MULTIPLE DISEASE PREDICTION SYSTEM

A MAJOR PROJECT REPORT

Submitted by

Gautam Dulgas	**03414902020**
Prashant	**06514902020**
Shikhar Suryan	**08214902020 (TL)**
Vaibhav Kapila	**09814902020**

in partial fulfillment for the award of the degree

of

BACHELOR OF COMPUTER APPLICATION

MAHARAJA SURAJMAL INSTITUTE

C-4, JANAK PURI,

NEW DELHI - 110058

JUN 2023

ACKNOWLEDGEMENT

We would like to express our profound gratitude to

Prof. (Dr.) Harish Singh (Director (O) of Maharaja Surajmal Institute) for his contributions to the completion of our project titled

"**Multiple Disease Prediction System**".

We would like to express our special thanks to our mentor

Mr. Sundeep Kumar (Assistant Professor, Computer Science Shift - 1) for his time and efforts he provided throughout the year. Your useful advice and suggestions were really helpful to us during the project's completion. In this aspect, we are eternally grateful to you.

We would like to acknowledge that this project was completed entirely by our team and not by someone else.

Gautam Dulgas

Prashant

Shikhar Suryan (Team Leader)

Vaibhav Kapila

MAHARAJA SURAJMAL INSTITUTE

BONA FIDE CERTIFICATE

Certified that this project report **"Multiple Disease Prediction System"** is the bona fide work of "**Gautam Dulgas, Prashant, Shikhar Suryan and Vaibhav Kapila"** who carried out the project work under my supervision.

SIGNATURE

Mr. Sundeep Kumar

(SUPERVISOR)

Assistant Professor, Computer Science (1st Shift),

Maharaja Surajmal Institute, Department of Computer Science,

C-4 Janakpuri, New Delhi, Delhi 110058

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 DISEASE PREDICTION

Multiple Disease Prediction using Machine Learning is a system which predicts the disease based on the information provided by the user. It also predicts the disease of the patient or the user 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. This DPUML is previously done by many other organizations but our intention is to make it different and beneficial for the users who are using this system. This Multiple Disease Prediction Using Machine Learning is completely done with the help of Machine Learning and Python Programming language with FLASK Interface for it and also using the dataset that is available previously by the hospitals using that we will predict the disease. Now a day's doctors are adopting many scientific technologies and methodology for both identification and diagnosing not only common disease, but also many fatal diseases. The successful treatment is always attributed by right and accurate diagnosis. Doctors may sometimes fail to take accurate decisions while diagnosing the disease of a patient, therefore Multiple Disease Prediction systems which use machine learning algorithms assist in such cases to get accurate results. The project Multiple Disease Prediction using machine learning is developed to overcome general disease in earlier stages as we all know in competitive environment of economic development the mankind has involved so much that he/she is not concerned about health according to research there are 40% peoples how ignores about general disease which leads to harmful disease later. The

main reason of ignorance is laziness to consult a doctor and time concern the peoples have involved themselves so much that they have no time to take an appointment and consult the doctor which later results into fatal disease. According to research there are 70% peoples in India suffers from general disease and 25% of peoples face death due to early ignorance the main motive to develop this project is that a user can sit at their convenient place and have a check-up of their health the UI is designed in such a simple way that everyone can easily operate on it and can have a check-up.

1.2 PROBLEM DEFINITION

Nowadays 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 in information yet and knowledge poor and this industry is very vast industry which has 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 the need. So the problem here is that many people goes to hospitals or clinic to know how is their health 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 persons 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 Multiple Disease Prediction will help a lot to the various people's ranging from children to teenagers to adults and also the senior citizens.

1.3 PROJECT PURPOSE

The purpose of making this project called "Multiple Disease Prediction Using Machine Learning" 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 predicts the disease of the patient he/she is been through. If this Prediction is done at the early stages of the disease with the help of this project and all other necessary measure the disease can be cured and in general this prediction system can also be very useful in health industry. If 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 Multiple Disease Prediction is to provide prediction for the various and generally occurring diseases that when unchecked and sometimes ignored can turns into fatal disease and cause lot of problem 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 in information yet and knowledge poor and this industry is very vast industry which has 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 peoples who are in the need.

1.4 PROJECT FEATURES

The features of Multiple Disease Prediction Using Machine Learning are as follows.

- This Project will predict the diseases of the patients based on the symptoms and other general information using the datasets.
- This is done based on the previous datasets of the hospitals so after comparing it can provide up to 80% of accurate results, and the project is still developing further to get the 100% accurate results.
- With the help of Disease prediction, it can predict the disease of the patient and can solve various problems and prevents from various aspects.
- It provides security for the system so that no one can break into that and no one can make any changes in the system.

- The disease is predicted using the algorithms and the user has to enter the symptoms from the given drop-down menu, in order to get correct accuracy, the user has to enter all the symptoms.
- Here we can easily prepare the data and transform that data into algorithm, which will reduce the overall work of the project.
- To make user more application friendly rather than discussing with others for their disease.
- It provides the necessary options to choose from the types and attributes.
- Here the user has to register first, in order to use the prediction and then login to the system using the credentials such as username and password.
- Once user open the system to login user needs to register by clicking on register/signup button.
- After which user needs to provide some basic details of signup and then the details of user are saved in system

CHAPTER 2

LITERATURE SURVEY

2.1 MACHINE LEARNING

Tom Mitchell states machine learning as “A computer program is said to learn from experience and from some tasks and some performance on, as measured by, improves with experience”. Machine Learning is combination of correlations and relationships, most machine learning algorithms in existence are concerned with finding and/or exploiting relationship between datasets. Once Machine Learning Algorithms can pinpoint on certain correlations, the model can either use these relationships to predict future observations or generalize the data to reveal interesting patterns. In Machine Learning there are various types of algorithms such as Regression, Linear Regression, Logistic Regression, Naive Bayes Classifier, Bayes theorem, KNN (K-Nearest Neighbor Classifier), Decision Tress, Entropy, ID3, SVM (Support Vector Machines), K-means Algorithm, Random Forest and etc.,

The name machine learning was coined in 1959 by Arthur Samuel. Machine learning explores the study and construction of algorithms that can learn from and make predictions on data Machine learning is closely related to (and often overlaps with) computational statistics, which also focuses on prediction-making through the use of computers. It has strong ties to mathematical optimization, which delivers methods, theory and application domains to the field. Machine learning is sometimes conflated with data mining, where the latter subfield focuses more on exploratory data analysis and is known as unsupervised learning.

Within the field of data analytics, machine learning is a method used to devise complex models and algorithms that lend themselves to prediction; in commercial use, this is known as predictive analytics. These analytical models allow researchers, data scientists, engineers, and analysts to "produce reliable, repeatable decisions and results" and uncover "hidden insights" through learning from historical relationships and trends in the data.

Machine learning tasks Machine learning tasks are typically classified into several broad categories:

Supervised learning: The computer is presented with example inputs and their desired outputs, given by a "teacher", and the goal is to learn a general rule that maps inputs to outputs. As special cases, the input signal can be only partially available, or restricted to special feedback.

Semi-supervised learning: The computer is given only an incomplete training signal: a training set with some (often many) of the target outputs missing.

Active learning: The computer can only obtain training labels for a limited set of instances (based on a budget), and also has to optimize its choice of objects to acquire labels for. When used interactively, these can be presented to the user for labelling.

Unsupervised learning: No labels are given to the learning algorithm, leaving it on its own to find structure in its input. Unsupervised learning can be a goal in itself (discovering hidden patterns in data) or a means towards an end (feature learning).

Reinforcement learning: Data (in form of rewards and punishments) are given only as feedback to the program's actions in a dynamic environment, such as driving a vehicle or playing a game against an opponent.

2.1.1 FEATURES OF MACHINE LEARNING

- It is nothing but automating the Automation.
- Getting computers to program themselves.
- Writing Software is bottleneck.
- Machine leaning models involves machines learning from data without the help of humans or any kind of human intervention.
- Machine Learning is the science of making of making the computers learn and act like humans by feeding data and information without being explicitly programmed.

- Machine Learning is totally different from traditionally programming, here data and output is given to the computer and in return it gives us the program which provides solution to the various problems. Below is the figure.

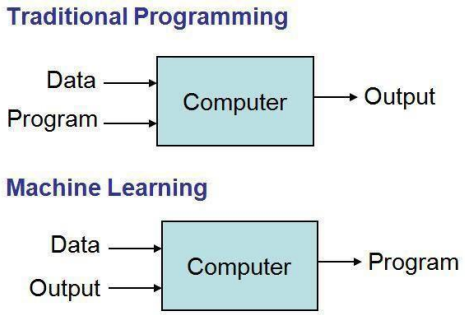


Fig 2.1.1 Traditional Programming vs Machine Learning

- Machine Learning is a combination of Algorithms, Datasets, and Programs.
- There are Many Algorithms in Machine Learning through which we will provide us the exact solution in predicting the disease of the patients.
- How Does Machine Learning Works?
- Solution to the above question is Machine learning works by taking in data, finding relationships within that data and then giving the output.

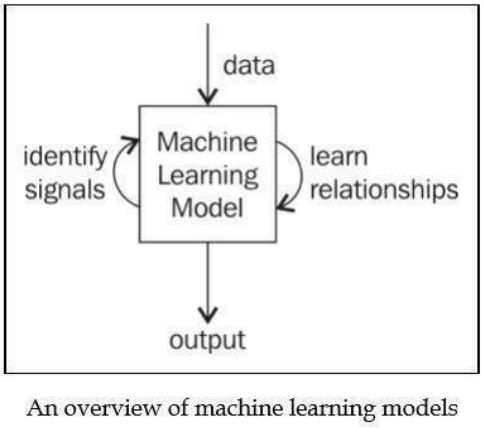


Fig 2.1.2 Machine Learning Model

- There are various applications in which machine learning is implemented such as Web search, computing biology, finance, e-commerce, space exploration, robotics, social networks, debugging and much more.
- There are 3 types of machine learning supervised, unsupervised, and reinforcement.

2.2 EXISTING SYSTEM

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 group. But these models are only valuable in clinical situations and not in big industry sector. 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.

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 Multiple Disease Prediction system which predicts the disease of the patients from which he/she is suffering from and to do this we have used some performance measures like ROC, KAPPA Statistics, RMSE, MEA and various other tools. After using various techniques such as neural networks to make predictions of the diseases and after doing that we come to conclusion that it can predicts up to 90% accuracy rate after doing the experimentation and verifying the results. The information of patient statistics, results, disease history in recorded in EHR, which enables to identify the potential data centric solution, which reduces the cost of medical case studies. Existing system can predict the disease but not the sub type 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 PROPOSED SYSTEM

The proposed system of Multiple Disease Prediction using machine learning is that we have used many techniques and algorithms and all other various tools to build a system which predicts the disease of the patient using the symptoms and by taking those symptoms we are comparing with the system's dataset that is previously available. By taking those datasets and comparing with the patient's disease we will predict the accurate percentage disease of the patient. The dataset and symptoms go to the prediction model of the system where the data is pre-processed for the future references and then the feature selection is done by the user where he will enter the various symptoms. Then the classification of those data is done with the help of various algorithms and techniques such as Decision Tree, KNN, Naïve Bayes, Random Forest and etc. Then the data goes in the recommendation model, there it shows the risk analysis that is involved in the system and it also provides the probability estimation of the system such that it shows the various probability like how the system behaves when there are n number of predictions are done and it also does the recommendations for the patients from their final result and also from their symptoms like it can show what to use and what not to use from the given datasets and the final results. Here we have combined the overall structure and unstructured form of data for the overall risk analysis that is required for doing the prediction of the disease. Using the structured analysis, we can identify the chronic types of disease in a particular region and particular community. In unstructured analysis we select the features automatically with the help of algorithms and techniques. This system takes symptoms from the user and predicts the disease accordingly based on the symptoms that it takes and also from the previous datasets, it also helps in continuous evaluation of viral diseases, heart rate, blood pressure, sugar level and much more which is in the system and along with other external symptoms its predicts the appropriate and accurate disease.

2.4 SOFTWARE DESCRIPTION

2.4.1 PYTHON

Python is a multi-paradigm programming language. Object-oriented programming and structured programming are fully supported, and many of its features support functional programming and aspect-oriented programming (including by metaprogramming and metaobjects. Many other paradigms are supported via extensions, including design by contract and logic programming. Python uses dynamic typing and a combination of reference counting and a cycle-detecting garbage collector for memory management. It also features dynamic name resolution (late binding), which binds method and variable names during program execution.

Python's developers strive to avoid premature optimization, and reject patches to non-critical parts of CPython that would offer marginal increases in speed at the cost of clarity. When speed is important, a Python programmer can move time-critical functions to extension modules written in languages such as C, or use PyPy, a just-in-time compiler. Cython is also available, which translates a Python script into C and makes direct C-level API calls into the Python interpreter. An important goal of Python's developers is keeping it fun to use. Python's design offers some support for functional programming in the Lisp tradition. It has filter, map, and reduce functions, list comprehensions, dictionaries, sets, and generator expressions. The standard library has two modules (itertools and functools) that implement functional tools borrowed from Haskell and Standard ML.

2.4.2 BENEFITS OF PYTHON

- Presence of Third-Party Modules
- Extensive Support Libraries
- Open Source and Community Development
- Learning Ease and Support Available
- User-friendly Data Structures
- Productivity and Speed
- Highly Extensible and Easily Readable Language.

2.4.3 Flask Interface

A Flask interface in Python refers to the development of a web application using the Flask framework. Flask is a lightweight and flexible web framework that allows you to build web applications and APIs in Python. It provides the necessary tools and libraries to handle routing, request handling, template rendering, and other web-related tasks.

Here are some key aspects of a Flask interface:

Web Application Development: Flask allows you to develop web applications using Python. You can define routes to handle different URLs and HTTP methods, such as GET and POST. Flask provides decorators to associate Python functions with specific routes, making it easy to handle different requests and generate responses.

Templating: Flask supports template rendering using templating engines like Jinja2. Templating allows you to separate the presentation logic from the application logic by defining HTML templates with placeholders that can be dynamically filled with data.

Request Handling: Flask provides request and response objects to handle incoming requests and generate responses. You can access request data, such as form inputs or query parameters, and perform appropriate actions based on the request. You can also set response headers, cookies, or redirect users to different URLs.

Routing: Flask uses routing to map URLs to specific functions or views. You can define routes with specific URL patterns and associate them with corresponding functions that handle the requests. For example, you can define a route for the home page ("/") and associate it with a function that renders the appropriate template.

Flask Extensions: Flask has a vast ecosystem of extensions that provide additional functionality and integrations. These extensions can be used to handle forms,

manage databases, implement authentication, handle file uploads, and more. They extend the capabilities of Flask and make it easier to develop complex web applications.

Deployment: Flask interfaces can be deployed to various web servers or cloud platforms. Flask applications can be run using a built-in development server during development and can be deployed to production servers like Gunicorn or uWSGI. Flask can also be deployed as a containerized application using platforms like Docker or can be deployed on cloud platforms like Heroku or AWS.

Overall, a Flask interface allows you to create web applications using Python with the help of the Flask framework. It provides a flexible and lightweight way to handle routing, request handling, templating, and other web-related tasks, making it easier to develop and deploy web applications in Python.

To create a flask interface:

To create a Flask interface, you can follow these steps:

Set up a Flask Project: Start by creating a new directory for your Flask project. Open a command prompt or terminal, navigate to the directory, and create a virtual environment (optional but recommended) to isolate the project dependencies. Activate the virtual environment.

Install Flask: Install Flask using a package manager like pip. Run the following command in the command prompt or terminal:

pip install flask

Create the Flask Application: Create a new Python file (e.g., app.py) in your project directory. Open the file in a text editor or an integrated development environment (IDE).

Import Flask and Create the App: In the app.py file, import the Flask module and create a Flask application instance. Add the following code:

```
python
from flask import Flask

app = Flask(__name__)
```

Define Routes and Views: Below the app creation code, define the routes and associated view functions. A view function is a Python function that handles a specific route and generates a response. For example, to define a route for the home page ("/"), add the following code:

```
python
@app.route("/")
def home():
    return "Welcome to the Flask interface!"
```

Run the Application: At the end of the app.py file, add the following code to run the Flask application:

```
python
if __name__ == "__main__":
    app.run(debug=True)
```

Save the File: Save the app.py file.

Start the Flask Application: In the command prompt or terminal, navigate to the project directory containing the app.py file. Run the following command to start the Flask application:

```
python app.py
```

Access the Interface: Once the Flask application is running, open a web browser and visit http://localhost:5000 (or the URL specified in the command prompt or terminal). You should see the message "Welcome to the Flask interface!" displayed on the page.

Add More Routes and Views: Expand your Flask interface by adding more routes

and associated view functions. You can define routes for different pages, handle form submissions, render HTML templates, and more. Refer to the Flask documentation for more information on route decorators, request handling, template rendering, and other Flask features.

That's it! You have created a basic Flask interface. You can continue building on this foundation by adding more functionality, incorporating templates, handling form submissions, connecting to databases, and integrating other Flask extensions or libraries as needed for your specific project requirements.

2.4.4 HTML

HTML, or Hypertext Markup Language, is a standard markup language used for creating the structure and presenting content on the World Wide Web. It is the foundation of web pages and is interpreted by web browsers to render and display the content to users. HTML uses a set of tags and attributes to define the structure, formatting, and functionality of web documents.

Here's a detailed explanation of HTML components and their purpose:

Tags: HTML documents consist of elements enclosed in tags. Tags define the beginning and end of an element and are denoted by angle brackets (< >). There are two types of tags: opening tags (<tag>) and closing tags (</tag>).

Elements: Elements are the building blocks of HTML documents and are defined by tags. An element consists of an opening tag, content, and a closing tag. For example, is the opening tag for a paragraph element, and is the closing tag.

Attributes: Attributes provide additional information about an HTML element. They are specified within the opening tag and consist of a name and a value. Attributes define characteristics such as the appearance, behavior, or functionality of an element. For example, the href attribute in an anchor tag (<a>) specifies the URL of the link.

Head: The <head> element is a container for metadata and other non-visible elements. It includes elements like the document title (<title>), character encoding (<meta charset="UTF-8">), CSS stylesheets (<link rel="stylesheet" href="styles.css">), and JavaScript scripts (<script src="script.js"></script>).

Body: The <body> element contains the visible content of an HTML document. It includes elements such as headings (<h1> to <h6>), paragraphs ( ), lists (<ul>, <ol>, <li>), images (<img>), links (<a>), tables (<table>, <tr>, <td>), forms (<form>, <input>, <button>), and more.

Text Formatting: HTML provides various tags for text formatting. For example:

 or <b>: Renders text in bold.
 or <i>: Renders text in italics.
<u>: Renders text with an underline.
<s>: Renders text with a strikethrough.
Hyperlinks: Hyperlinks are created using the <a> (anchor) tag. The href attribute specifies the URL to which the link points. For example: Example. Clicking on the link will navigate the user to the specified URL.

Images: Images are displayed using the <img> tag. The src attribute specifies the source URL of the image file, and the alt attribute provides alternative text that is displayed if the image cannot be loaded. Example: <img src="image.jpg" alt="Description">.

Lists: HTML supports ordered (<ol>) and unordered (<ul>) lists. List items are marked with the <li> tag. For example:

Item 1
Item 2

Tables: Tables are created using the <table>, <tr> (table row), and <td> (table data) tags. They are used to organize data into rows and columns. Example:
<table>
<tr>
<td>Data 1</td>

2.4.5 CSS

CSS, or Cascading Style Sheets, is a style sheet language used for describing the presentation and visual formatting of HTML and XML documents. It provides a set of rules that define how elements in a document should be displayed, allowing developers to control the layout, colors, fonts, and other aspects of the web page's appearance.

Here's a detailed explanation of CSS components and their purpose:

Selectors: Selectors are used to target specific HTML elements that you want to style. CSS selectors can target elements based on their tag names, classes, IDs, attributes, and more. Selectors allow you to specify which elements the CSS rules should apply to.

Properties: CSS properties define the visual properties of the selected elements. They determine how the elements should be displayed, such as their color, size, font, margins, padding, and positioning. CSS properties can be set to specific values, including keywords, numeric values, colors, or other CSS units.

Values: CSS values are assigned to properties and define the specific settings for each property. For example, the color property can have values like "red", "#FF0000", or "rgb(255, 0, 0)" to set the text color to red.

Declarations: Declarations are made up of a CSS property and its corresponding value. They are enclosed within curly braces ({ }) and are written as key-value pairs. Multiple declarations can be grouped together to define the styles for a specific selector.

Rules: CSS rules consist of selectors and their corresponding declarations. A rule targets specific elements in an HTML document and defines the styles to be applied to them. For example:

```
h1 {
  color: blue;
  font-size: 24px;
}
```

In this rule, the h1 selector targets all <h1> elements, and the declarations set the text color to blue and the font size to 24 pixels.

Stylesheets: CSS rules are typically written in external CSS files, which are referred to as stylesheets. Stylesheets are linked to HTML documents using the <link> tag within the <head> section. Multiple stylesheets can be included in a single HTML document, allowing for modularity and reusability of styles across multiple pages.

Inline CSS: CSS rules can also be applied directly within HTML elements using the style attribute. This is called inline CSS. Inline styles take precedence over external stylesheets, allowing for specific styles to be applied to individual elements. For example:

<h1 style="color: red;">Hello</h1>

In this case, the inline style sets the text color of the <h1> element to red.

CSS Selectors: CSS offers a wide range of selectors to target elements in different ways. Some commonly used selectors include:

Tag selectors (e.g., h1, p, div): Selects elements based on their tag names.

Class selectors (e.g., .classname): Selects elements with a specific class assigned to them.

ID selectors (e.g., #idname): Selects an element with a specific ID assigned to it.

Attribute selectors (e.g., [attribute=value]): Selects elements based on specific attribute values.

Pseudo-classes (e.g., :hover, :first-child): Selects elements based on their state or position.

Inheritance and Specificity: CSS rules can inherit styles from parent elements, allowing for consistent styles across multiple elements. Specificity determines which CSS rule takes precedence when multiple rules target the same element. It is determined by the type of selector used

2.4.6 Javascript

JavaScript is a high-level, interpreted programming language primarily used for adding interactivity and dynamic behavior to web pages. It is a fundamental technology of the web and is supported by all major web browsers. JavaScript allows developers to create interactive elements, manipulate and modify web page content, handle events, and communicate with servers to build dynamic web applications.

Here's a detailed explanation of JavaScript and its key aspects:

Scripting Language: JavaScript is often referred to as a scripting language because it is typically used in the context of web pages to enhance their functionality. It is executed by the browser on the client-side, meaning it runs on the user's device rather than on the web server.

