symptom based disease prediction in medical system by using K-means algorithm." International Journal of Advances in Computer Science and Technology 3.

[6] Pingale, Kedar, et al. "Disease Prediction using Machine Learning." (2019).Mr. Chala Beyene, Prof. Pooja Kamat, “Survey on Prediction and Analysis the Occurrence of Heart Disease Using Data Mining Techniques”, International Journal of Pure and Applied Mathematics, 2018.

[7] Hariharan, M., Polat, K., & Sindhu, R. (2014). A new hybrid intelligent system for accurate detection of Parkinson's disease. Computer Methods and Programs in Biomedicine, 113(3), 904–913.

[8] Hashem, S., Esmat, G., Elakel, W., Habashy, S., Raouf, S. A., ElHefnawi, M., Eladawy, M., & ElHefnawi, M. (2018). Comparison of Machine Learning Approaches for Prediction of Advanced Liver Fibrosis in Chronic Hepatitis C Patients. IEEE/ACM Transactions on Computational Biology and Bioinformatics, 15(3), 861–868.

[9] Iswanto, I., Laxmi Lydia, E., Shankar, K., Nguyen, P. T., Hashim, W., & Maseleno, A. (2019). Identifying diseases and diagnosis using machine learning. International Journal of Engineering and Advanced Technology, 8(6 Special Issue 2), 978–981.

[10] Javeed, A., Zhou, S., Yongjian, L., Qasim, I., Noor, A., & Nour, R. (2019). An Intelligent Learning System Based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection. IEEE Access, 7, 180235–180243.

[11] Javeed, A., Zhou, S., Yongjian, L., Qasim, I., Noor, A., & Nour, R. (2019). An Intelligent Learning System Based on Random Search Algorithm and Optimized Random Forest Model for Improved Heart Disease Detection. IEEE Access, 7, 180235–180243.

[12] Chinmayi Chitnis and Roger Lee. Improving Health-Care Systems by Disease Prediction. 2018 International Conference on Computational Science and Computational Intelligence (CSCI)

[13] A.Davis, D., V.Chawla, N., Blumm, N., Christakis, N., & Barbasi, A. L. (2008). Predicting Individual Disease Risk Based On Medical History. Adam, S., & Parveen, A. (2012). Prediction System For Heart Disease Using Naive Bayes.

[14] P. Groves, B. Kayyali, D. Knott, and S. V. Kuiken, “The ’big data’ revolution in healthcare: Accelerating value and innovation,” 2016.

[15] Data mining, Margaret Rouse, Search SQL Server last accessed 12.09.2018, the article can be found here. https://searchsqlserver.techtarget.com/definition/data- mining

[16] https://www.kaggle.com/neelima98/disease-prediction-using-machine-learning

[17] Supervised and Unsupervised Machine Learning Algorithms, Jason Brownlee, 16.03.2016, published in Machine Learning Algorithms, last accessed 12.09.2018.

[18] Decision trees, scikit-learn.org last accessed 12.09.2018.

[19] RandomForestClassifier, scikit-learn.org last accessed 12.09.2018.

[20] GradientBoostingClassifier, scikit-learn.org last accessed 12.09.2018.