Object-Oriented: JavaScript is an object-oriented language, which means it uses objects to represent and manipulate data. Objects in JavaScript can have properties (variables) and methods (functions) associated with them. Objects can be created from predefined classes or custom-defined using constructor functions or object literals.

Syntax and Structure: JavaScript syntax is derived from the C programming language and shares similarities with other programming languages such as Java and C++. JavaScript code is typically embedded directly within HTML documents using <script> tags or can be placed in separate external files and linked to HTML documents.

Variables and Data Types: JavaScript is dynamically typed, meaning variables can hold values of any data type, and their types can change during runtime. JavaScript supports various data types, including numbers, strings, booleans, arrays, objects, and more. Variables are declared using the var, let, or const keywords.

Functions: Functions in JavaScript are blocks of reusable code that perform a specific task. They can be defined using the function keyword or as arrow functions (=>). Functions can accept parameters, perform operations, and return values. JavaScript also supports anonymous functions and higher-order functions, which can be passed as arguments or assigned to variables.

DOM Manipulation: The Document Object Model (DOM) represents the structure of an HTML document as a hierarchical tree of objects. JavaScript provides powerful APIs to interact with the DOM, allowing developers to manipulate HTML elements, modify their content, change styles, handle events, create or remove elements dynamically, and update the web page in response to user actions.

Event Handling: JavaScript enables the handling of various events triggered by user interactions, such as clicks, key presses, form submissions, and page load. Event listeners can be attached to HTML elements, and JavaScript code can respond to these events by executing specific actions or functions.

Asynchronous Programming: JavaScript supports asynchronous programming through features like callbacks, promises, and async/await. Asynchronous operations, such as

making HTTP requests or handling file operations, can be performed without blocking the execution of other code. This allows for smoother user experiences and efficient handling of time-consuming tasks.

Libraries and Frameworks: JavaScript has a vast ecosystem of libraries and frameworks that extend its capabilities and simplify web development. Popular JavaScript frameworks include React, Angular, and Vue.js, which provide tools and abstractions for building complex web applications.

Server-Side JavaScript: In addition to client-side JavaScript, JavaScript can also be used on the server-side with technologies like Node.js. This allows JavaScript to be used for building full-stack applications, where the same programming language is used on both the client and server sides.

JavaScript's versatility and wide browser support have made it an essential language for web development, enabling the creation of interactive web pages and powerful web applications.

2.4.7 Jupyter Notebook

Jupyter Notebook is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and explanatory text. It is widely used for interactive data analysis, scientific computing, machine learning, and data visualization. Here are some key details about Jupyter Notebook:

Interactive Computing Environment: Jupyter Notebook provides an interactive computing environment where users can write and execute code in "cells." Each cell can contain code written in different programming languages, including Python, R, Julia, and others. The cells can be executed individually, allowing for experimentation and quick feedback.

Notebooks: Jupyter Notebook documents are called "notebooks" and are organized into a collection of cells. Notebooks have a file extension of ".ipynb" and can be created, opened, and saved within the Jupyter Notebook interface. Notebooks store both the code and its output, making them self-contained and reproducible.

Markdown Support: Jupyter Notebook supports Markdown, a lightweight markup language, allowing users to create formatted text, headings, lists, tables, and mathematical equations within the notebook. Markdown cells can be used to provide explanations, document code, and create rich text narratives alongside the code.

Code Execution: Cells can be executed individually by clicking the "Run" button or using keyboard shortcuts. When a code cell is executed, the code is sent to the kernel associated with the notebook, which runs the code and returns the output. The output can include text, tables, plots, interactive widgets, or any other relevant result produced by the code.

Kernel: A kernel is a computational engine that executes the code in Jupyter Notebook. Each notebook is associated with a specific kernel, allowing users to write code in different programming languages. For example, a Python kernel can execute Python code, while an R kernel can execute R code. Kernels provide an isolated environment for code execution, ensuring that variables and imports are retained across different cells.

Rich Output: Jupyter Notebook supports rich output, enabling the display of interactive plots, images, HTML, LaTeX equations, audio, and video directly within the notebook. This feature allows for easy exploration and visualization of data, making it an effective tool for data analysis and scientific computing.

Collaboration and Sharing: Jupyter Notebook facilitates collaboration by allowing users to share their notebooks with others. Notebooks can be saved and exported in various formats, including HTML, PDF, Markdown, and executable scripts. The notebooks can be shared via email, version control systems, or hosted on platforms like GitHub, making it easy for others to reproduce and extend the analyses.

Jupyter Ecosystem: Jupyter Notebook is part of a larger ecosystem that includes JupyterLab, JupyterHub, and nbconvert. JupyterLab provides a more flexible and powerful user interface for working with notebooks and other computational workflows. JupyterHub allows multiple users to access Jupyter Notebook instances on shared servers. nbconvert is a tool for converting notebooks to different formats, making it convenient for generating reports, slides, or static websites from Jupyter notebooks.

Jupyter Notebook has gained popularity due to its versatility, interactivity, and support for various programming languages. It has become an essential tool for data scientists, researchers, educators, and professionals working in fields that involve data analysis, visualization, and prototyping.

2.4.8 Visual Studio Code

Visual Studio Code (VS Code) is a free, lightweight, open-source source code editor developed by Microsoft. It has gained significant popularity among developers due to its extensive features, flexibility, and strong community support. Here's a detailed explanation of VS Code:

Code Editing: VS Code provides a powerful and customizable code editing experience. It supports syntax highlighting, code completion, code formatting, and intelligent code suggestions for various programming languages. The editor includes features like bracket matching, code folding, and multiple cursors, which enhance productivity and streamline the coding process.

Extensions and Marketplace: VS Code has a rich ecosystem of extensions contributed by the community, which extends its functionality for different programming languages, frameworks, and tools. The VS Code Marketplace allows users to browse and install extensions directly within the editor, enabling customization and integration with various development workflows.

Integrated Terminal: VS Code features an integrated terminal that allows developers to run command-line tools and scripts within the editor. The terminal supports multiple shells, such as PowerShell, Bash, and Command Prompt, and provides seamless integration with the code

editing environment. This eliminates the need to switch between different applications while working on a project.

Version Control: VS Code integrates with popular version control systems like Git, Mercurial, and SVN. It provides built-in source control features, allowing users to view changes, commit files, create branches, and perform other version control operations without leaving the editor. This simplifies collaboration and makes it easy to manage code repositories.

Debugging: VS Code offers a powerful debugging experience for various programming languages and frameworks. It provides breakpoints, call stacks, variable inspection, and step-by-step execution, allowing developers to identify and fix issues efficiently. Debug configurations can be easily set up for different environments, making it straightforward to debug code locally or remotely.

IntelliSense: IntelliSense is a code completion feature in VS Code that provides context-aware suggestions and autocompletion while writing code. It offers information about functions, methods, classes, and APIs, helping developers write code faster and with fewer errors. IntelliSense is available for a wide range of programming languages and can be enhanced with language-specific extensions.

Task Automation: VS Code allows the automation of repetitive tasks through the use of tasks and build systems. It provides a flexible task runner that can be configured to execute commands, scripts, or build processes. This enables seamless integration with build tools, testing frameworks, and task runners specific to the project requirements.

Integrated Git GUI: VS Code includes a built-in Git Graphical User Interface (GUI) that visualizes Git repositories and their history. Users can view commit details, branch information, and perform Git operations like staging changes, committing, and pushing directly from the editor. The Git GUI enhances productivity and simplifies version control workflows.

Customization and Theming: VS Code offers extensive customization options to personalize the editor's appearance and behavior. Users can choose from a wide range of themes, icon sets, and fonts to create a visually appealing environment. The editor also provides configurable settings, keybindings, and snippets, allowing users to tailor their coding experience to their preferences and workflow.

Cross-Platform Support: VS Code is available for Windows, macOS, and Linux, ensuring a consistent coding experience across different operating systems. It provides a familiar interface and behavior regardless of the platform, allowing developers to work seamlessly on different machines and collaborate with team members using different operating systems.

Visual Studio Code has become a popular choice for developers working on a wide range of projects. Its flexibility, extensive features, and vibrant community ecosystem make it a versatile and powerful code editor for various programming languages and development workflows.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FUNCTIONAL REQUIREMENTS

A Functional requirement defines a function of a system or its component. A function is described as a set of inputs, the behavior, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioral requirements describing all cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements(also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

As defined in requirements engineering, functional requirements specify particular results of a system. This should be contrasted with non-functional requirements which specify overall characteristics such as cost and reliability. Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.

- Functional Requirements concerns with the specific functions delivered by the system. So, Functional requirements are statements of the services that the system must provide.
- The functional requirements of the system should be both complete and consistent
- Completeness means that all the services required by the user should be defined.
- Consistency means that requirements should not have any contradictory definitions.
- The requirements are usually described in a fairly abstract way. However, functional system requirements describe the system function in details, its inputs and outputs, exceptions and so on.
- Take user id and password match it with corresponding file entries. If a match is found then continue else raise an error message.

3.2 NON-FUNCTIONAL REQUIREMENTS

- Non-functional Requirements refer to the constraints or restrictions on the system. They may relate to emergent system properties such as reliability, response time and store occupancy or the selection of language, platform, implementation techniques and tools.
- The non-functional requirements can be built on the basis of needs of the user, budget

constraints, organization policies and etc.

1. **Performance requirement:** All data entered shall be up to mark and no flaws shall be there for the performance to be 100%.
2. **Platform constraints:** The main target is to generate an intelligent system to predict the adult height.
3. **Accuracy and Precision**: Requirements are accuracy and precision of the data
4. **Modifiability:** Requirements about the effort required to make changes in the software. Often, the measurement is personnel effort (person-months).
5. **Portability:** Since mobile phone is handy so it is portable and can be carried and used when ever required.
6. **Reliability**: Requirements about how often the software fails. The definition of a failure must be clear .Also, don't confuse reliability with availability which is quite a different kind of requirement. Be sure to specify the consequences of software failure, how to protect from failure a strategy for error Prediction, and a strategy for correction.
7. **Security**: One or more requirements about protection of your system and its data.
8. **Usability:** Requirements about how difficult it will be to learn and operate the system. The requirements are often expressed in learning time or similar metrics.

ACCESSIBILITY:

Accessibility is a general term used to describe the degree to which a product, device, service, or environment is accessible by as many people as possible. In our project people who have registered with the cloud can access the cloud to store and retrieve their data with the help of a secret key sent to their email ids. User interface is simple and efficient and easy to use.

MAINTAINABILITY:

In software engineering, maintainability is the ease with which a software product can be modified in order to include new functionalities can be added in the project based on the user requirements just by adding the appropriate files to existing project using .net and programming languages. Since the programming is very simple, it is easier to find and correct the defects and to make the changes in the project.

SCALABILITY:

System is capable of handling increase total throughput under an increased load when resources (typically hardware) are added. System can work normally under situations such as low bandwidth and large number of users.

PORTABILITY:

Portability is one of the key concepts of high-level programming. Portability is the software code base feature to be able to reuse the existing code instead of creating new code when moving software from an environment to another. Project can be executed under different operation conditions provided it meet its minimum configurations. Only system files and dependent assemblies would have to be configured in such case.

VALIDATION:

It is the process of checking that a software system meets specifications and that it fulfils its intended purpose. It may also be referred to as software quality control. It is normally the responsibility of software testers as part of the software development lifecycle. Software validation checks that the software product satisfies or fits the intended use (high-level checking), i.e., the software meets the user requirements, not as specification are facts or as needs of those who will operate the software only; but, as the needs of all the stakeholders.

3.2 HARDWARE REQUIREMENTS

- System : Pentium 4, Intel Core i3, i5, i7 and 2 GHz Minimum
- RAM : 512Mb or above
- Hard Disk : 10 GB or above
- Input Device : Keyboard and Mouse
- Output Device : Monitor or PC

3.3 SOFTWARE REQUIREMENTS

- Operating System : Windows 7, 10 or Higher Versions
- Platform : Jupiter Notebook, VS Code
- Front End : Web3
- Back End : Python and Files
- Programming Lang : Python, HTML, CSS, JS

CHAPTER 4

DESIGN

4.1 DESIGN GOALS

The Design goals consist of various design which we have implemented in our system Multiple Disease Prediction using machine learning. This system has built with various designs such as data flow diagram, sequence diagram, class diagram, use case diagram, component diagram, activity diagram, state chart diagram, deployment diagram. After doing these various diagrams and based on these diagrams we have done our project.

We have designed our system in such a way that whenever user log in into the system, the user has to register to the system, and new user cannot use the system without registering in the system. After that for registration the user requires basic credentials such as username, age, email, phone, password. Then the user has to login to the system using the same username and password. Here are the things that this system can perform.

a. Entering Symptoms

b. Disease Prediction

Entering Symptoms: Once user successfully logged in to the system then he/she has to select the symptoms from the given drop-down menu.

Disease prediction: The predictive model predicts the disease of a person he might have, based on the user entered symptoms.

4.2 SYSTEM ARCHITECTURE

Multiple 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 architecture of the system Multiple 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 Multiple Disease Prediction model using all the inputs from the user that is mentioned above. Then after user entering 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.

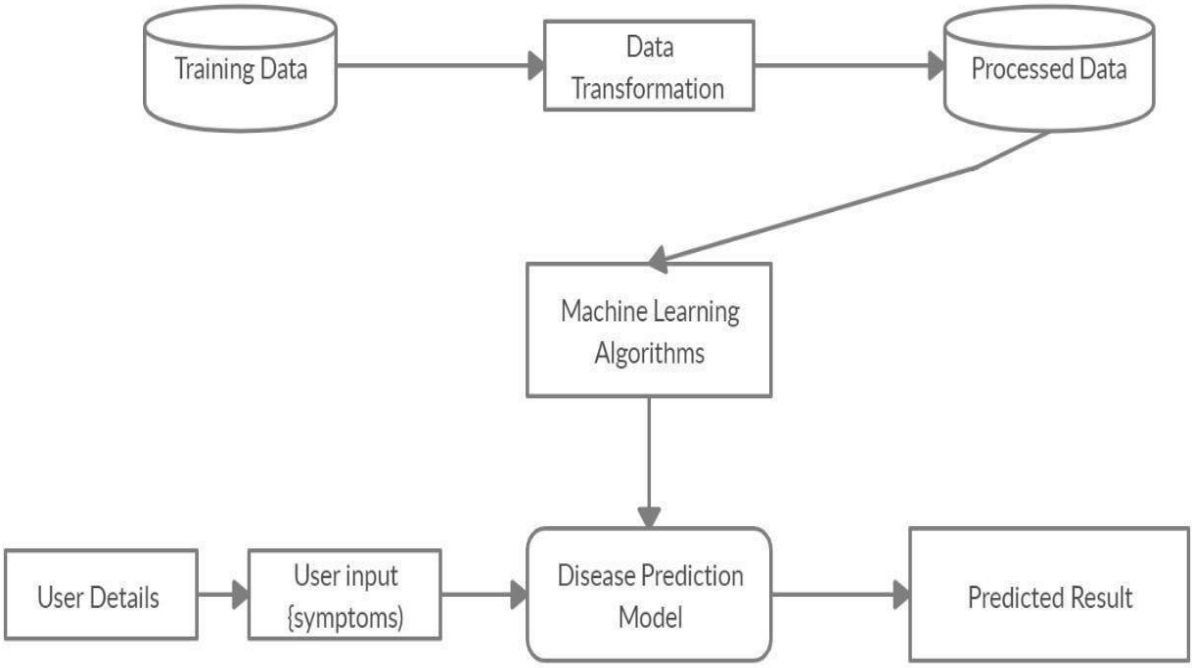


Fig 4.2 System Architecture

4.3 DATA FLOW DIAGRAM

The dataflow diagram of the project Multiple Disease Prediction using machine learning consist of all the various aspects a normal flow diagram requires. This dataflow diagram shows how from starting the model flows from one step to another, like he enter 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 mod eland if true is predicts the appropriate results otherwise it shows the details where the user if gone wrong while entering the information.

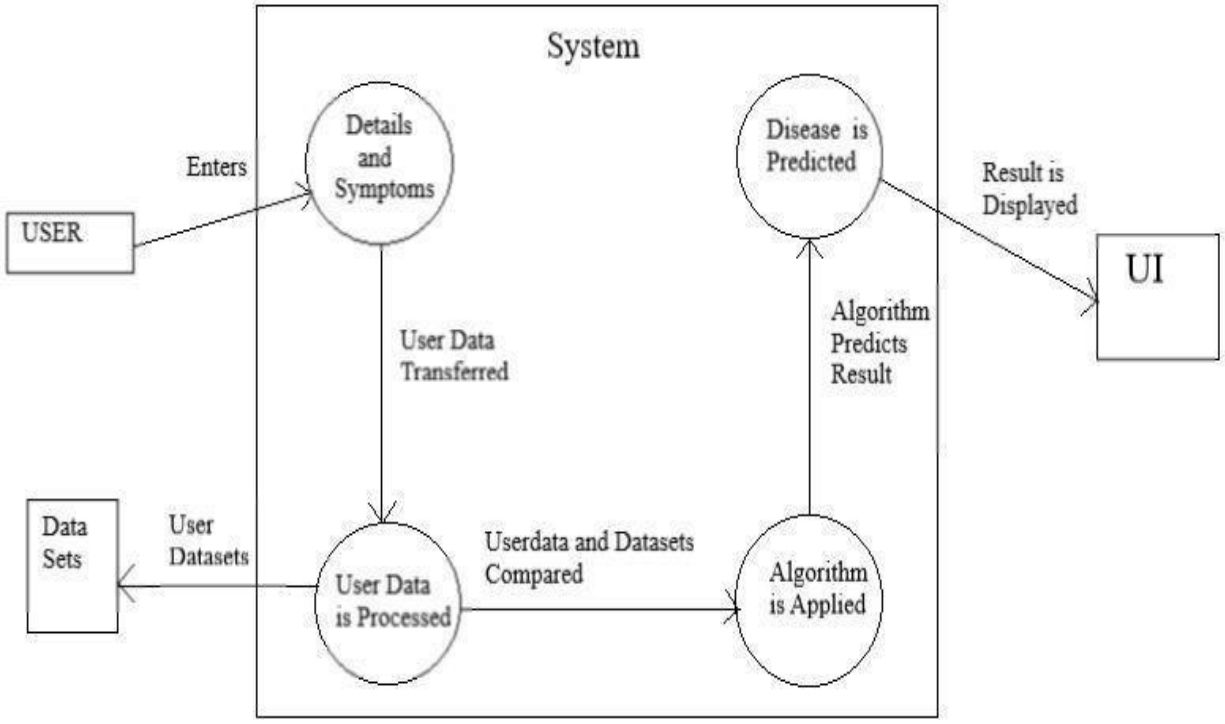


Fig 4.3 Data Flow Diagram

4.4 CLASS DIAGRAM

Multiple Disease Prediction using machine learning consist of class diagram that all the other application that consists the basic class diagram, here the class diagram is the basic entity that is required in order to carry on with the project. Class diagram consist information about all the classes that is used and all the related datasets, and all the other necessary attributes and their relationships with other entities, all these information is necessary in order to use the concept of the prediction, where the user will enter all necessary information such as user name, email, phone number, and many more attributes that is required in order to login into the system and using the files concept we will store the information of the users who are registering into the system and retrieves those information later while logging into the system.

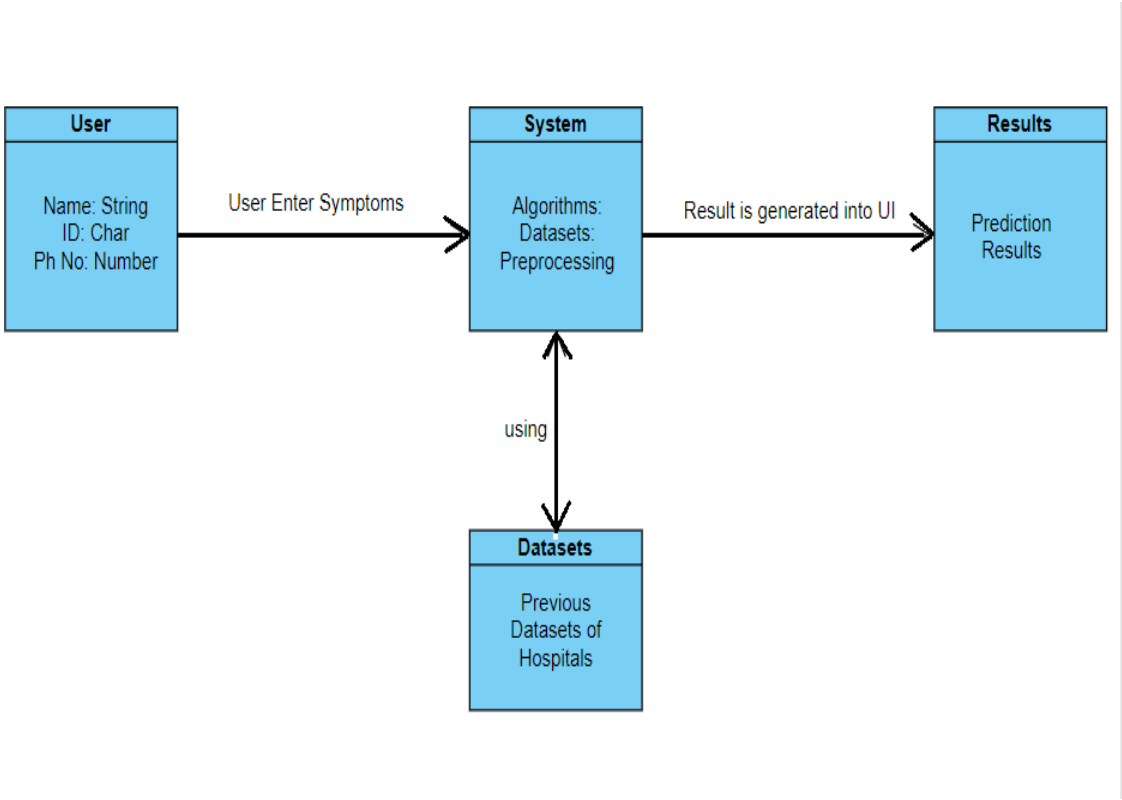


Fig 4.4 Class Diagram

4.5 SEQUENCE DIAGRAM

The Sequence diagram of the project Multiple Disease Prediction using machine learning consist of all the various aspects a normal sequence diagram requires. This sequence diagram shows how from starting the model flows from one step to another, like he enter 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's and it also shows the appropriate precautionary measure for the user to follow. Here the sequence of all the entities are linked to each other where the user gets started with the system.

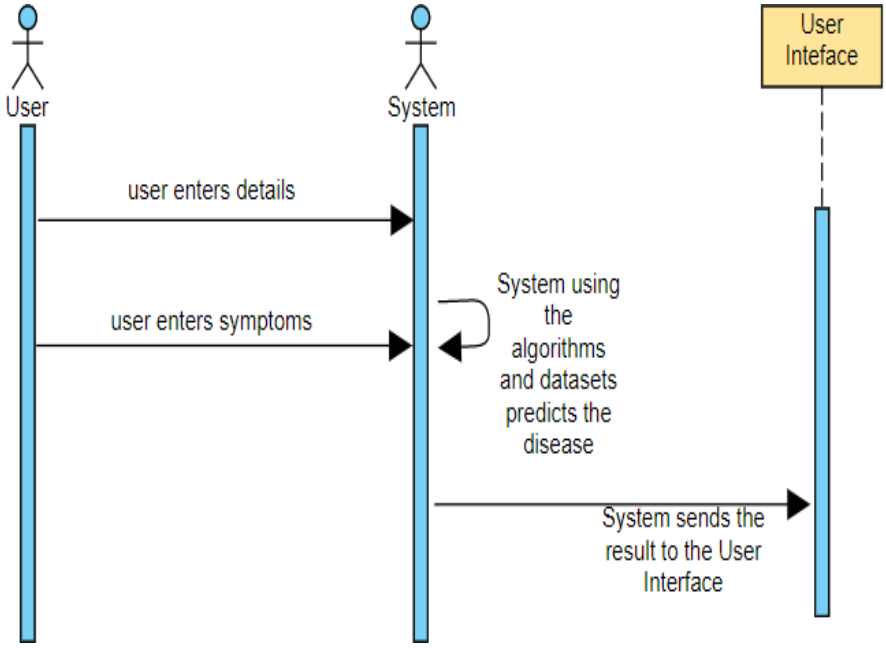


Fig 4.5 Sequence Diagram

4.6 USE CASE DIAGRAM

The Use Case diagram of the project Multiple Disease Prediction using machine learning consist of all the various aspects a normal use case diagram requires. This use case diagram shows how from starting the model flows from one step to another, like he enter 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's and it also shows the appropriate precautionary measure for the user to follow. Here the use case diagram of all the entities are linked to each other where the user gets started with the system.

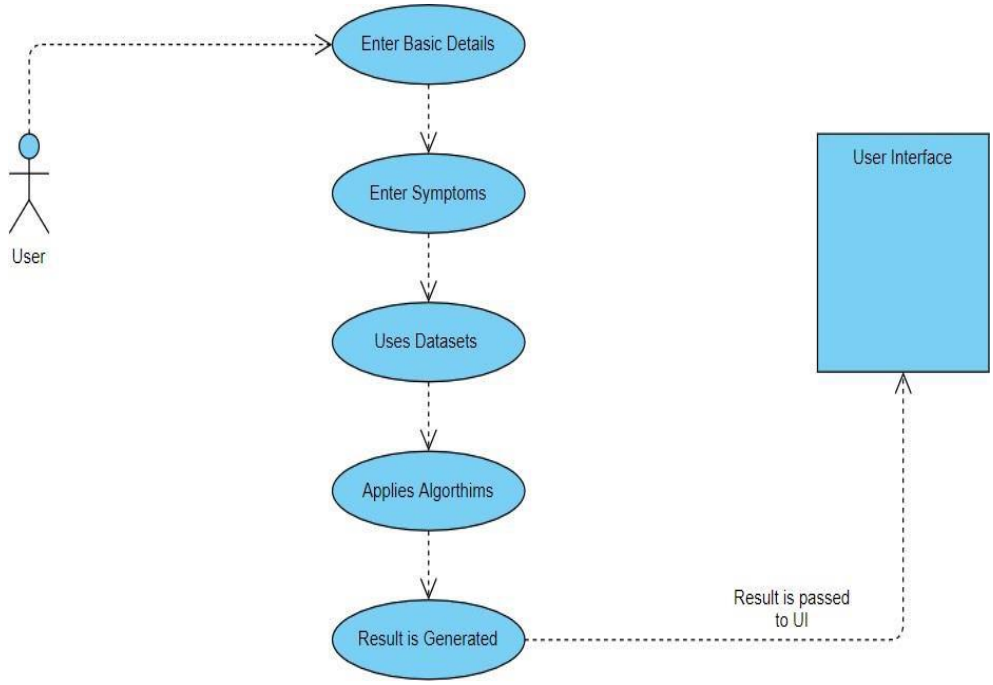


Fig 4.6 Use Case Diagram

4.7 ACTIVITY DIAGRAM

Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. The control flow is drawn from one operation to another. Here in this diagram the activity starts from user where the user registers into the system then login using the credentials and then the credentials are matched in the system and if its true, then the user proceeds to the prediction phase where the prediction happens. Then finally after processing the data from datasets the analysis will happen then the correct result will be displayed that is nothing but the Output.

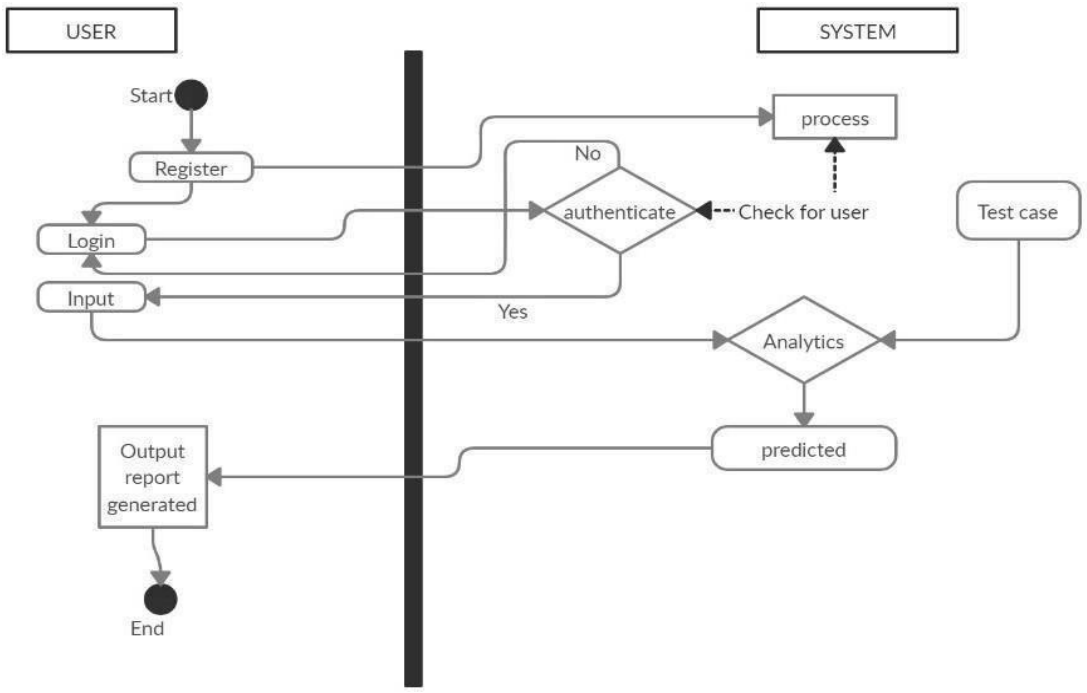


Fig 4.7 Activity Diagram

4.8 COMPONENT DIAGRAM

A component diagram, also known as a UML component diagram, describes the organization and wiring of the physical components in a system. Component diagrams are often drawn to help model implementation details and double-check that every aspect of the system's required function is covered by planned development. Here component diagram consists of all major components that is used to built a system. So, Design, Algorithm, File System and Datasets all are linked to one another. Datasets are used to compare the results and algorithm is used to process those results and give a correct accuracy and design UI is used to show the result in an appropriate way in the system and file system is used to store the user data. So, like this all components are interlinked to each other.

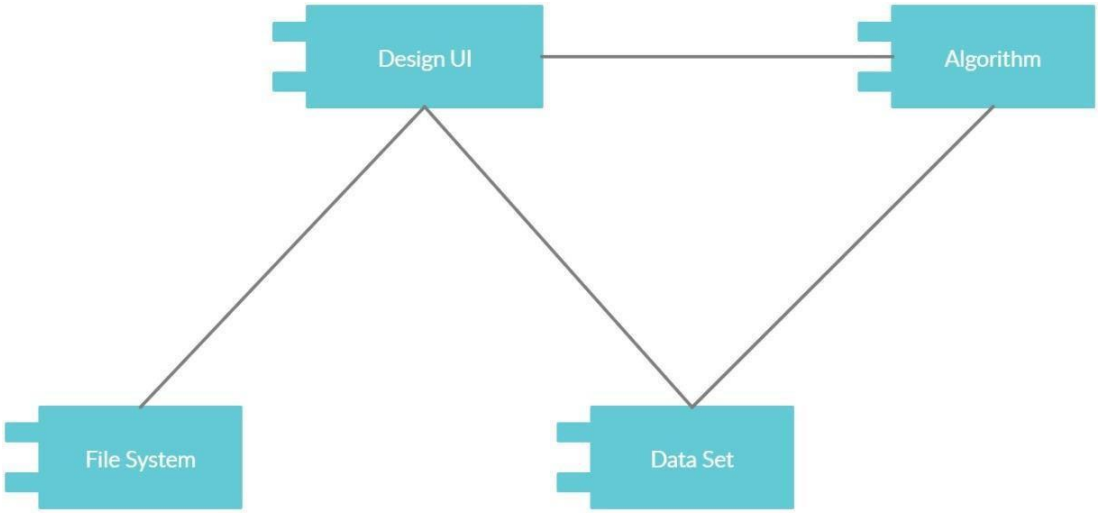


Fig 4.8 Component Diagram

4.9 STATE CHART DIAGRAM

A State chart diagram describes the behavior of a single object in response to a series of events in a system. Sometimes it's also known as a Hare l state chart or a state machine diagram. This UML diagram models the dynamic flow of control from state to state of a particular object within a system. It is similar to activity diagram but here there are only few rules like how it starts and how it end all are denoted with the help of the symbol given below, the system starts with the registration and then login comes, if the login is successfulthenitwillgotothenextstepandifloginisincorrectthencomesbacktosame page stating incorrect details. After the successful login the user needs to enter the symptoms and then press the prediction button, at the same time the backend process will do their work and the correct result is predicted.

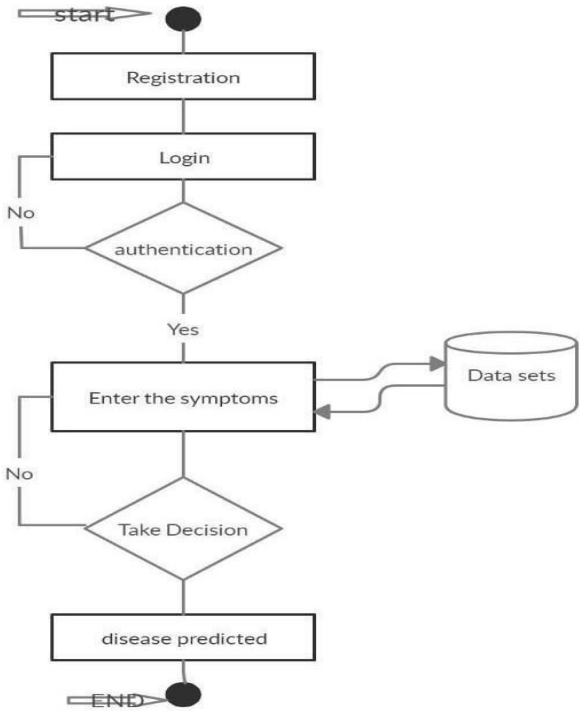


Fig 4.9 State Chart Diagram

4.10 COLLABORATION DIAGRAM

A collaboration diagram, also known as a communication diagram, is an illustration of the relationships and interactions among software objects in the Unified Modelling Language (UML). These diagrams can be used to portray the dynamic behavior of a particular use case and define the role of each object. Here this diagram shows how all the models are connected to show the correct result starting from user, where he opens the system then using the system he does registration and that registration data is saved into file system and the using those data user logs in to the system and then he provides all the necessary information in order to get the accurate result, then system evaluates the user entered information and finally gives the correct result.

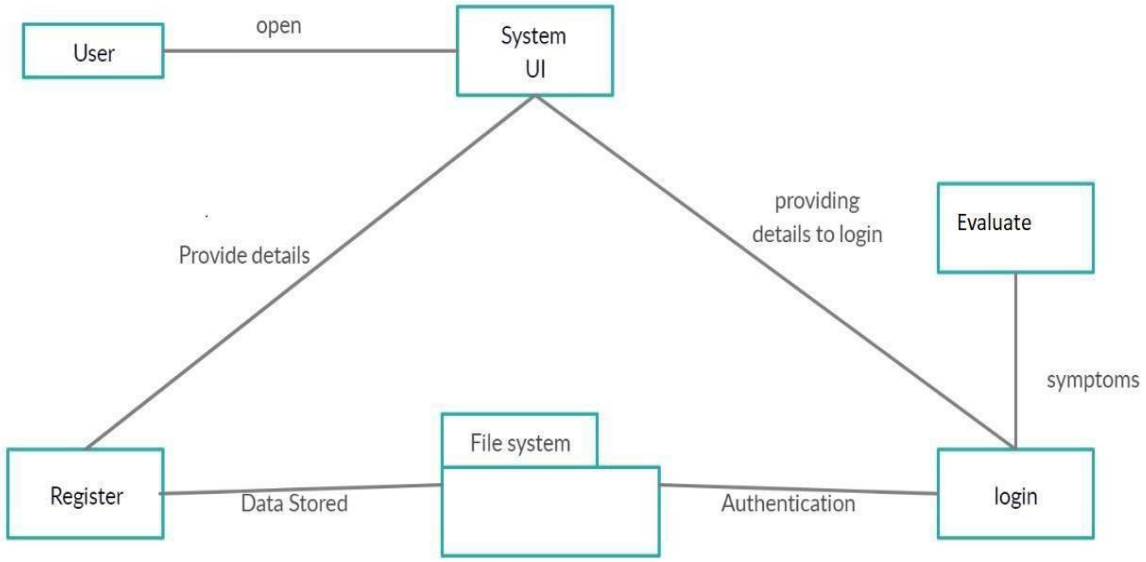


Fig 4.10 Collaboration Diagram

4.11 DEPLOYMENT DIAGRAM

A deployment diagram shows the configuration of run time processing nodes and the components that live on them. Deployment diagrams is a kind of structure diagram used in modelling the physical aspects of an object-oriented system. Here the deployment diagram show the final stage of the project and it also shows how the model looks like after doing all the processes and deploying in the machine. Starting from the system how it processes the user entered information and then comparing that information with the help of datasets, then training and testing those data using the algorithms such as decision tree, naïve Bayes, random forest. Then finally processing all those data and information the system gives the desired result in the interface.

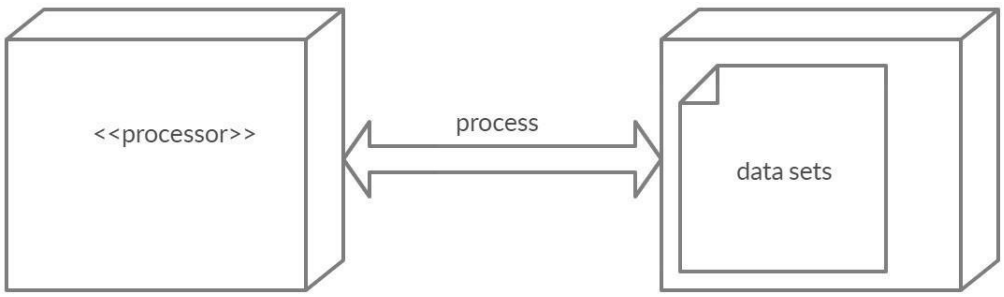


Fig 4.11 Deployment Diagram

4.12 INTERFACE AND FRAMEWORK DIAGRAM

- Below is the structure which we will use in our project Multiple Disease Prediction using Machine learning.
- The User Interface of this system consists of Python's library interface called flask
- Then it goes into the framework model where all the actions and services are combined and then the result is processed.
- Italsoconsistsoffilesystemwherealltheuserrelatedinformationisstoredsuch as username, password, age, phone, email.
- Below is the structure of the User Interface along with necessary implementations.

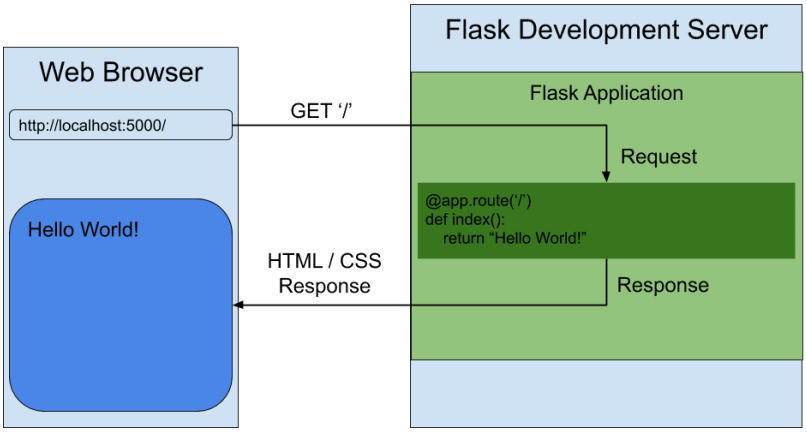


Fig 4.12 Interface and Framework Diagram

- After the User Interface it consist of the framework in which the system works accordingly using all the technologies, algorithms and various tools in which the project works accordingly.
- The framework consists of all the modules starting from the data preparation, data building and assessment stage.
- All these three factors are then going into the data collection phase, where the data is classified accordingly using the appropriate models and algorithms such as decision tree, naïve bayes, random forest.
- Then all those algorithms use the datasets and it forms the sets where all the previous data is stored, then using that data it compares with the new data and result is generated.
- Then pre-processing work will happen to reduce and analyze the data that is present in the system.
- Then with the help of UI the data is transferred into the main screen.
- Then later all those data are analyzed and validated then the final result is generated.
- Finally, after user enters the symptoms, all backend mechanisms works and the predicted result is displayed in the User Interface.

CHAPTER 5

IMPLEMENTATION

5.1 OVERVIEW

To implement a Multiple Disease Predictor using Flask, Random Forest algorithm, and Jupyter Notebook, you can follow these steps:

1. Prepare the Dataset: Collect or obtain a suitable dataset that contains relevant features and target labels for multiple diseases. Ensure the dataset is properly cleaned and preprocessed.

2. Train the Random Forest Model: In a Jupyter Notebook, import the necessary libraries such as scikit-learn and pandas. Load the dataset and split it into features (X) and target labels (y). Apply any necessary preprocessing steps such as data normalization or handling missing values. Import the Random Forest Classifier from scikit-learn and train the model using the dataset.

3. Save the Trained Model: After training the Random Forest model, save it using the pickle library. This will allow you to load and use the trained model later in your Flask application.

4. Set up the Flask Application: Create a new directory for your Flask project. In this directory, create a Python file (e.g., app.py) to define your Flask application. Install Flask and other necessary dependencies.

5. Import Libraries and Load the Model: In the app.py file, import the required libraries such as Flask and pickle. Load the trained Random Forest model using the pickle library.

6. Define Routes and Views: Define the routes and associated view functions in the app.py file. For example, you can have a route for the home page ("/") and a route for the disease prediction ("/predict"). In the view function for the prediction route, retrieve the input values from the user, preprocess the input data (if required), and use the loaded model to make predictions. Return the prediction results to the user.
7. Set up HTML Templates: Create HTML templates (e.g., using Jinja2) to define the structure and layout of the web pages. Create templates for the home page, prediction form, and result display.
8. Render Templates and Handle Form Submission: In the view functions, use the render_template function from Flask to render the appropriate HTML templates. Create a form in the prediction template to collect input from the user. Handle the form submission in the corresponding view function and pass the input data to the prediction function that uses the loaded Random Forest model.

9. Run the Flask Application: At the end of the app.py file, add code to run the Flask application.

For example:

```
python
if __name__ == "__main__":
    app.run(debug=True)
```

10. Test the Application: Start the Flask application by running the app.py file. Open a web browser and visit the specified URL (e.g., http://localhost:5000). Test the Multiple Disease Predictor by entering the required inputs and submitting the form.
11. This implementation allows users to interact with the Flask application through a web interface, enter input data related to symptoms or other relevant features, and receive predictions for multiple diseases based on the trained Random Forest model.

Note: Make sure to handle errors, validate user input, and ensure the security of your application as necessary.

5.2 DECISION TREE ALGORITHM

Decision tree induction is the learning of decision trees from class-labelled training tuples. A decision tree is a flowchart-like tree structure,

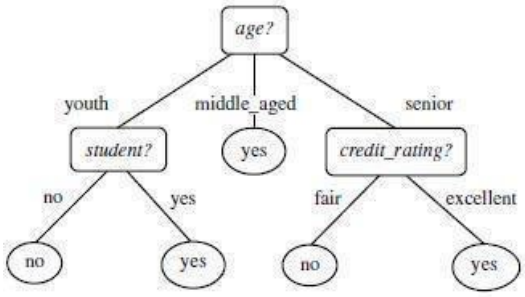


Fig 5.2.1 Decision Tree problem

- Decision tree induction is a non-parametric approach for building classification models.
- Finding an optimal decision tree is an NP-complete problem
- Techniques developed for constructing decision trees are computationally inexpensive, making it possible to construct models even when the training set size is very large.
- Decision trees, especially smaller-sized trees, are relatively easy to interpret.
- Decision tree provide an expressive representation for learning discrete- valued functions.
- Decision tree algorithms are quite robust to the presence of noise, especially when methods for avoiding overfitting.

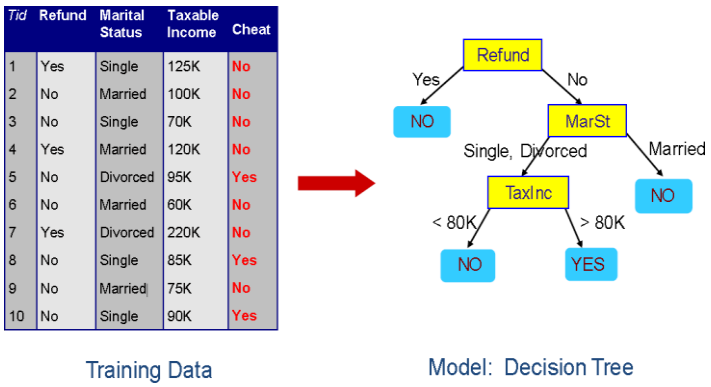


Fig 5.2.2 Decision Tree Example

- The presence of redundant attributes does not adversely affect the accuracy of decision tree.

- The construction of decision tree classifiers does not require any domain knowledge or parameter setting, and therefore I appropriate for exploratory knowledge discovery. Decision trees can handle high dimensional data.
- Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans.
- The learning and classification steps of decision tree induction are simple and fast.
- In general, decision tree classifiers have good accuracy.
- Decision tree induction algorithm shave been used for classification in many application areas, such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology.

Algorithm: Generate_decision_tree. Generate a decision tree from the training tuples of data partition *D*.

Input:

- Data partition, *D*, which is a set of training tuples and their associated class labels;
- *attribute_list*, the set of candidate attributes;
- *Attribute_selection_method*, a procedure to determine the splitting criterion that "best" partitions the data tuples into individual classes. This criterion consists of a *splitting_attribute* and, possibly, either a *split point* or *splitting subset*.

Output: A decision tree.

Method:

```
(1)  create a node N;
(2)  If tuples in D are all of the same class, C then
(3)      return N as a leaf node labeled with the class C;
(4)  If attribute_list is empty then
(5)      return N as a leaf node labeled with the majority class in D; // majority voting
(6)  apply Attribute_selection_method(D, attribute_list) to find the "best" splitting_criterion;
(7)  label node N with splitting_criterion;
(8)  If splitting_attribute is discrete-valued and
         multiway splits allowed then // not restricted to binary trees
(9)      attribute_list ← attribute_list − splitting_attribute; // remove splitting_attribute
(10) for each outcome j of splitting_criterion
     // partition the tuples and grow subtrees for each partition
(11)     let Dj be the set of data tuples in D satisfying outcome j; // a partition
(12)     If Dj is empty then
(13)         attach a leaf labeled with the majority class in D to node N;
(14)     else attach the node returned by Generate_decision_tree(Dj, attribute_list) to node N;
     endfor
(15) return N;
```

Fig 5.2.3 Decision Tree Algorithm

5.3 RANDOM FOREST ALGORITHM

- It is an ensemble classifier using many decision trees models; it can be used for regression as well as classification.
- Accuracy and variable importance information can be provided with the results.
- A random forest is the classifier consisting of a collection of tree structured classifiers k, where the k is independently, identically distributed random trees and each random tree consist of the unit of vote for classification of input.
- Random forest uses the Gini index for the classification and determining the final class in each tree.
- The final class of each tree is aggregated and voted by the weighted values to construct the final classifier.
- The working of random forest is, A random seed is chosen which pulls out at random, a collection of samples from the training datasets while maintaining the class distribution.

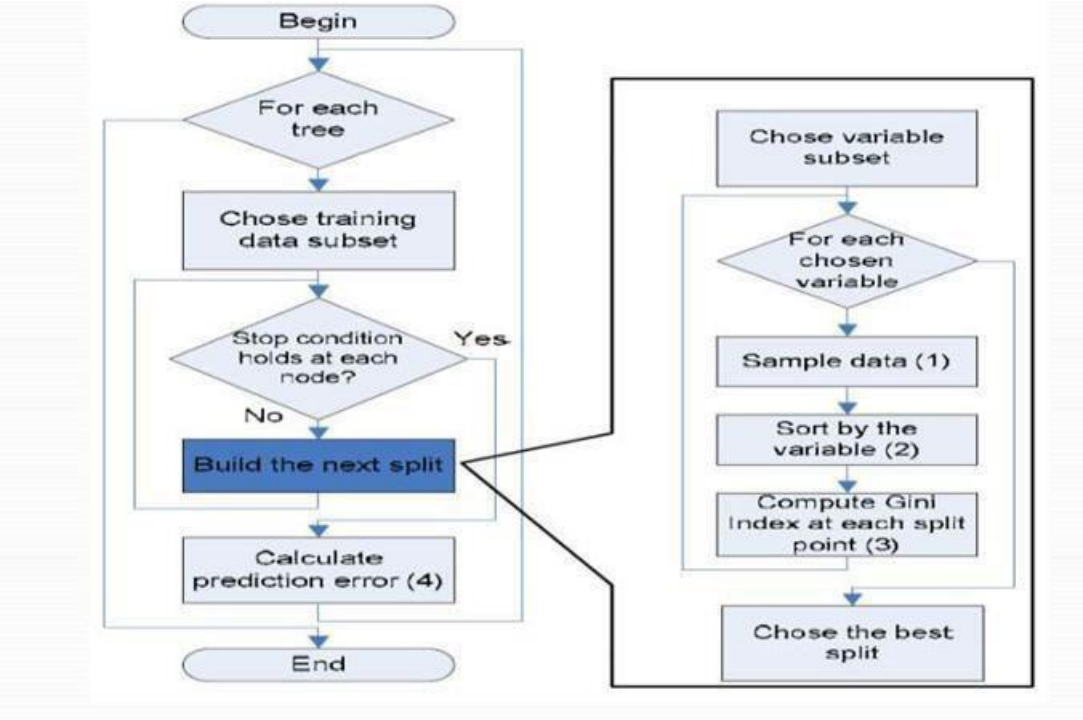


Fig 5.3 Random Forest Example

CHAPTER 6

RESEARCH PAPER

Abstract - Multi Malady Prediction" framework based on prescient modeling predicts the malady of the client on the premise of the side effects that client gives as an input to the framework. The framework analyzes the side effects given by the client as input and gives the likelihood of the illness as an yield Illness Forecast is done by executing the irregular woodland classifier conjointly , we actualize Jungle fever and Pneumonia utilizing Profound Learning Model(CNN) and within the prosed strategy it gives the superior precision and we plan web assignment for expectation system.

Index Terms - Irregular Woodland, Unremitting Malady, CNN, Prescient Modeling.

I.INTRODUCTION

If someone is sick, they need to go to a doctor, but this can be costly and take a while. It is hard for people if they cannot see doctors or go to hospitals when they are sick, because their illness may not be found. If we can use a computer to do the procedure quickly and cheaply, it is good for the person and makes the process easier. There are computer programs that check how dangerous heart disease is for someone by looking at their medical information. Disease Predictor is a tool that can tell you what sickness you might have using the symptoms you tell it. The Disease Prediction system got information from different websites about health. You can find out if you're likely to have a disease by telling a machine your symptoms with a Disease Predictor. People like to learn new things, especially now that more and more people are using the internet. When there is a problem, many people want to search for it on the internet. Doctors and hospitals don't have as much access to the Internet as regular people do. feel well and their body is not healthy.

You have lots of choices. This system can help people. Chronic illness means having a sickness that stays for a long time or takes a long time to get better. Most chronic illnesses cannot be fully cured; however, it is possible to manage them through daily treatment. India, just like other countries, is changing a lot. These changes are leading to many people getting heart disease quickly. A lot of countries, even India, are facing problems with long-term diseases like heart disease and diabetes. This can be bad for the whole world's health, safety, and money. Many cities are growing quickly and there are many jobs and different ways of living now. Many people in all countries are now suffering from chronic diseases. It affects about one-third of the population in each country. It is costly and hard for people with long-term illnesses to receive treatment. In medicine, lots of information about long-term illnesses is collected and analyzed by using data mining to help find diseases early. Some illnesses like heart disease, diabetes, liver problems, Alzheimer's, and Parkinson's are very expensive to treat.

It is difficult for medical and healthcare industries to give good care to all patients. Only people who can pay for it can take advantage of it. There is a lot of healthcare information that people are not using well to make good choices. We need to do a better job of finding important information in that data. We are using computers to find diseases like diabetes, cancer, and heart disease before they become too serious. Machine learning is when computers learn from examples or past data to do a better job. Machine learning is when computers are taught to learn by using data and past experiences. When we use a machine to learn things, there are two parts called Training and Testing. Doctors use signs and past medical information to guess which disease a patient may have. It has been hard to use machines to do this job for a long time. Machine learning helps doctors to fix health problems quickly and effectively in the medical field.

II. RESEARCH OBJECTIVE

We need to create a system that helps people guess if they have a long-lasting sickness without going to the doctor. To find out different illnesses by looking at how the patients feel and using different ways of teaching computers. There is no correct way to deal with written words and organized information. The new plan will look at both organized and unorganized information. "Using Machine Learning can make predictions more accurate. "

III. LITERATURE REVIEW

The think about for the finest restorative conclusion mining method was performed by K.M. Al-Aidaroos, A.A. Bakar, and Z. Othman. For this ponder, the creators compared Nave Baeyes to five other classifiers: LR, KStar (K*), Choice Tree (DT), Neural Organize (NN), and a fundamental rule-based calculation (ZeroR). The proficiency of all calculations was assessed utilizing 15 real-world therapeutic issues from the UCI machine learning store (Asuncion and Newman, 2007). Within the try, NB beated the other calculations in 8 of the 15 information sets, driving to the conclusion that the prescient exactness comes about in Nave Baeyes are predominant to other methods. Darcy A. Davis, Nitesh

Chawla, Nicholas Blumm, Nicholas Christakis, and Albert-Laszlo Barabasi found that treating inveterate sickness at a worldwide level is not one or the other time nor fetched viable. As a result, the creators performed this ponder in arrange to estimate potential illness hazard. CARE (which employments as it were a patient's therapeutic history and ICD- 9-CM codes to foresee conceivable infection risks) was utilized for this. Based on their claim restorative history which of comparative patients, CARE consolidates collective sifting approaches with clustering to foresee each patient's most prominent malady dangers. ICARE, an iterative form that coordinating outfit standards for progressed effectiveness, has moreover been characterized by the authors.

These cutting-edge frameworks do not require any progressed information and can foresee a wide extend of therapeutic conditions in a single run. ICARE's exceptional potential hazard scope implies more exact early cautions for thousands of ailments, a few a long time ahead of time. When utilized to its full degree, the CARE framework can be

utilized to examine a more extensive run of malady foundations, raise already unconsidered questions, and encourage dialogs with respect to early location and prevention.

This term paper was composed by JyotiSoni, Ujma Ansari, Dipesh Sharma, and SunitaSoni to supply a study of existing strategies of data revelation in databases utilizing information mining procedures that are utilized in today's therapeutic inquire about, particularly in Heart Illness Expectation. A number of tests have been carried out to compare the execution of prescient information mining procedures on the same dataset, and the comes about appear that Choice Tree beats, with Bayesian classification having comparable precision to Choice Tree in a few cases, but other predictive approaches such as KNN, Neural Systems, and Classification based on Clustering underperform. Shadab Adam Pattekari and Asma Parveen conducted a consider to anticipate heart infections utilizing the Choice Tree Algorithm, in which the shopper gives information that's compared to a qualified set of values. As a result of this consider, patients were able to supply essential data that was compared to information, and heart malady was anticipated. M.A.NisharaBanu and B. Gomathy examined the different sorts of heart-related issues utilizing restorative information mining methods such as affiliation run the show mining, gathering, and clustering I. The point of a choice tree is to appear any possible outcome of a decision. To realize the leading result, various rules are concocted. The criteria utilized in this think about were age, sex, smoking, being overweight, drinking liquor, blood sugar, heart rate, and blood weight. The chance level for different parameters is spared with their ids extending from 1 to 100. (1-8). The standard level of expectation is spoken to by IDs less than 1, while higher IDs other than 1 speak to higher chance levels. The design within the dataset is considered utilizing the K-means clustering method. The calculation isolates the information into k bunches. The closed cluster is designated to each point within the dataset. Each cluster middle is recalculated as the normal of the cluster's points.

IV PROPOSED SYSTEM

We have mixed structured and unstructured data in the healthcare fields to determine disease risk in this project. The use of a latent factor model to recreate missing data in medical records obtained from online sources. We could also assess the major chronic diseases in a specific area and population using statistical information. We consult hospital experts to learn about useful features when dealing with structured data. In the case of unstructured text files, we use the randrom forest algorithm to automatically select features.

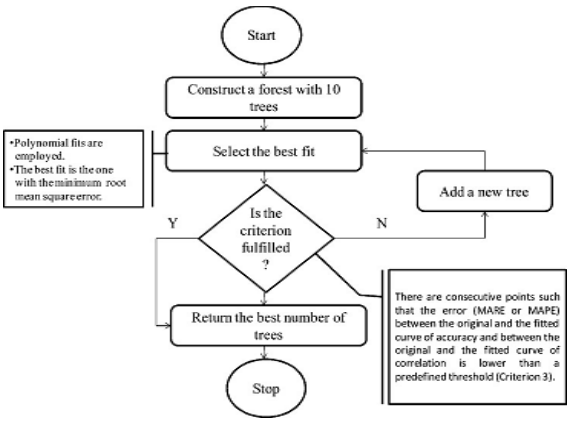


Fig 1: - System Model

4.1 Data collection

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 different health related websites.

4.2 Data Preprocessing

Before feeding the data into the Prediction model, following data cleaning and

preprocessing steps are performed

- Checking null values and filling using forward fillmethod
- Converting data into different cases
- Standardizing the data using mean and standard deviation
- Splitting the dataset into training and testing sets

4.3 Building Model

Many methods are used to perform data mining. Machine learning is one of the approaches. Random forest Machine learning strategies include grouping, clustering, summarization, and many others. Since classification techniques are used in this project, classification is one of the data mining processes in this phase of categorical data classification. And this step is divided into two phases: training and testing. Inthe training phase, predetermined data and associated class labels are used for classification. The training stage is often referred to as supervised learning. The preparation and testing phases of the classification process are depicted in the diagram. In the training process, training tuples are used, and in the test data phase, test data tuples are used, and the classification rule's accuracy is calculated. Assume that the classification rule's accuracy on testing data issufficient for the rule to be used for classification of unmined data.

4.4 Prediction:

Prediction using Random Forest: -
Prediction done by Random Forest Model using Flask framework model trained by training chronic disease dataset

4.5Algorithm

4.4.1 Random Forest
AlgorithmInput:
Dataset
Output: Predicted class label

Step 1 : Set Number of classes = N, Number of features =M

Step2 : Let „m" determine the number of features at anode of decision tree, (m < M)

Step3 : For each decision tree do
Select randomly: a subset (with replacement) of training data that represents the N classes and use the rest of data to measure the error of the tree

Step4 : For Each node of this tree do Select randomly: m features to determine the decision at this node and calculate the best split accordingly.

Step5:
End for

4.2.2 CNN(CONVOLUTIONAL NEURALNETWORK):

A convolutional neural network is a feed-forward neural network that is generally used to analyze visualimages by processing data with grid-like topology. It'salso known as a ConvNet. A convolutional neural network is used to detect and classify objects in an image. A convolution neural network has multiplehidden layers that help in extracting information froman image.
The four important layers in CNN are:

1. Convolution layer
2. ReLU layer
3. Pooling layer
4. Fully connected layer Convolution Layer

This is the first step in the process of extracting valuable features from an image. A convolution layerhas several filters that perform the convolution operation. Every image is considered as a matrix of pixel values

4.2.2.1 ReLU layer

ReLU stands for the rectified linear unit. Once the feature maps are extracted, the next step is tomove them to a ReLU layer. ReLU performs an element- wise operation and sets all the negative pixels to 0. It introduces non-linearity to the network, and the generated output is a rectified feature map.

4.2.2.2 Pooling Layer

Pooling is a down-sampling operation that reduces the dimensionality of the feature map. The rectified feature map now goes through a pooling layer togenerate a pooled feature map.

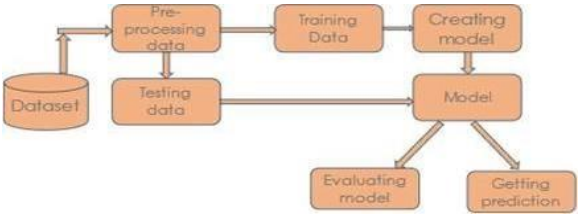


FIG 2: SYSTEM ARCHITECTURE

VII. CONCLUSION

The aim of this project is to predict disease based on symptoms. The project is set up in such a way that thedevice takes the user's symptoms as input and generates an output, which is disease prediction. A prediction accuracy probability of 95% is obtained on average. The grails system was used to successfully incorporate Disease Predictor.

REFERENCE

[1] A.Davis, D., V.Chawla, N., Blumm, N., Christakis, N., & Barbasi, A. L. (2008). Predicting Individual Disease Risk Based on Medical History.

[2] Adam, S., & Parveen, A. (2012). Prediction System for Heart Disease Using Naive Bayes.

[3] Al-Aidaroos, K., Bakar, A., & Othman, Z. (2012). Medical Data Classification with Naive Bayes Approach. *Information Technology Journal* .

[4] Darcy A. Davis, N. V.-L. (2008). Predicting Individual Disease Risk Based on Medical History.

[5] JyotiSoni, Ansari, U., Sharma, D., & Soni, S.(2011). Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction.

[6] K.M. Al-Aidaroos, A. B. (n.d.). K.M. Al- Aidaroos, A. B. (n.d.). 2012. *Medical Data Classification with Naive Bayes Approach* .

[7] NisharBanu, MA; Gomathy, B.; (2013). Disease Predicting System Using Data Mining Techniques.

[1] Al-Aidaroos, K., Bakar, A., & Othman, Z. (2012). Medical Data Classification with Naive Bayes Approach. *Information Technology Journal* .

[2] Darcy A. Davis, N. V.-L. (2008). Predicting Individual Disease Risk Based on Medical History.

[3] JyotiSoni, Ansari, U., Sharma, D., & Soni, S.(2011). Predictive Data Mining for Medical Diagnosis: An Overview of Heart Disease Prediction.

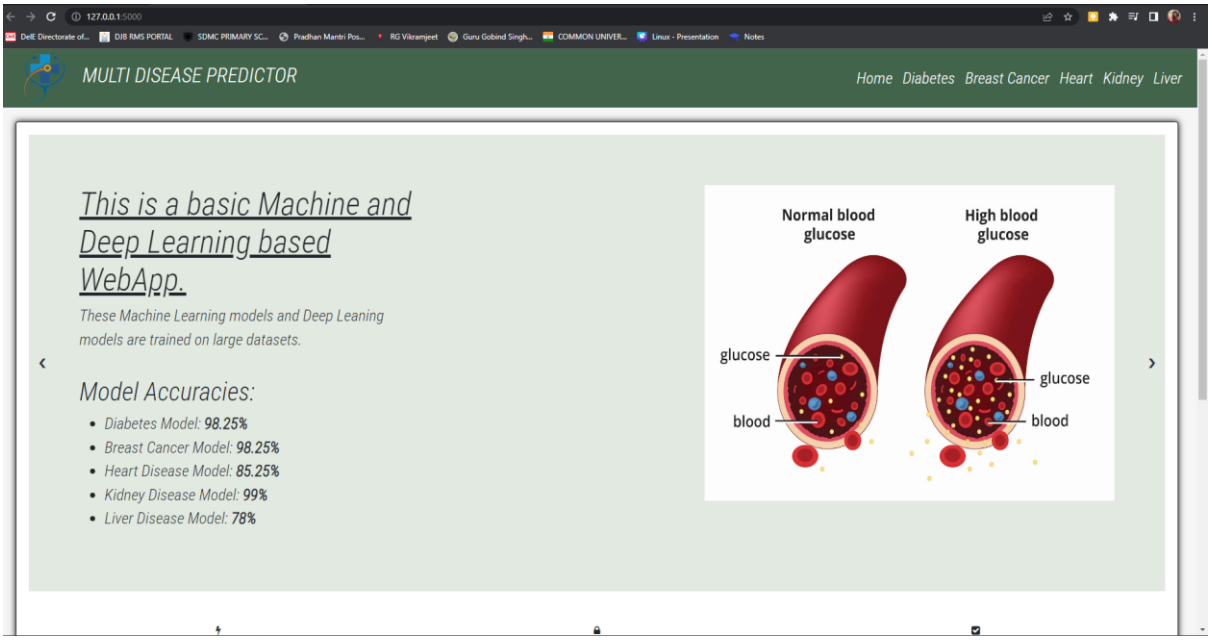
[4] K.M. Al-Aidaroos, A. B. (n.d.). K.M. Al- Aidaroos, A. B. (n.d.). 2012. *Medical Data Classification with Naive Bayes Approach* .

NisharBanu, MA; Gomathy, B.; (2013). Disease Predicting System Using Data Mining Techniques.

CHAPTER 7

CODING AND SNAPSHOTS

7.1 HOME PAGE



CODE:

```
{% extends 'main.html' %} {% block content %} {% if message %}
<div class="alert alert-danger">{{ message }}</div>
{% endif %}
<style>
  li {
    font-size: 25px;
  }
  h1 {
    font-size: 50px;
    text-decoration: underline;
  }
  h3 {
    font-size: 40px;
  }
  p {
```

```
    font-size: 25px;
  }
</style>
<div class="container-fluid">
  <div
    class="card card-body"
    style="border: 1px solid black; box-shadow: 0 0 10px black"
  >
    <!-- Slideshow container -->
    <div class="slideshow-container" style="margin-bottom: 15px">
      <!-- Full-width slides/quotes -->
      <div class="mySlides">
        <div class="row">
          <div class="col-sm-4">
            <h1>This is a basic Machine and Deep Learning based WebApp.</h1>
            <p>
              These Machine Learning models and Deep Leaning models are trained
              on large datasets.
            </p>
            <br />
            <h3>Model Accuracies:</h3>
            <ul>
              <li>Diabetes Model: <strong>98.25%</strong></li>
              <li>Breast Cancer Model: <strong>98.25%</strong></li>
              <li>Heart Disease Model: <strong>85.25%</strong></li>
              <li>Kidney Disease Model: <strong>99%</strong></li>
              <li>Liver Disease Model: <strong>78%</strong></li>
            </ul>
          </div>
          <div class="align-img col-sm-8">
            <img src="static/BloodGlucoseImage.jpg" />
          </div>
        </div>
      </div>

      <div class="mySlides" style="background: #87ffff">
```

```
  <div class="row">
    <div class="col-sm-4">
      <h1>Diabetes</h1>
      <p>
        Diabetes is a disease that occurs when your blood glucose, also
        called blood sugar, is too high. Blood glucose is your main source
        of energy and comes from the food you eat. Insulin, a hormone made
        by the pancreas, helps glucose from food get into your cells to be
        used for energy. Sometimes your body doesn’t make enough—or
        any—insulin or doesn’t use insulin well. Glucose then stays in
        your blood and doesn’t reach your cells.
      </p>
      <h3>Symptoms</h3>
      <ul>
        <li>Urinating often.</li>
        <li>Feeling very thirsty.</li>
        <li>Extreme fatigue.</li>
        <li>Blurry vision.</li>
      </ul>
    </div>
    <div class="align-img col-sm-8">
      <img src="static/BloodGlucoseImage.jpg" />
    </div>
  </div>
  <p>
    <a class="btn btn-lg btn-primary" href="/diabetes" role="button"
      >Go to Diabetes Section</a
    >
  </p>
</div>

<div class="mySlides" style="background: #f598f0">
  <div class="row">
    <div class="col-sm-4">
      <h1>Breast Cancer</h1>
      <p>
```

```
          Breast cancer is cancer that forms in the cells of the breasts.
          After skin cancer, breast cancer is the most common cancer
          diagnosed in women in the United States. Breast cancer can occur
          in both men and women, but it's far more common in women.
        </p>
        <h3>Symptoms</h3>
        <ul>
          <li>
            A breast lump or thickening that feels different from the
            surrounding tissue
          </li>
          <li>Change in the size, shape or appearance of a breast</li>
          <li>Changes to the skin over the breast, such as dimpling</li>
          <li>
            Redness or pitting of the skin over your breast, like the skin
            of an orange
          </li>
        </ul>
      </div>
      <div class="align-img col-sm-8">
        <img src="static/BCAwareness.png" />
      </div>
    </div>
    <p>
      <a class="btn btn-lg btn-primary" href="/cancer" role="button"
        >Go to Breast Cancer Section</a
      >
    </p>
  </div>
  <div class="mySlides" style="background: #ff9f87">
    <div class="row">
      <div class="col-sm-4">
        <h1>Heart Disease</h1>
        <p>
          Heart disease describes a range of conditions that affect your
          heart. Diseases under the heart disease umbrella include blood
```

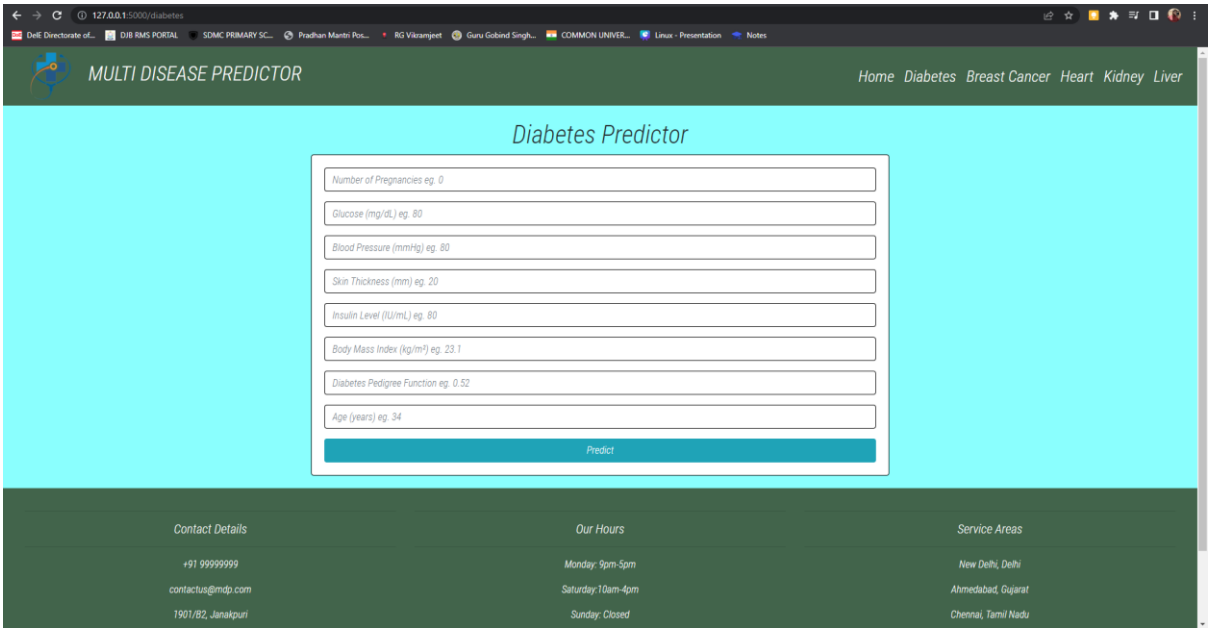
```
          vessel diseases, such as coronary artery disease; heart rhythm
          problems (arrhythmias); and heart defects you're born with
          (congenital heart defects), among others. The term "heart disease"
          is often used interchangeably with the term "cardiovascular
          disease." Cardiovascular disease generally refers to conditions
          that involve narrowed or blocked blood vessels that can lead to a
          heart attack, chest pain (angina) or stroke. Other heart
          conditions, such as those that affect your heart's muscle, valves
          or rhythm, also are considered forms of heart disease. Many forms
          of heart disease can be prevented or treated with healthy
          lifestyle choices.
        </p>
        <h3>Symptoms</h3>
        <ul>
          <li>
            Chest pain, chest tightness, chest pressure and chest discomfort
            (angina)
          </li>
          <li>Shortness of breath</li>
          <li>
            Pain, numbness, weakness or coldness in your legs or arms if the
            blood vessels in those parts of your body are narrowed
          </li>
          <li>Pain in the neck, jaw, throat, upper abdomen or back</li>
        </ul>
      </div>
      <div class="align-img col-sm-8">
        <img src="static/heartD.jpg" />
      </div>
    </div>
    <p>
      <a class="btn btn-lg btn-primary" href="/heart" role="button"
        >Go to Heart Section</a
      >
    </p>
  </div>
```

```
<div class="mySlides" style="background: #f58433">
  <div class="row">
    <div class="col-sm-4">
      <h1>Chronic kidney disease</h1>
      <p>
        Chronic kidney disease, also called chronic kidney failure,
        describes the gradual loss of kidney function. Your kidneys filter
        wastes and excess fluids from your blood, which are then excreted
        in your urine. When chronic kidney disease reaches an advanced
        stage, dangerous levels of fluid, electrolytes and wastes can
        build up in your body.
      </p>
      <h3>Symptoms</h3>
      <ul>
        <li>Nausea</li>
        <li>Vomiting</li>
        <li>Fatigue and weakness</li>
        <li>Muscle twitches and cramps</li>
      </ul>
    </div>
    <div class="align-img col-sm-8">
      <img src="static/CKD.jpg" />
    </div>
  </div>
  <p>
    <a class="btn btn-lg btn-primary" href="/kidney" role="button"
      >Go to Kidney Section</a
    >
  </p>
</div>
<div class="mySlides" style="background: #faf56b">
  <div class="row">
    <div class="col-sm-4">
      <h1>Liver disease</h1>
      <p>
        Symptoms of liver disease can vary, but they often include
```

```
          swelling of the abdomen and legs, bruising easily, changes in the
          color of your stool and urine, and jaundice, or yellowing of the
          skin and eyes. Sometimes there are no symptoms. Tests such as
          imaging tests and liver function tests can check for liver damage
          and help to diagnose liver diseases.
        </p>
        <h3>Symptoms</h3>
        <ul>
          <li>Skin and eyes that appear yellowish (jaundice)</li>
          <li>Abdominal pain and swelling.</li>
          <li>Swelling in the legs and ankles.</li>
          <li>Dark urine color.</li>
        </ul>
      </div>
      <div class="align-img col-sm-8">
        <img src="static/liverD.webp" />
      </div>
    </div>
    <p>
      <a class="btn btn-lg btn-primary" href="/liver" role="button"
        >Go to Liver Section</a
      >
    </p>
  </div>

  <!-- Next/prev buttons -->
  <a class="prev" onclick="plusSlides(-1)">&#10094;</a>
  <a class="next" onclick="plusSlides(1)">&#10095;</a>
</div>

<!-- Dots/bullets/indicators -->
<div class="dot-container" style="display:none;">
  <span class="dot" onclick="currentSlide(1)"></span>
  <span class="dot" onclick="currentSlide(2)"></span>
  <span class="dot" onclick="currentSlide(3)"></span>
  <span class="dot" onclick="currentSlide(4)"></span>
```

```
        <span class="dot" onclick="currentSlide(5)"></span>
        <span class="dot" onclick="currentSlide(6)"></span>
      </div>
      <div class="container-fluid" style="margin-top: 2%">
        <div class="row text-center padding">
          <div class="col-xs-12 col-sm-6 col-md-4">
            <i class="fa fa-bolt" aria-hidden="true"></i>
            <h3>Fast</h3>
            <p>Fast and Smooth Assessment.</p>
          </div>
          <div class="col-xs-12 col-sm-6 col-md-4">
            <i class="fa fa-lock" aria-hidden="true"></i>
            <h3>Secure</h3>
            <p>Secure Assessment Process.</p>
          </div>
          <div class="col-sm-12 col-md-4">
            <i class="fa fa-check-square" aria-hidden="true"></i>
            <h3>Reliable</h3>
            <p>Reliable and Trustworthy Process.</p>
          </div>
        </div>
        <hr class="my-4" />
      </div>
      {% endblock %}
    </div>
  </div>
```

7.2 DIABETES PAGE

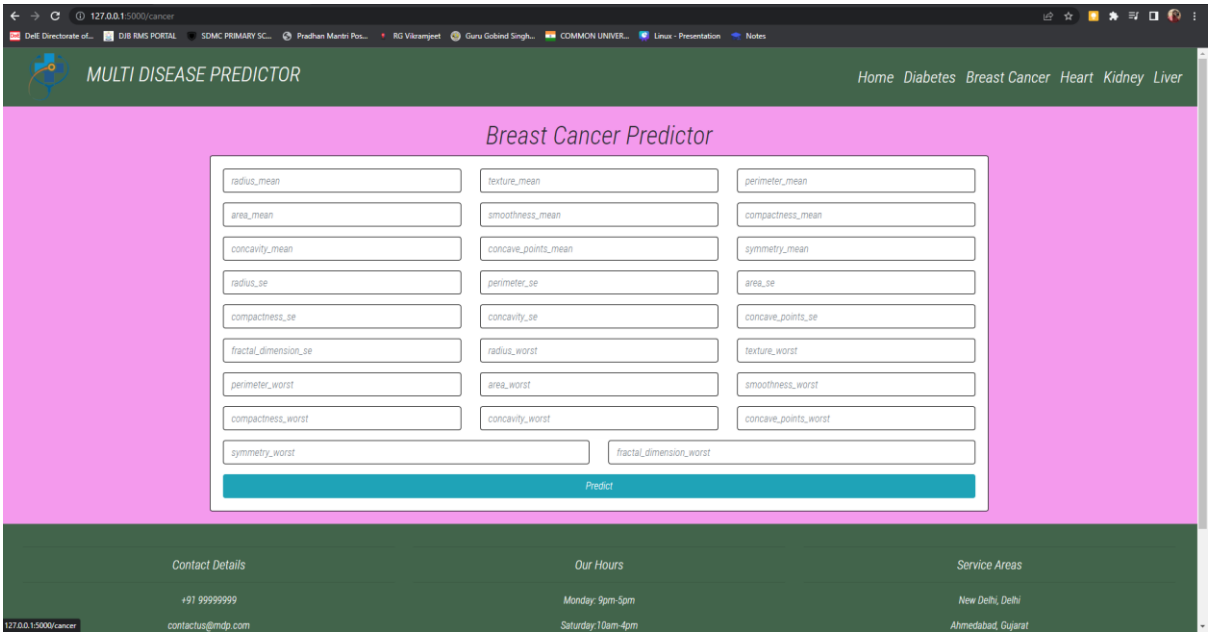


CODE:

```
{% extends 'main.html' %}
{% block content %}
{% if message %}
        <div class="alert alert-danger">{{ message }}</div>
    {% endif %}
    <style>
        body{
          background-color:#87ffff
        }
  </style>
    <div class="row">
        <div class="col-md-3"></div>
        <div class="col-md-6">
            <center><h1>Diabetes Predictor</h1></center>
            <div class="card card-body" style="border: 1px solid black;">
                <form class="form-horizontal" action="{{ url_for('predictPage') }}" method="POST">
                    <div class="form-group">
                        <input style="border: 1px solid black;" class="form-control" type="text"
name="pregnancies" placeholder="Number of Pregnancies eg. 0">
                    </div>
```

```
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="glucose" placeholder="Glucose (mg/dL) eg. 80">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="bloodpressure" placeholder="Blood Pressure (mmHg) eg. 80">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="skinthickness" placeholder="Skin Thickness (mm) eg. 20">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="insulin" placeholder="Insulin Level (IU/mL) eg. 80">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="bmi" placeholder="Body Mass Index (kg/m²) eg. 23.1">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="dpf" placeholder="Diabetes Pedigree Function eg. 0.52">
                    </div>
                    <div class="form-group">
                      <input style="border: 1px solid black;" class="form-control" type="text"
name="age" placeholder="Age (years) eg. 34">
                    </div>

                    <input type="submit" class="btn btn-info btn-block" value="Predict">
                </form>
            </div>
        </div>
        <div class="col-md-3"></div>
    </div>
{% endblock %}
```

7.3 BREAST CANCER PAGE



CODE:

```
{% extends 'main.html' %}

{% block content %}

<style>

    body{

      background-color:#f598f0

    }

</style>

<div class="row">

    <div class="col-md-2"></div>

    <div class="col-md-8">

        <center><h1>Breast Cancer Predictor</h1></center>

        <div class="card card-body" style="border: 1px solid black;">

            <form class="form-horizontal" action="{{ url_for('predictPage') }}" method="POST">

                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="radius_mean" placeholder="radius_mean">

                        </div>

                    </div>
```

```
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="texture_mean" placeholder="texture_mean">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="perimeter_mean" placeholder="perimeter_mean">
                        </div>
                    </div>
                </div>

                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="area_mean" placeholder="area_mean">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="smoothness_mean" placeholder="smoothness_mean">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="compactness_mean" placeholder="compactness_mean">
                        </div>
                    </div>
                </div>

                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
```

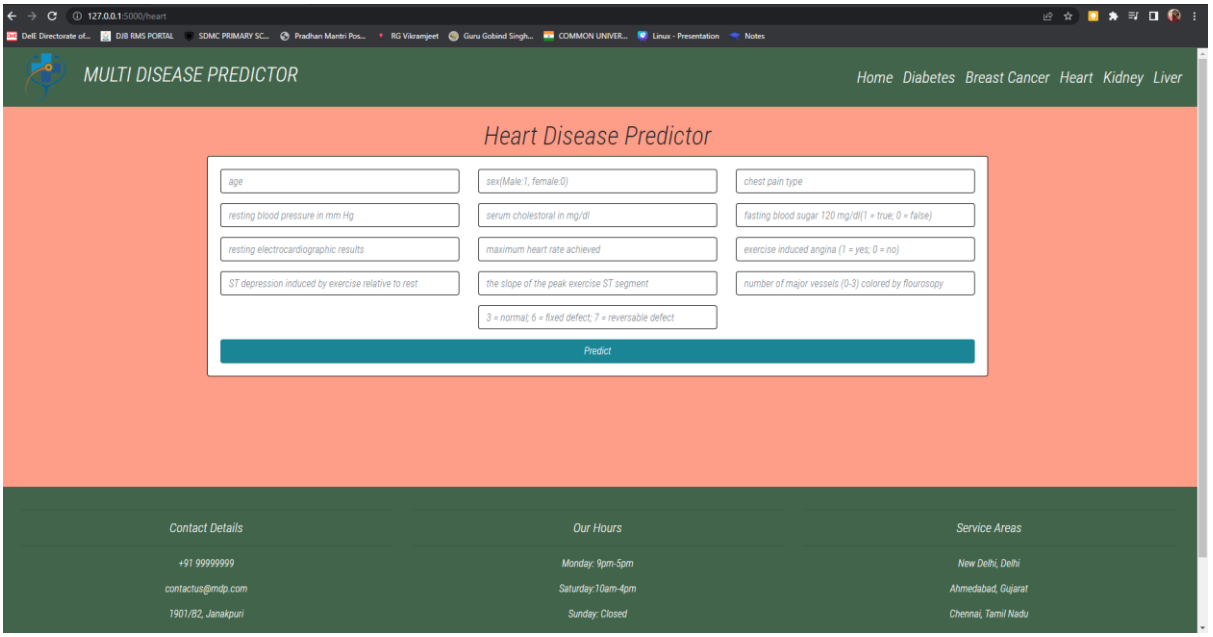
```
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concavity_mean" placeholder="concavity_mean">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concave_points_mean" placeholder="concave_points_mean">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="symmetry_mean" placeholder="symmetry_mean">

                        </div>

                    </div>

                </div>


                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="radius_se" placeholder="radius_se">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="perimeter_se" placeholder="perimeter_se">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="area_se" placeholder="area_se">

                        </div>

                    </div>

                </div>
```

```
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="compactness_se" placeholder="compactness_se">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concavity_se" placeholder="concavity_se">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concave_points_se" placeholder="concave_points_se">
                        </div>
                    </div>
                </div>


                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="fractal_dimension_se" placeholder="fractal_dimension_se">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="radius_worst" placeholder="radius_worst">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
```

```
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="texture_worst" placeholder="texture_worst">

                        </div>

                    </div>

                </div>


                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="perimeter_worst" placeholder="perimeter_worst">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="area_worst" placeholder="area_worst">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="smoothness_worst" placeholder="smoothness_worst">

                        </div>

                    </div>

                </div>


                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="compactness_worst" placeholder="compactness_worst">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concavity_worst" placeholder="concavity_worst">
```

```
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="concave_points_worst" placeholder="concave_points_worst">
                        </div>
                    </div>
                </div>


                <div class="row">
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="symmetry_worst" placeholder="symmetry_worst">
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="fractal_dimension_worst" placeholder="fractal_dimension_worst">
                        </div>
                    </div>
                </div>



                <input type="submit" class="btn btn-info btn-block" value="Predict">
            </form>
        </div>
    </div>
    <div class="col-md-2"></div>
</div>
{% endblock %}
```

7.4 HEART DISEASE PAGE



CODE:

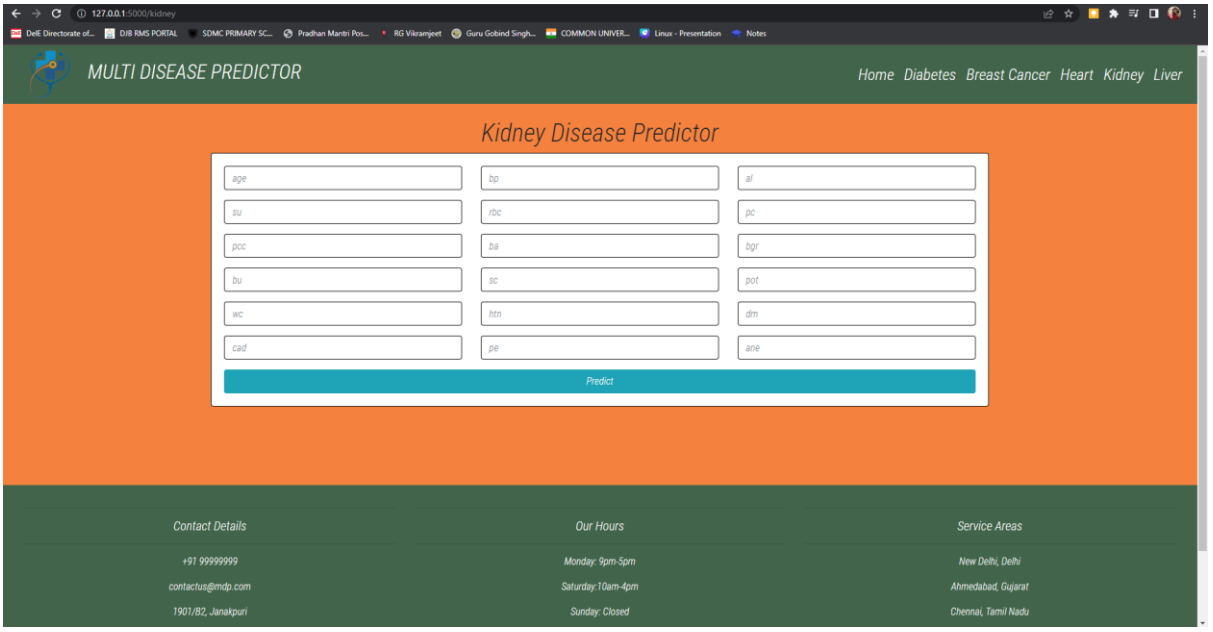
```
{% extends 'main.html' %}

{% block content %}

<style>

    body{

      background-color:#ff9f87

    }

</style>

<div class="row" style="margin-bottom: 175px;">

    <div class="col-md-2"></div>

    <div class="col-md-8">

        <center><h1>Heart Disease Predictor</h1></center>

        <div class="card card-body" style="border: 1px solid black;">

            <form class="form-horizontal" action="{{ url_for('predictPage') }}" method="POST">

                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="age" placeholder="age">

                        </div>

                    </div>
```

```
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="sex" placeholder="sex(Male:1, female:0)">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="cp" placeholder="chest pain type"></div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="trestbps" placeholder="resting blood pressure in mm Hg">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="chol" placeholder="serum cholestoral in mg/dl">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="fbs" placeholder="fasting blood sugar 120 mg/dl(1 = true; 0 = false)">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="restecg" placeholder="resting electrocardiographic results">
                        </div>
```

```
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="thalach" placeholder="maximum heart rate achieved">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="exang" placeholder="exercise induced angina (1 = yes; 0 = no)">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="oldpeak" placeholder="ST depression induced by exercise relative to rest">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="slope" placeholder="the slope of the peak exercise ST segment">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="ca" placeholder="number of major vessels (0-3) colored by flourosopy">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4"></div>
                    <div class="col-md-4">
                        <div class="form-group">
```

```
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="thal" placeholder="3 = normal; 6 = fixed defect; 7 = reversable defect">
                        </div>
                    </div>
                    <div class="col-md-4"></div>
                </div>
                <input type="submit" class="btn btn-info btn-block" value="Predict">
            </form>
        </div>
    </div>
    <div class="col-md-2"></div>
</div>


{% endblock %}
```

7.5 KIDNEY DISEASE PAGE



CODE:

```
{% extends 'main.html' %}

{% block content %}

<style>

    body{

      background-color:#f58433

    }

</style>

<div class="row"  style="margin-bottom: 125px;">

    <div class="col-md-2"></div>

    <div class="col-md-8">

        <center><h1>Kidney Disease Predictor</h1></center>

        <div class="card card-body" style="border: 1px solid black;">

            <form class="form-horizontal" action="{{ url_for('predictPage') }}" method="POST">

                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="age" placeholder="age">

                        </div>

                    </div>
```

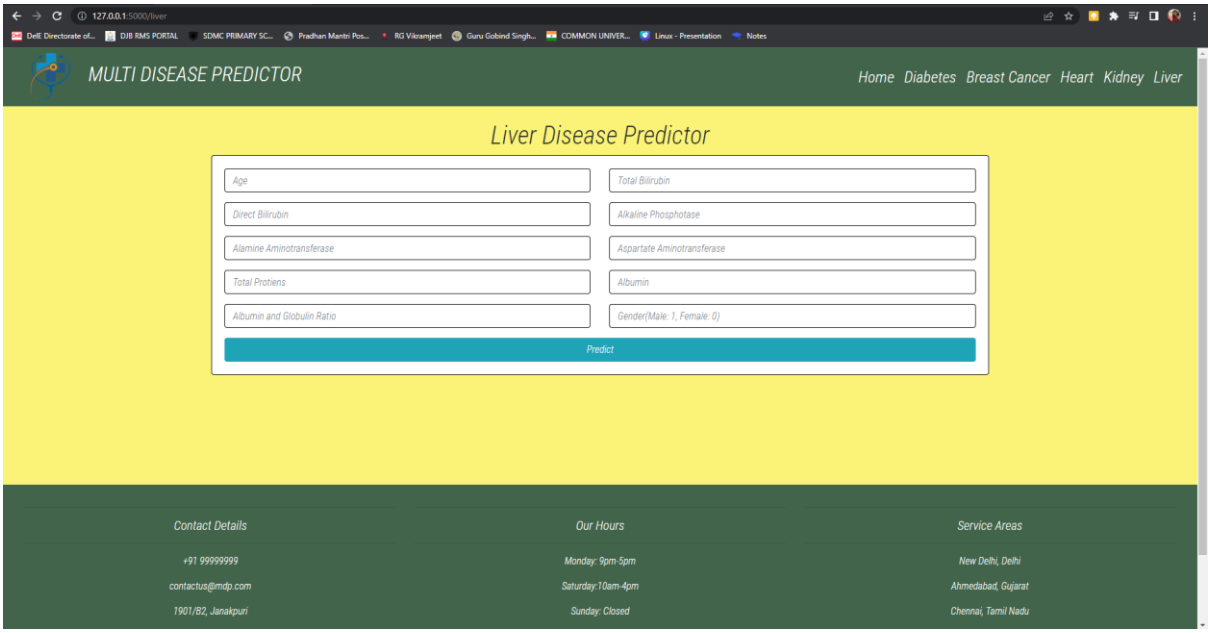
```
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="bp" placeholder="bp">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="al" placeholder="al">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="su" placeholder="su">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="rbc" placeholder="rbc">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="pc" placeholder="pc">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="pcc" placeholder="pcc">
```

```
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="ba" placeholder="ba">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="bgr" placeholder="bgr">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="bu" placeholder="bu">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="sc" placeholder="sc">
                        </div>
                    </div>
                    <div class="col-md-4">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="pot" placeholder="pot">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-4">
                        <div class="form-group">
```

```
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="wc" placeholder="wc">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="htn" placeholder="htn">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="dm" placeholder="dm">

                        </div>

                    </div>

                </div>

                <div class="row">

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="cad" placeholder="cad">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="pe" placeholder="pe">

                        </div>

                    </div>

                    <div class="col-md-4">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="ane" placeholder="ane">

                        </div>

                    </div>

                </div>

                <input type="submit" class="btn btn-info btn-block" value="Predict">
```

```
            </form>
        </div>
    </div>
    <div class="col-md-2"></div>
</div>

{% endblock %}
```

7.6 LIVER DISEASE PAGE



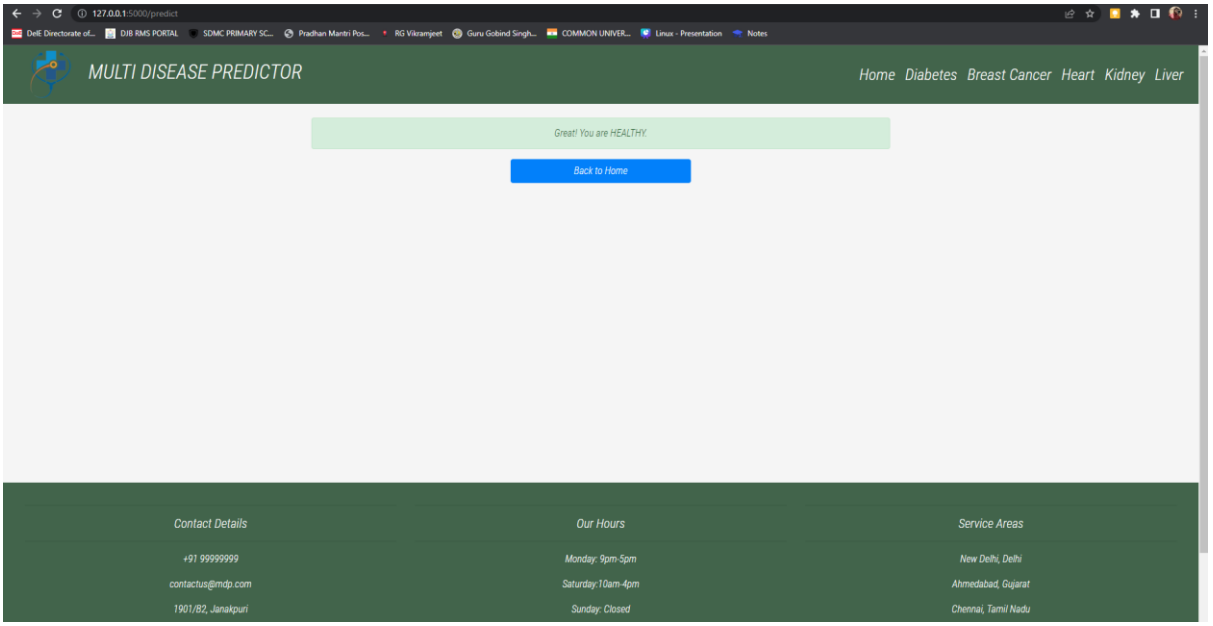
CODE:

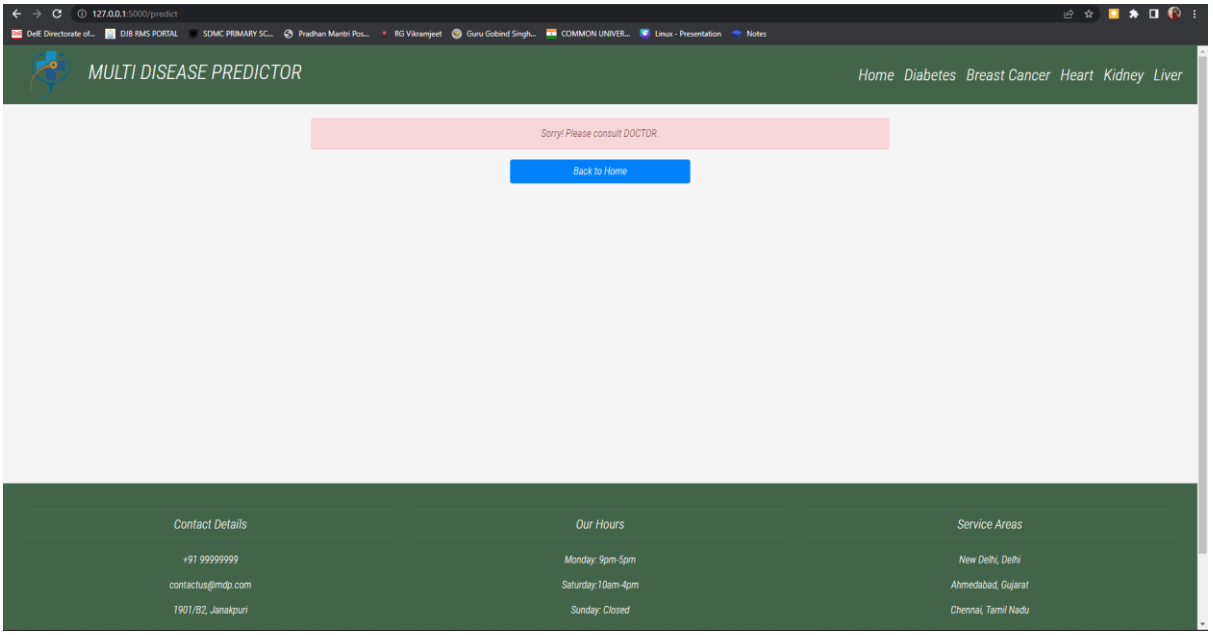
```
{% extends 'main.html' %}

{% block content %}

<style>

      body{

        background-color:#faf56b

      }

</style>


<div class="row"  style="margin-bottom: 175px;">

    <div class="col-md-2"></div>

    <div class="col-md-8">

        <center><h1>Liver Disease Predictor</h1></center>

        <div class="card card-body" style="border: 1px solid black;">

            <form class="form-horizontal" action="{{ url_for('predictPage') }}" method="POST">

                <div class="row">

                    <div class="col-md-6">

                        <div class="form-group">

                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Age" placeholder="Age">

                        </div>
```

```
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Total_Bilirubin" placeholder="Total Bilirubin">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Direct_Bilirubin" placeholder="Direct Bilirubin">
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Alkaline_Phosphotase" placeholder="Alkaline Phosphotase">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Alamine_Aminotransferase" placeholder="Alamine Aminotransferase">
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Aspartate_Aminotransferase" placeholder="Aspartate Aminotransferase">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6">
```

```
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Total_Protiens" placeholder="Total Protiens">
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Albumin" placeholder="Albumin">
                        </div>
                    </div>
                </div>
                <div class="row">
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Albumin_and_Globulin_Ratio" placeholder="Albumin and Globulin Ratio">
                        </div>
                    </div>
                    <div class="col-md-6">
                        <div class="form-group">
                            <input style="border: 1px solid black;" class="form-control"
type="text" name="Gender_Male" placeholder="Gender(Male: 1, Female: 0)">
                        </div>
                    </div>
                </div>
                <input type="submit" class="btn btn-info btn-block" value="Predict">
            </form>
        </div>
    </div>
    <div class="col-md-2"></div>
</div>


{% endblock %}
```

7.7 PREDICTION/RESULT PAGE





CODE:

```
{% extends 'main.html' %}
{% block content %}


    <div id = "assess">
        <div class="row"  style="margin-bottom: 477px;">
            <div class="col-md-3"></div>
            <div class="col-md-6">
```

```
                {% if pred == 1 %}
                    <div class="card card-body alert alert-danger"><center>Sorry! Please consult
DOCTOR.</center></div>
                {% else %}
                    <div class="card card-body alert alert-success"><center>Great! You are
HEALTHY.</center></div>
                {% endif %}
                <div class="row">
                    <div class="col-md-4"></div>
                    <div class="col-md-4"><a href="{{ url_for('home') }}" class="btn btn-block
btn-primary">Back to Home</a></div>
                    <div class="col-md-4"></div>
                </div>
            </div>
            <div class="col-md-3"></div>


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

7.8 APP.PY FILE (THROUGH WHICH THE WHOLE WEBPAGE IS CONNECTED)

CODE:

```
from flask import Flask, render_template, request, flash, redirect
import pickle
import numpy as np
from PIL import Image



app = Flask(__name__)


def predict(values, dic):
    if len(values) == 8:
        model = pickle.load(open('models/diabetes.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]
    elif len(values) == 26:
        model = pickle.load(open('models/breast_cancer.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]
    elif len(values) == 13:
        model = pickle.load(open('models/heart.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]
    elif len(values) == 18:
        model = pickle.load(open('models/kidney.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]
    elif len(values) == 10:
        model = pickle.load(open('models/liver.pkl','rb'))
        values = np.asarray(values)
        return model.predict(values.reshape(1, -1))[0]
```

```
@app.route("/")
def home():
    return render_template('home.html')


@app.route("/diabetes", methods=['GET', 'POST'])
def diabetesPage():
    return render_template('diabetes.html')


@app.route("/cancer", methods=['GET', 'POST'])
def cancerPage():
    return render_template('breast_cancer.html')


@app.route("/heart", methods=['GET', 'POST'])
def heartPage():
    return render_template('heart.html')


@app.route("/kidney", methods=['GET', 'POST'])
def kidneyPage():
    return render_template('kidney.html')


@app.route("/liver", methods=['GET', 'POST'])
def liverPage():
    return render_template('liver.html')


@app.route("/predict", methods = ['POST', 'GET'])
def predictPage():
    try:
        if request.method == 'POST':
            to_predict_dict = request.form.to_dict()
            to_predict_list = list(map(float, list(to_predict_dict.values())))
            pred = predict(to_predict_list, to_predict_dict)
    except:
        message = "Please enter valid Data"
        return render_template("home.html", message = message)

    return render_template('predict.html', pred = pred)
```

```
if __name__ == '__main__':
        app.run(debug = True)
```

7.8 JUPYTER NOTEBOOKS FOR CREATION OF MODEL

7.8.1 BREAST CANCER NOTEBOOK

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('data.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	...	tex
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	...	
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	...	
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	...	
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	...	
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	...	

5 rows × 33 columns

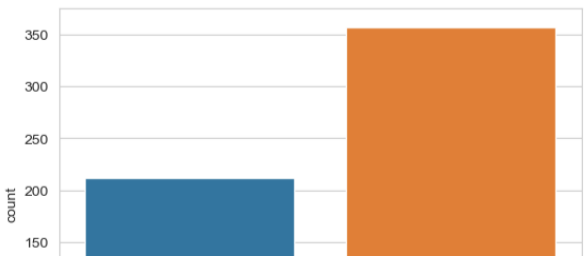
In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype
---  ------                   --------------  -----
 0   id                       569 non-null    int64
 1   diagnosis                569 non-null    object
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [5]:
```
import seaborn as sns
```

In [6]:
```
sns.set_style('whitegrid')
sns.countplot(x = 'diagnosis', data = data)
```

Out[6]:
```
<Axes: xlabel='diagnosis', ylabel='count'>
```

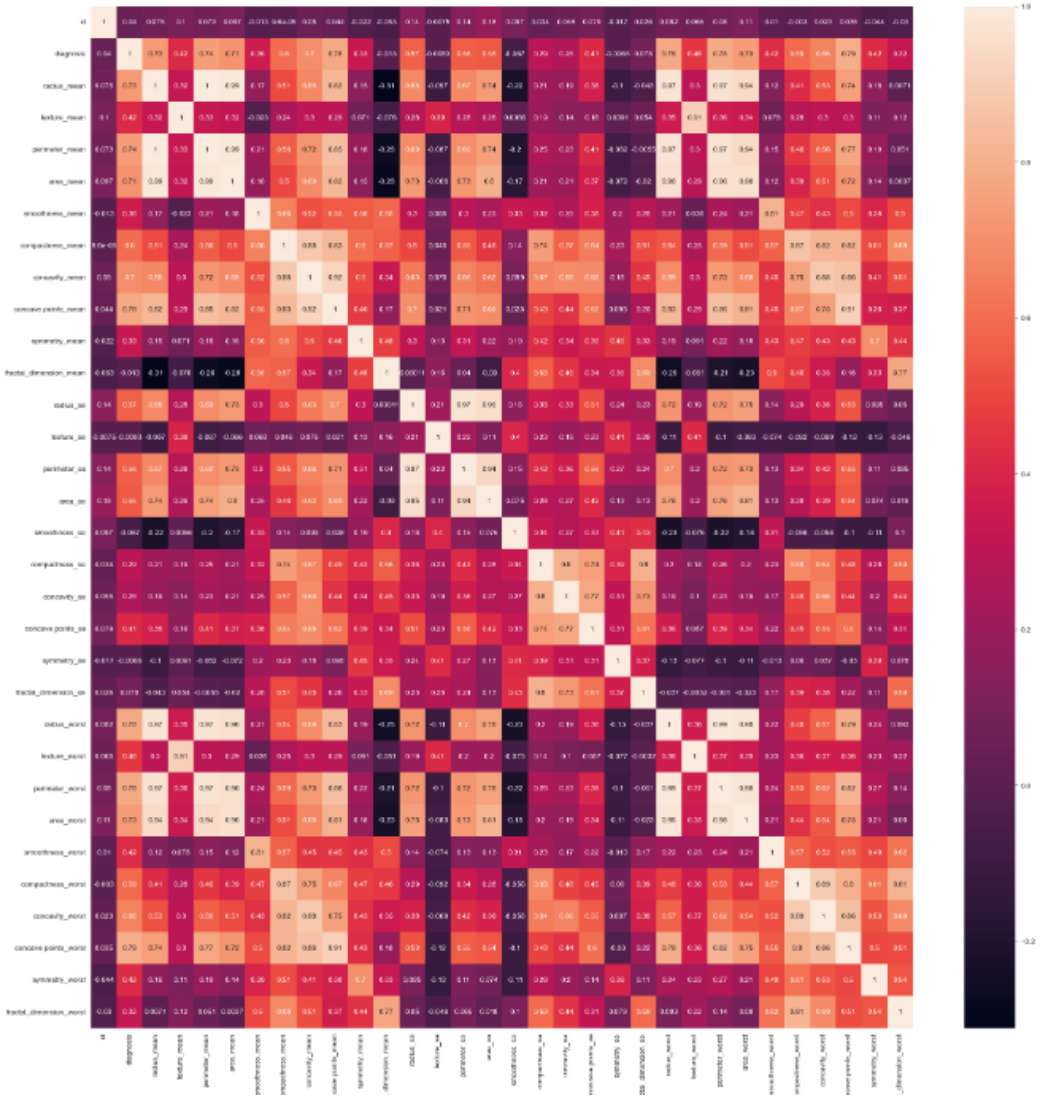


In [7]:
```
dataset = data
dataset['diagnosis'].replace(['M','B'], [1,0], inplace = True)
```

In [8]:
```
dataset.drop('Unnamed: 32',axis = 1, inplace = True)
```

In [9]:
```
corr = dataset.corr()
plt.figure(figsize = (25,25))
sns.heatmap(corr, annot = True)
```

Out[9]: <Axes: >



In [10]:
```
dataset.corr()
```

Out[10]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean
id	1.000000	0.039769	0.074626	0.099770	0.073159	0.096893	-0.012968	0.000096	0.050080
diagnosis	0.039769	1.000000	0.730029	0.415185	0.742636	0.708984	0.358560	0.596534	0.696360
radius_mean	0.074626	0.730029	1.000000	0.323782	0.997855	0.987357	0.170581	0.506124	0.676764
texture_mean	0.099770	0.415185	0.323782	1.000000	0.329533	0.321086	-0.023389	0.236702	0.302418
perimeter_mean	0.073159	0.742636	0.997855	0.329533	1.000000	0.986507	0.207278	0.556936	0.716136
area_mean	0.096893	0.708984	0.987357	0.321086	0.986507	1.000000	0.177028	0.498502	0.685983
smoothness_mean	-0.012968	0.358560	0.170581	-0.023389	0.207278	0.177028	1.000000	0.659123	0.521984
compactness_mean	0.000096	0.596534	0.506124	0.236702	0.556936	0.498502	0.659123	1.000000	0.883121
concavity_mean	0.050080	0.696360	0.676764	0.302418	0.716136	0.685983	0.521984	0.883121	1.000000
concave points_mean	0.044158	0.776614	0.822529	0.293464	0.850977	0.823269	0.553695	0.831135	0.921391
symmetry_mean	-0.022114	0.330499	0.147741	0.071401	0.183027	0.151293	0.557775	0.602641	0.500667
fractal_dimension_mean	-0.052511	-0.012838	-0.311631	-0.076437	-0.261477	-0.283110	0.584792	0.565369	0.336783
radius_se	0.143048	0.567134	0.679090	0.275869	0.691765	0.732562	0.301467	0.497473	0.631925
texture_se	-0.007526	-0.008303	-0.097317	0.386358	-0.086761	-0.066280	0.068406	0.046205	0.076218
perimeter_se	0.137331	0.556141	0.674172	0.281673	0.693135	0.726628	0.296092	0.548905	0.660391
area_se	0.177742	0.548236	0.735864	0.259845	0.744983	0.800086	0.246552	0.455653	0.617427
smoothness_se	0.096781	-0.067016	-0.222600	0.006614	-0.202694	-0.166777	0.332375	0.135299	0.098564
compactness_se	0.033961	0.292999	0.206000	0.191975	0.250744	0.212583	0.318943	0.738722	0.670279
concavity_se	0.055239	0.253730	0.194204	0.143293	0.228082	0.207660	0.248396	0.570517	0.691270
concave points_se	0.078768	0.408042	0.376169	0.163851	0.407217	0.372320	0.380676	0.642262	0.683260
symmetry_se	-0.017306	-0.006522	-0.104321	0.009127	-0.081629	-0.072497	0.200774	0.229977	0.178009
fractal_dimension_se	0.025725	0.077972	-0.042641	0.054458	-0.005523	-0.019887	0.283607	0.507318	0.449301
radius_worst	0.082405	0.776454	0.969539	0.352573	0.969476	0.962746	0.213120	0.535315	0.688236
texture_worst	0.064720	0.456903	0.297008	0.912045	0.303038	0.287489	0.036072	0.248133	0.299879
perimeter_worst	0.079986	0.782914	0.965137	0.358040	0.970387	0.959120	0.238853	0.590210	0.729565
area_worst	0.107187	0.733825	0.941082	0.343546	0.941550	0.959213	0.206718	0.509604	0.675987
smoothness_worst	0.010338	0.421465	0.119616	0.077503	0.150549	0.123523	0.805324	0.565541	0.448822
compactness_worst	-0.002968	0.590998	0.413463	0.277830	0.455774	0.390410	0.472468	0.865809	0.754968
concavity_worst	0.023203	0.659610	0.526911	0.301025	0.563879	0.512606	0.434926	0.816275	0.884103
concave points_worst	0.035174	0.793566	0.744214	0.295316	0.771241	0.722017	0.503053	0.815573	0.861323
symmetry_worst	-0.044224	0.416294	0.163953	0.105008	0.189115	0.143570	0.394309	0.510223	0.409464
fractal_dimension_worst	-0.029866	0.323872	0.007066	0.119205	0.051019	0.003738	0.499316	0.687382	0.514930

32 rows × 32 columns

In [11]:
```
dataset.drop(['id','symmetry_se','smoothness_se','texture_se','fractal_dimension_mean'], axis = 1, inplace = True)
dataset.head()
```

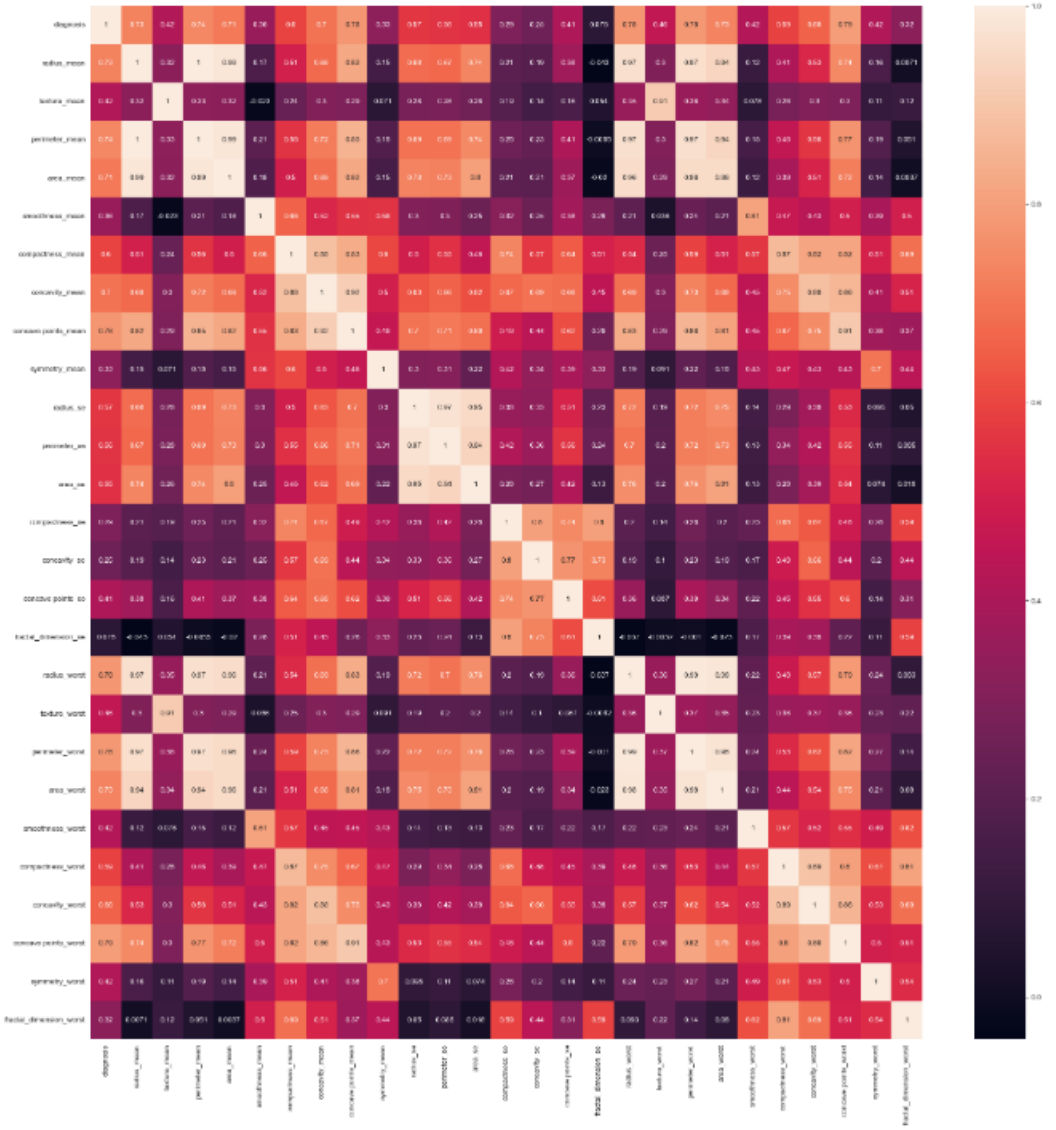
Out[11]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	compactness_mean	concavity_mean	concave points_mean	symmetry_mean
0	1	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419
1	1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812
2	1	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069
3	1	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597
4	1	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809

5 rows × 27 columns

In [12]:
```
plt.figure(figsize = (25,25))
sns.heatmap(dataset.corr(), annot = True)
```

Out[12]: <Axes: >



In [13]:
```
X = dataset.drop('diagnosis', axis = 1)
y = dataset['diagnosis']
```

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [13]:
```
X = dataset.drop('diagnosis', axis = 1)
y = dataset['diagnosis']
```

In [14]:
```
from sklearn.model_selection import train_test_split
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [16]:
```
print("Train Set: ", X_train.shape, y_train.shape)
print("Test Set: ", X_test.shape, y_test.shape)
```

```
Train Set:  (455, 26) (455,)
Test Set:  (114, 26) (114,)
```

In [17]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
```

Out[17]:
```
RandomForestClassifier(n_estimators=20)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [18]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [19]:
```
confusion_matrix(y_test, model.predict(X_test))
```

Out[19]:
```
array([[69,  2],
       [ 3, 40]], dtype=int64)
```

In [20]:
```
print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test))*100,2)}")
```

```
Accuracy is 95.61
```

In [23]:
```
classifier = RandomForestClassifier(n_jobs = -1)
```

In [24]:
```
import pickle
pickle.dump(classifier, open('cancer.pkl', 'wb'))
```

In [ ]:

7.8.2 DIABETES NOTEBOOK

In [1]:
```
import numpy as mp
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [13]:
```
data = pd.read_csv('diabetes.csv')
data.head()
```

Out[13]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

In [16]:
```
data.shape
```

Out[16]:
```
(768, 9)
```

In [15]:
```
data.isnull().sum()
```

Out[15]:
```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

In [17]:
```
data.corr()
```

Out[17]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
Pregnancies	1.000000	0.129459	0.141282	-0.081672	-0.073535	0.017683	-0.033523	0.544341	0.221898
Glucose	0.129459	1.000000	0.152590	0.057328	0.331357	0.221071	0.137337	0.263514	0.466581
BloodPressure	0.141282	0.152590	1.000000	0.207371	0.088933	0.281805	0.041265	0.239528	0.065068
SkinThickness	-0.081672	0.057328	0.207371	1.000000	0.436783	0.392573	0.183928	-0.113970	0.074752
Insulin	-0.073535	0.331357	0.088933	0.436783	1.000000	0.197859	0.185071	-0.042163	0.130548
BMI	0.017683	0.221071	0.281805	0.392573	0.197859	1.000000	0.140647	0.036242	0.292695
DiabetesPedigreeFunction	-0.033523	0.137337	0.041265	0.183928	0.185071	0.140647	1.000000	0.033561	0.173844
Age	0.544341	0.263514	0.239528	-0.113970	-0.042163	0.036242	0.033561	1.000000	0.238356
Outcome	0.221898	0.466581	0.065068	0.074752	0.130548	0.292695	0.173844	0.238356	1.000000

In [18]:
```
import seaborn as sns
```

In [19]:
```
plt.figure(figsize=(10,10))
```

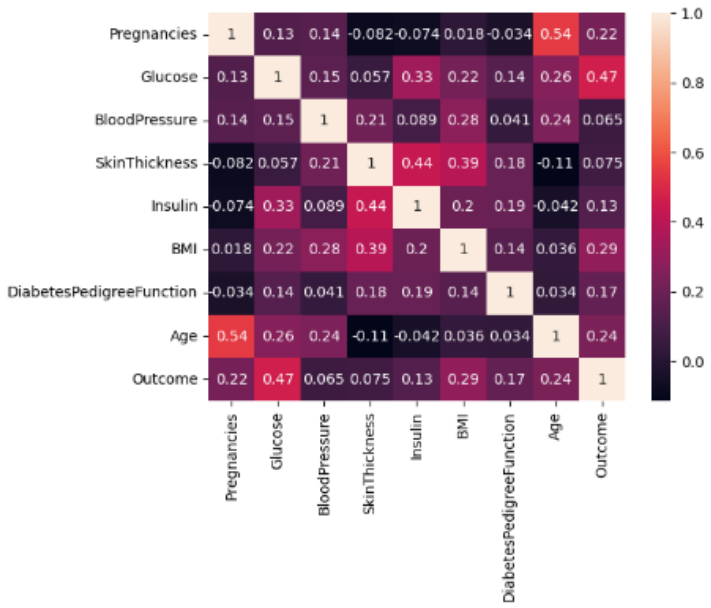
Out[19]:
```
<Figure size 1000x1000 with 0 Axes>
```

```
<Figure size 1000x1000 with 0 Axes>
```

In [20]:
```
sns.heatmap(data.corr(), annot = True)
```

Out[20]:
```
<Axes: >
```

Out[20]: <Axes: >



In [21]:
```
x = data.iloc[:,:-1]
y = data['Outcome']
```

In [22]:
```
x.shape
```

Out[22]: (768, 8)

In [23]:
```
y.shape
```

Out[23]: (768,)

In [24]:
```
from sklearn.model_selection import train_test_split
```

In [25]:
```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.2, random_state = 10)
```

In [26]:
```
print("Train Set: ", x_train.shape, y_train.shape)
```

```
Train Set:  (614, 8) (614,)
```

In [27]:
```
print("Test Set: ", x_test.shape, y_test.shape)
```

```
Test Set:  (154, 8) (154,)
```

In [29]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(x_train, y_train)
```

Out[29]:
```
RandomForestClassifier(n_estimators=20)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [30]:
```
from sklearn.metrics import accuracy_score
```

In [31]:
```
print(accuracy_score(y_test, model.predict(x_test))*100)
```

```
79.22077922077922
```

In [32]:
```
import pickle
pickle.dump(model, open("diabetes.pkl",'wb'))
```

7.8.3 HEART DISEASE NOTEBOOK

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('heart.csv')
data.head()
```

Out[2]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [3]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

In [4]:
```
data.isnull().sum()
```

Out[4]:
```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

In [5]:
```
data.describe()
```

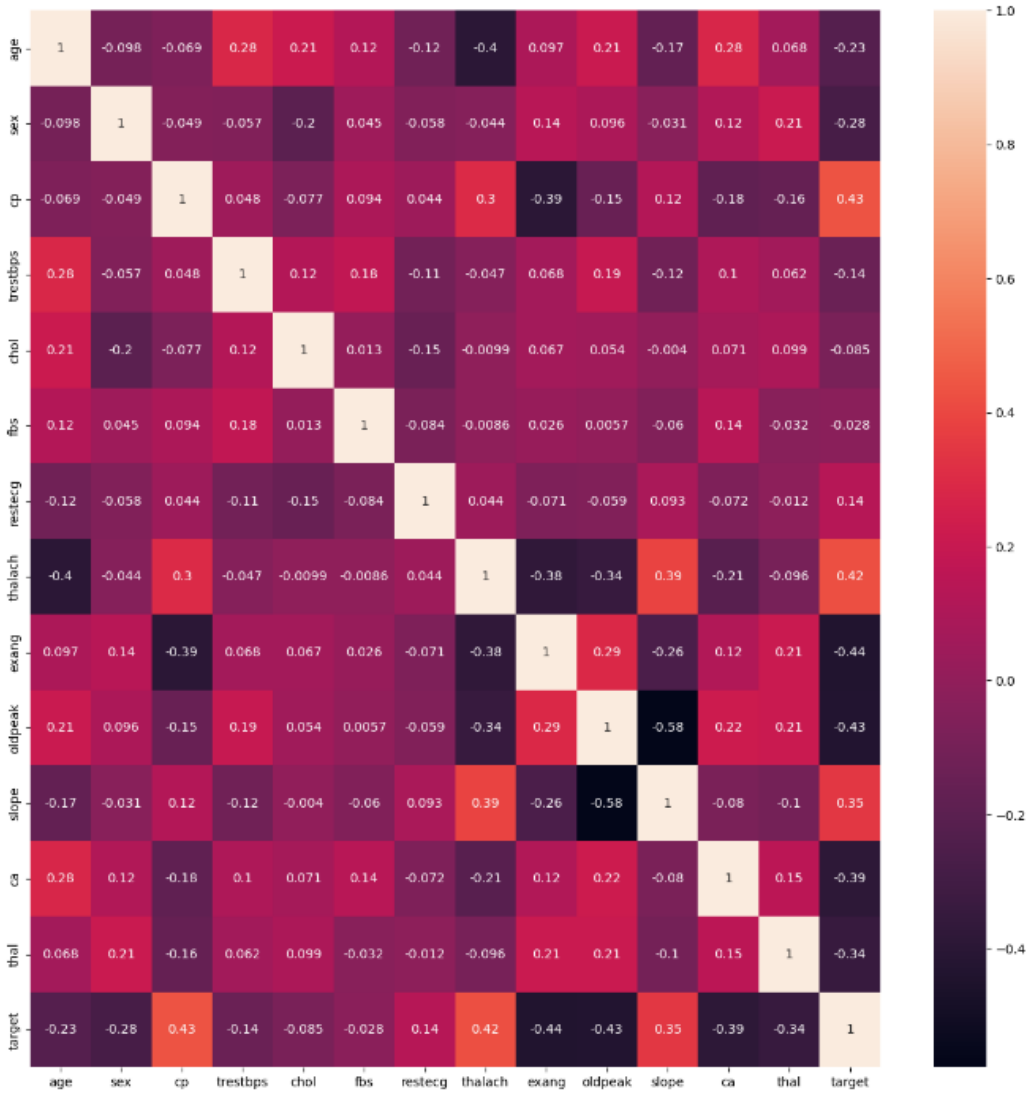
Out[5]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.0
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.3
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.6
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.0

max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.0(

In [6]:
```
import seaborn as sns

corr = data.corr()

plt.figure(figsize = (15,15))
sns.heatmap(corr, annot = True)
```

Out[6]: <Axes: >



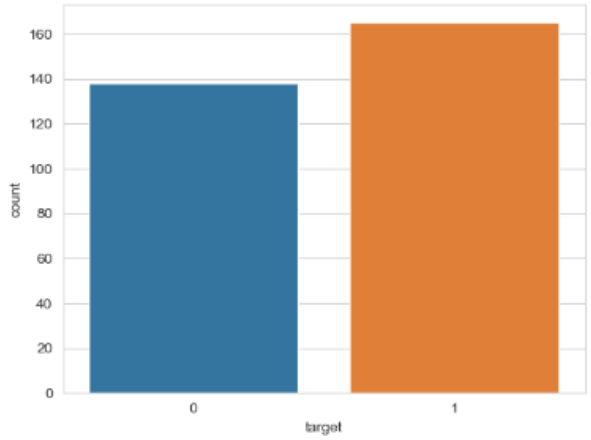
In [7]:
```
corr
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.398522	0.096801	0.210013	-0.168814	0.276326	0.068001	-0.225439
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.044020	0.141664	0.096093	-0.030711	0.118261	0.210041	-0.280937
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.295762	-0.394280	-0.149230	0.119717	-0.181053	-0.161736	0.433798
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.046698	0.067616	0.193216	-0.121475	0.101389	0.062210	-0.144931
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.009940	0.067023	0.053952	-0.004038	0.070511	0.098803	-0.085239
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.008567	0.025665	0.005747	-0.059894	0.137979	-0.032019	-0.028046
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.044123	-0.070733	-0.058770	0.093045	-0.072042	-0.011981	0.137230
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.000000	-0.378812	-0.344187	0.386784	-0.213177	-0.096439	0.421741
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.378812	1.000000	0.288223	-0.257748	0.115739	0.206754	-0.436757
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.344187	0.288223	1.000000	-0.577537	0.222682	0.210244	-0.430696
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.386784	-0.257748	-0.577537	1.000000	-0.080155	-0.104764	0.345877
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.213177	0.115739	0.222682	-0.080155	1.000000	0.151832	-0.391724
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.096439	0.206754	0.210244	-0.104764	0.151832	1.000000	-0.344029
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.421741	-0.436757	-0.430696	0.345877	-0.391724	-0.344029	1.000000

In [8]:
```
sns.set_style('whitegrid')
sns.countplot(x = 'target', data = data)
```

Out[8]:
```
<Axes: xlabel='target', ylabel='count'>
```



In [9]:
```
dataset = data.copy()
dataset.head()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [9]:
```
dataset = data.copy()
dataset.head()
```

Out[9]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [10]:
```
X = dataset.drop(['target'], axis = 1)
y = dataset['target']
X.columns
```

Out[10]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal'],
      dtype='object')
```

In [11]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
```

Out[12]:
```
RandomForestClassifier(n_estimators=20)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [13]:
```
pred = model.predict(X_test)
pred[:10]
```

Out[13]:
```
array([0, 1, 1, 0, 1, 1, 1, 0, 0, 0], dtype=int64)
```

In [14]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test, pred)
```

Out[14]:
```
array([[24,  5],
       [ 5, 27]], dtype=int64)
```

In [15]:
```
from sklearn.metrics import accuracy_score
print(f"Accuracy of model is {round(accuracy_score(y_test, pred)*100, 2)}%")
```

```
Accuracy of model is 83.61%
```

In [16]:
```
classifier = RandomForestClassifier(n_jobs = -1)
```

In [17]:
```
import pickle
pickle.dump(classifier, open('heart.pkl', 'wb'))
```

In [ ]:
```

```

7.8.4 KIDNEY DISEASE NOTEBOOK

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('kidney_disease.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	id	age	bp	sg	al	su	rbc	pc	pcc	ba	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 26 columns

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 26 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              400 non-null    int64
 1   age             391 non-null    float64
 2   bp              388 non-null    float64
 3   sg              353 non-null    float64
 4   al              354 non-null    float64
 5   su              351 non-null    float64
 6   rbc             248 non-null    object
 7   pc              335 non-null    object
 8   pcc             396 non-null    object
 9   ba              396 non-null    object
 10  bgr             356 non-null    float64
 11  bu              381 non-null    float64
 12  sc              383 non-null    float64
 13  sod             313 non-null    float64
 14  pot             312 non-null    float64
 15  hemo            348 non-null    float64
 16  pcv             330 non-null    object
 17  wc              295 non-null    object
 18  rc              270 non-null    object
 19  htn             398 non-null    object
 20  dm              398 non-null    object
 21  cad             398 non-null    object
 22  appet           399 non-null    object
 23  pe              399 non-null    object
 24  ane             399 non-null    object
 25  classification  400 non-null    object
dtypes: float64(11), int64(1), object(14)
memory usage: 81.4+ KB
```

In [5]:
```
data.classification.unique()
```

Out[5]:
```
array(['ckd', 'ckd\t', 'notckd'], dtype=object)
```

In [6]:
```
data.classification=data.classification.replace("ckd\t","ckd")
data.classification.unique()
```

Out[6]:
```
array(['ckd', 'notckd'], dtype=object)
```

In [7]:
```
data.drop('id', axis = 1, inplace = True)
data.head()
```

Out[7]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	ckd
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	ckd
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	ckd
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	ckd
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	ckd

5 rows × 25 columns

In [8]:
```
data['classification'] = data['classification'].replace(['ckd','notckd'], [1,0])
data.head()
```

Out[8]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	80.0	1.020	1.0	0.0	NaN	normal	notpresent	notpresent	121.0	...	44	7800	5.2	yes	yes	no	good	no	no	1
1	7.0	50.0	1.020	4.0	0.0	NaN	normal	notpresent	notpresent	NaN	...	38	6000	NaN	no	no	no	good	no	no	1
2	62.0	80.0	1.010	2.0	3.0	normal	normal	notpresent	notpresent	423.0	...	31	7500	NaN	no	yes	no	poor	no	yes	1
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	1
4	51.0	80.0	1.010	2.0	0.0	normal	normal	notpresent	notpresent	106.0	...	35	7300	4.6	no	no	no	good	no	no	1

5 rows × 25 columns

In [9]:
```
data.isnull().sum()
```

Out[9]:
```
age                9
bp                12
sg                47
al                46
su                49
rbc              152
pc                65
pcc                4
ba                 4
bgr               44
bu                19
sc                17
sod               87
pot               88
hemo              52
pcv               70
wc               105
rc               130
htn                2
dm                 2
cad                2
appet              1
pe                 1
ane                1
classification     0
dtype: int64
```

In [10]:
```
df = data.dropna(axis = 0)
print(f"Before dropping all NaN values: {data.shape}")
print(f"After dropping all NaN values: {df.shape}")
```

```
Before dropping all NaN values: (400, 25)
After dropping all NaN values: (158, 25)
```

In [11]:
```
df.head()
```

Out[11]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
3	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	1
5	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	70.0	...	29	12100	3.7	yes	yes	no	poor	no	yes	1
11	63.0	70.0	1.010	3.0	0.0	abnormal	abnormal	present	notpresent	380.0	...	32	4500	3.8	yes	yes	no	poor	yes	no	1
14	68.0	80.0	1.010	3.0	2.0	normal	abnormal	present	present	157.0	...	16	11000	2.6	yes	yes	yes	poor	yes	no	1

In [12]:
```
df.index = range(0,len(df),1)
df.head()
```

Out[12]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	70.0	1.005	4.0	0.0	normal	abnormal	present	notpresent	117.0	...	32	6700	3.9	yes	no	no	poor	yes	yes	1
1	53.0	90.0	1.020	2.0	0.0	abnormal	abnormal	present	notpresent	70.0	...	29	12100	3.7	yes	yes	no	poor	no	yes	1
2	63.0	70.0	1.010	3.0	0.0	abnormal	abnormal	present	notpresent	380.0	...	32	4500	3.8	yes	yes	no	poor	yes	no	1
3	68.0	80.0	1.010	3.0	2.0	normal	abnormal	present	present	157.0	...	16	11000	2.6	yes	yes	yes	poor	yes	no	1
4	61.0	80.0	1.015	2.0	0.0	abnormal	abnormal	notpresent	notpresent	173.0	...	24	9200	3.2	yes	yes	yes	poor	yes	yes	1

5 rows × 25 columns

In [13]:
```
for i in df['wc']:
    print(i)
```

```
6700
12100
4500
11000
9200
6900
9600
18900
7200
14600
6400
	6200
3800
9800
12500
5600
7000
15200
5000
```

In [14]:
```
df['wc']=df['wc'].replace(["\t6200","\t8400"],[6200,8400])
```

```
C:\Users\shikh\AppData\Local\Temp\ipykernel_20976\179227785.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['wc']=df['wc'].replace(["\t6200","\t8400"],[6200,8400])
```

In [15]:
```
for i in df['wc']:
    print(i)
```

```
6700
12100
4500
11000
9200
6900
9600
18900
7200
14600
6400
6200
3800
9800
12500
5600
7000
15200
5000
```

In [16]:
```
df.info()
```

In [16]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 25 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             158 non-null    float64
 1   bp              158 non-null    float64
 2   sg              158 non-null    float64
 3   al              158 non-null    float64
 4   su              158 non-null    float64
 5   rbc             158 non-null    object 
 6   pc              158 non-null    object 
 7   pcc             158 non-null    object 
 8   ba              158 non-null    object 
 9   bgr             158 non-null    float64
 10  bu              158 non-null    float64
 11  sc              158 non-null    float64
 12  sod             158 non-null    float64
 13  pot             158 non-null    float64
 14  hemo            158 non-null    float64
 15  pcv             158 non-null    object 
 16  wc              158 non-null    object 
 17  rc              158 non-null    object 
 18  htn             158 non-null    object 
 19  dm              158 non-null    object 
 20  cad             158 non-null    object 
 21  appet           158 non-null    object 
 22  pe              158 non-null    object 
 23  ane             158 non-null    object 
 24  classification  158 non-null    int64  
dtypes: float64(11), int64(1), object(13)
memory usage: 31.0+ KB
```

In [17]:
```
df['pcv']=df['pcv'].astype(int)
df['wc']=df['wc'].astype(int)
df['rc']=df['rc'].astype(float)
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 25 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             158 non-null    float64
 1   bp              158 non-null    float64
 2   sg              158 non-null    float64
 3   al              158 non-null    float64
 4   su              158 non-null    float64
 5   rbc             158 non-null    object 
 6   pc              158 non-null    object 
 7   pcc             158 non-null    object 
 8   ba              158 non-null    object 
 9   bgr             158 non-null    float64
 10  bu              158 non-null    float64
 11  sc              158 non-null    float64
 12  sod             158 non-null    float64
 13  pot             158 non-null    float64
 14  hemo            158 non-null    float64
 15  pcv             158 non-null    int32  
 16  wc              158 non-null    int32  
 17  rc              158 non-null    float64
 18  htn             158 non-null    object 
 19  dm              158 non-null    object 
 20  cad             158 non-null    object 
 21  appet           158 non-null    object 
 22  pe              158 non-null    object 
 23  ane             158 non-null    object 
 24  classification  158 non-null    int64  
dtypes: float64(12), int32(2), int64(1), object(10)
memory usage: 29.8+ KB
```

```
C:\Users\shikh\AppData\Local\Temp\ipykernel_20976\4219893708.py:1: SettingWithCopyWarning:
```

In [18]:
```
object_dtypes = df.select_dtypes(include = 'object')
object_dtypes.head()
```

Out[18]:

	rbc	pc	pcc	ba	htn	dm	cad	appet	pe	ane
0	normal	abnormal	present	notpresent	yes	no	no	poor	yes	yes
1	abnormal	abnormal	present	notpresent	yes	yes	no	poor	no	yes
2	abnormal	abnormal	present	notpresent	yes	yes	no	poor	yes	no
3	normal	abnormal	present	present	yes	yes	yes	poor	yes	no
4	abnormal	abnormal	notpresent	notpresent	yes	yes	yes	poor	yes	yes

In [19]:
```
dictonary = {
        "rbc": {
        "abnormal":1,
        "normal": 0,
    },
        "pc":{
        "abnormal":1,
        "normal": 0,
    },
        "pcc":{
        "present":1,
        "notpresent":0,
    },
        "ba":{
        "notpresent":0,
        "present": 1,
    },
        "htn":{
        "yes":1,
        "no": 0,
    },
        "dm":{
        "yes":1,
        "no":0,
    },
        "cad":{
        "yes":1,
        "no": 0,
    },
        "appet":{
        "good":1,
        "poor": 0,
    },
        "pe":{
        "yes":1,
        "no":0,
    },
        "ane":{
        "yes":1,
        "no":0,
    }
}
```

In [20]:
```
df=df.replace(dictonary)
df.head()
```

Out[20]:

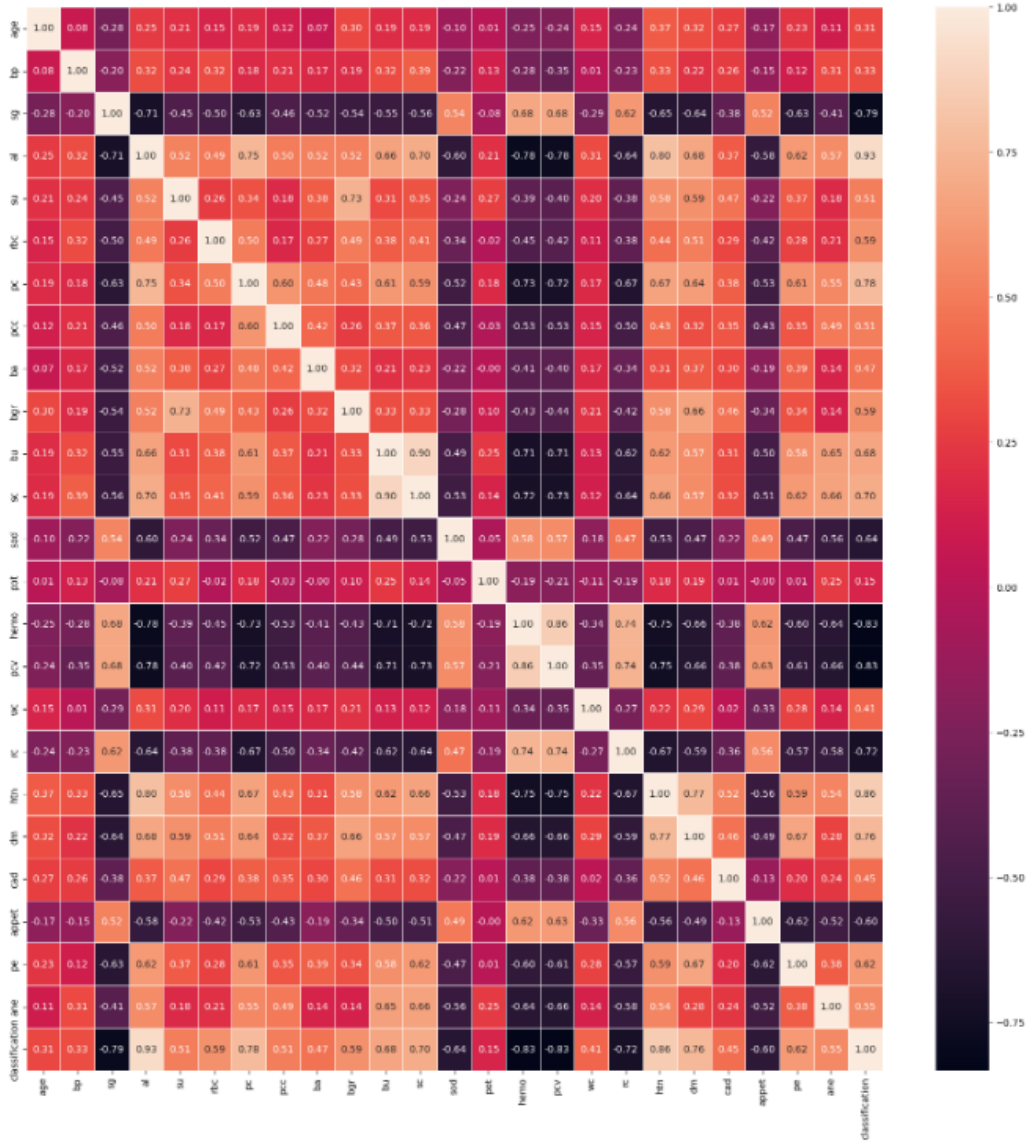
	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	htn	dm	cad	appet	pe	ane	classification
0	48.0	70.0	1.005	4.0	0.0	0	1	1	0	117.0	...	32	6700	3.9	1	0	0	0	1	1	1
1	53.0	90.0	1.020	2.0	0.0	1	1	1	0	70.0	...	29	12100	3.7	1	1	0	0	0	1	1
2	63.0	70.0	1.010	3.0	0.0	1	1	1	0	380.0	...	32	4500	3.8	1	1	0	0	1	0	1
3	68.0	80.0	1.010	3.0	2.0	0	1	1	1	157.0	...	16	11000	2.6	1	1	1	0	1	0	1
4	61.0	80.0	1.015	2.0	0.0	1	1	0	0	173.0	...	24	9200	3.2	1	1	1	0	1	1	1

5 rows × 25 columns

In [21]:
```
import seaborn as sns
plt.figure(figsize = (20,20))
```

In [21]:
```
import seaborn as sns
plt.figure(figsize = (20,20))
sns.heatmap(df.corr(), annot = True, fmt=".2f",linewidths=0.5)
```

Out[21]: <Axes: >



In [22]:
```
df.corr()
```

Out[22]:

	age	bp	sg	al	su	rbc	pc	pcc	ba	bgr	...	pcv	wc	rc	
age	1.000000	0.079712	-0.277303	0.253380	0.207711	0.147971	0.188907	0.124032	0.068353	0.301915	...	-0.235116	0.153132	-0.242235	0.
bp	0.079712	1.000000	-0.196897	0.322507	0.243828	0.316670	0.179834	0.206507	0.174555	0.190113	...	-0.353504	0.008274	-0.228533	0.
sg	-0.277303	-0.196897	1.000000	-0.712331	-0.448477	-0.500494	-0.630323	-0.460050	-0.516392	-0.544781	...	0.678472	-0.288930	0.619092	-0.
al	0.253380	0.322507	-0.712331	1.000000	0.521448	0.489941	0.752956	0.503341	0.516104	0.518123	...	-0.775528	0.314574	-0.640099	0.
su	0.207711	0.243828	-0.448477	0.521448	1.000000	0.256568	0.335901	0.177327	0.381929	0.730050	...	-0.404821	0.201000	-0.377726	0.
rbc	0.147971	0.316670	-0.500494	0.489941	0.256568	1.000000	0.498959	0.168592	0.273177	0.493857	...	-0.422537	0.108857	-0.379378	0.
pc	0.188907	0.179834	-0.630323	0.752956	0.335901	0.498959	1.000000	0.600092	0.481227	0.430646	...	-0.718042	0.169936	-0.667113	0.
pcc	0.124032	0.206507	-0.460050	0.503341	0.177327	0.168592	0.600092	1.000000	0.415033	0.257768	...	-0.534564	0.146742	-0.499401	0.
ba	0.068353	0.174555	-0.516392	0.516104	0.381929	0.273177	0.481227	0.415033	1.000000	0.318095	...	-0.397500	0.170071	-0.343299	0.
bgr	0.301915	0.190113	-0.544781	0.518123	0.730050	0.493857	0.430646	0.257768	0.318095	1.000000	...	-0.443818	0.212093	-0.418085	0.
bu	0.190636	0.316287	-0.545319	0.661940	0.312259	0.378478	0.613318	0.366726	0.205351	0.326496	...	-0.706582	0.128961	-0.621456	0.
sc	0.189721	0.386551	-0.563122	0.702889	0.347196	0.410408	0.588517	0.361965	0.229238	0.331284	...	-0.726187	0.123953	-0.639021	0.
sod	-0.102933	-0.224710	0.539285	-0.599334	-0.242491	-0.344916	-0.520324	-0.473954	-0.221374	-0.284968	...	0.570045	-0.176238	0.465125	-0.
pot	0.006866	0.127801	-0.075057	0.209492	0.271954	-0.019319	0.176150	-0.030297	-0.000279	0.102226	...	-0.213488	-0.107559	-0.193783	0.
hemo	-0.245645	-0.282365	0.682086	-0.784745	-0.385511	-0.452566	-0.733140	-0.531182	-0.410353	-0.434158	...	0.856775	-0.337435	0.743999	-0.
pcv	-0.235116	-0.353504	0.678472	-0.775528	-0.404821	-0.422537	-0.718042	-0.534564	-0.397500	-0.443818	...	1.000000	-0.349607	0.739019	-0.
wc	0.153132	0.008274	-0.288930	0.314574	0.201000	0.108857	0.169936	0.146742	0.170071	0.212093	...	-0.349607	1.000000	-0.272390	0.
rc	-0.242235	-0.228533	0.619092	-0.640099	-0.377726	-0.379378	-0.667113	-0.499401	-0.343299	-0.418085	...	0.739019	-0.272390	1.000000	-0.
htn	0.372348	0.334951	-0.648168	0.796876	0.577286	0.442400	0.666767	0.432876	0.314961	0.579407	...	-0.752043	0.223916	-0.671740	1.
dm	0.323957	0.218096	-0.639391	0.678582	0.591010	0.511777	0.636288	0.321900	0.367477	0.663012	...	-0.655039	0.287010	-0.594881	0.
cad	0.269868	0.257709	-0.379305	0.374755	0.466658	0.293269	0.384223	0.352255	0.297063	0.459164	...	-0.375627	0.021259	-0.362439	0.
appet	-0.170259	-0.145047	0.523944	-0.578080	-0.220547	-0.418639	-0.528435	-0.432515	-0.187815	-0.338924	...	0.629102	-0.328730	0.556182	-0.
pe	0.232327	0.117878	-0.633622	0.622268	0.374128	0.282868	0.606234	0.350171	0.393819	0.336141	...	-0.606829	0.282628	-0.566384	0.
ane	0.105809	0.311097	-0.413252	0.569529	0.179811	0.209797	0.545380	0.485941	0.141344	0.139854	...	-0.655724	0.139224	-0.581576	0.
classification	0.305119	0.326567	-0.790102	0.925816	0.510615	0.586391	0.775388	0.509915	0.468845	0.591217	...	-0.827983	0.407570	-0.719978	0.

25 rows × 25 columns

In [23]:
```
X = df.drop(['classification', 'sg', 'appet', 'rc', 'pcv', 'hemo', 'sod'], axis = 1)
y = df['classification']
X.columns
```

Out[23]:
```
Index(['age', 'bp', 'al', 'su', 'rbc', 'pc', 'pcc', 'ba', 'bgr', 'bu', 'sc',
       'pot', 'wc', 'htn', 'dm', 'cad', 'pe', 'ane'],
      dtype='object')
```

In [24]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators = 20)
model.fit(X_train, y_train)
```

Out[24]:
```
RandomForestClassifier(n_estimators=20)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [25]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
confusion_matrix(y_test, model.predict(X_test))
```

Out[25]:
```
array([[23,  0],
       [ 0,  9]], dtype=int64)
```

Out[25]:
```
array([[23,  0],
       [ 0,  9]], dtype=int64)
```

In [26]:
```
print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test))*100, 2)}%")
```

```
Accuracy is 100.0%
```

In [27]:
```
import pickle
pickle.dump(model, open('kidney.pkl', 'wb'))
```

In [ ]:
```

```

7.8.5 LIVER DISEASE NOTEBOOK

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
data = pd.read_csv('liver.csv')
```

In [3]:
```
data.head()
```

Out[3]:

	Age	Gender	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Albumi
0	65	Female	0.7	0.1	187	16	18	6.8	3.3	
1	62	Male	10.9	5.5	699	64	100	7.5	3.2	
2	62	Male	7.3	4.1	490	60	68	7.0	3.3	
3	58	Male	1.0	0.4	182	14	20	6.8	3.4	
4	72	Male	3.9	2.0	195	27	59	7.3	2.4	

In [4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 583 entries, 0 to 582
Data columns (total 11 columns):
Age                         583 non-null int64
Gender                      583 non-null object
Total_Bilirubin             583 non-null float64
Direct_Bilirubin            583 non-null float64
Alkaline_Phosphotase        583 non-null int64
Alamine_Aminotransferase    583 non-null int64
Aspartate_Aminotransferase  583 non-null int64
Total_Protiens              583 non-null float64
Albumin                     583 non-null float64
Albumin_and_Globulin_Ratio  579 non-null float64
Dataset                     583 non-null int64
dtypes: float64(5), int64(5), object(1)
memory usage: 50.2+ KB
```

In [5]:
```
data.describe()
```

Out[5]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protiens	Albumin	Alt
count	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	583.000000	
mean	44.746141	3.298799	1.486106	290.576329	80.713551	109.910806	6.483190	3.141852	
std	16.189833	6.209522	2.808498	242.937989	182.620356	288.918529	1.085451	0.795519	
min	4.000000	0.400000	0.100000	63.000000	10.000000	10.000000	2.700000	0.900000	
25%	33.000000	0.800000	0.200000	175.500000	23.000000	25.000000	5.800000	2.600000	
50%	45.000000	1.000000	0.300000	208.000000	35.000000	42.000000	6.600000	3.100000	
75%	58.000000	2.600000	1.300000	298.000000	60.500000	87.000000	7.200000	3.800000	
max	90.000000	75.000000	19.700000	2110.000000	2000.000000	4929.000000	9.600000	5.500000	

In [6]:
```
data.isnull().sum()
```

Out[6]:
```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  4
Dataset                     0
```

ion.ipynb#

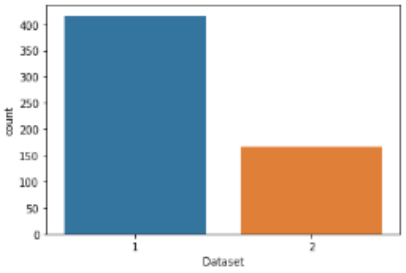
In [7]:
```
data['Albumin_and_Globulin_Ratio'] = data['Albumin_and_Globulin_Ratio'].fillna(0.947064)
```

In [8]:
```
data.isnull().sum()
```

Out[8]:
```
Age                         0
Gender                      0
Total_Bilirubin             0
Direct_Bilirubin            0
Alkaline_Phosphotase        0
Alamine_Aminotransferase    0
Aspartate_Aminotransferase  0
Total_Protiens              0
Albumin                     0
Albumin_and_Globulin_Ratio  0
Dataset                     0
dtype: int64
```

In [9]:
```
import seaborn as sns
sns.countplot('Dataset', data = data)
```

Out[9]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28b09f22a48>
```



Here 2 means suffering with disease and 1 means not suffering with disease.

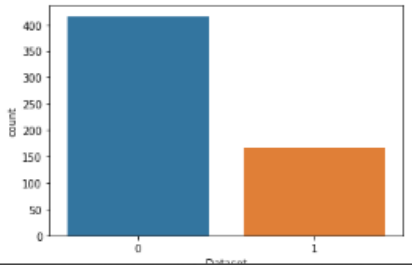
Replacing 2 with 1 and 1 with 0, for better understanding.

In [10]:
```
data['Dataset'] = data['Dataset'].replace([2,1],[1,0])
data['Dataset'].head()
```

Out[10]:
```
0    0
1    0
2    0
3    0
4    0
Name: Dataset, dtype: int64
```

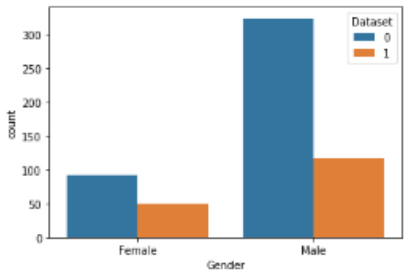
In [11]:
```
import seaborn as sns
sns.countplot('Dataset', data = data)
```

Out[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28b0b213748>
```



In [12]:
```
sns.countplot('Gender', data = data, hue = 'Dataset')
```

Out[12]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28b0b284348>
```



In [13]:
```
data = pd.get_dummies(data, columns = ['Gender'], drop_first = True)
```

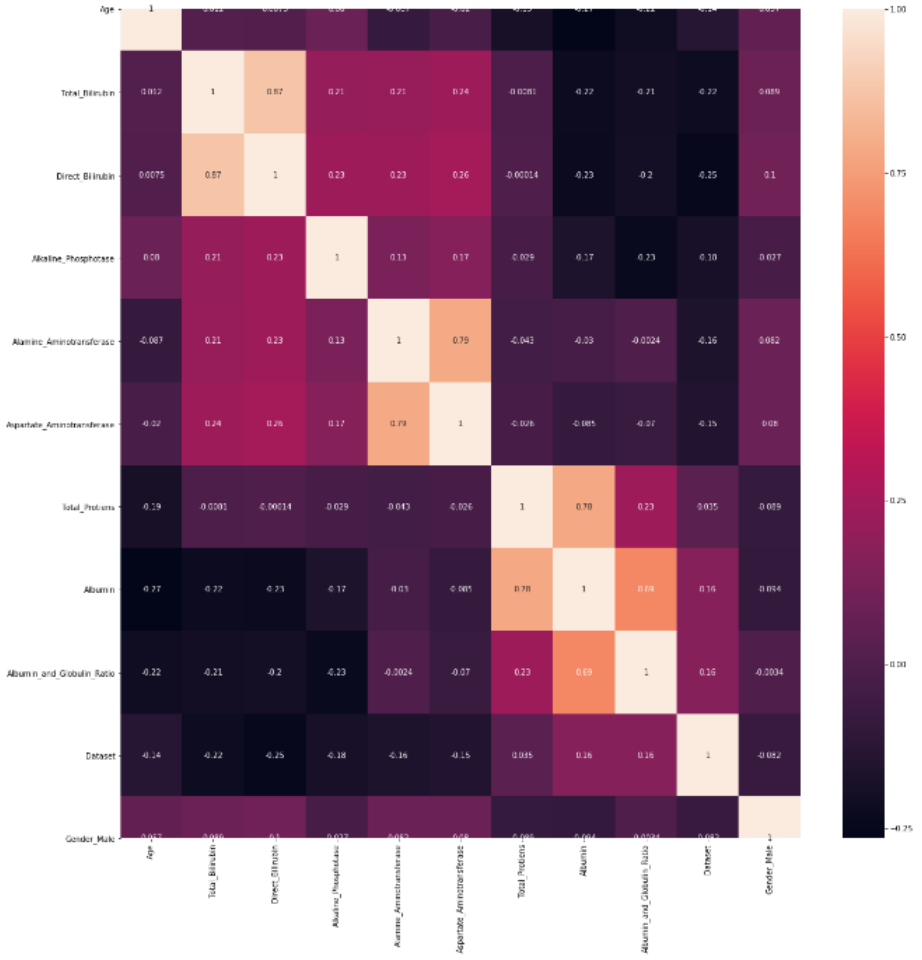
In [14]:
```
data.head()
```

Out[14]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Proteins	Albumin	Albumin_and_Gl
0	65	0.7	0.1	187	16	18	6.8	3.3	
1	62	10.9	5.5	699	64	100	7.5	3.2	
2	62	7.3	4.1	490	60	68	7.0	3.3	
3	58	1.0	0.4	182	14	20	6.8	3.4	
4	72	3.9	2.0	195	27	59	7.3	2.4	

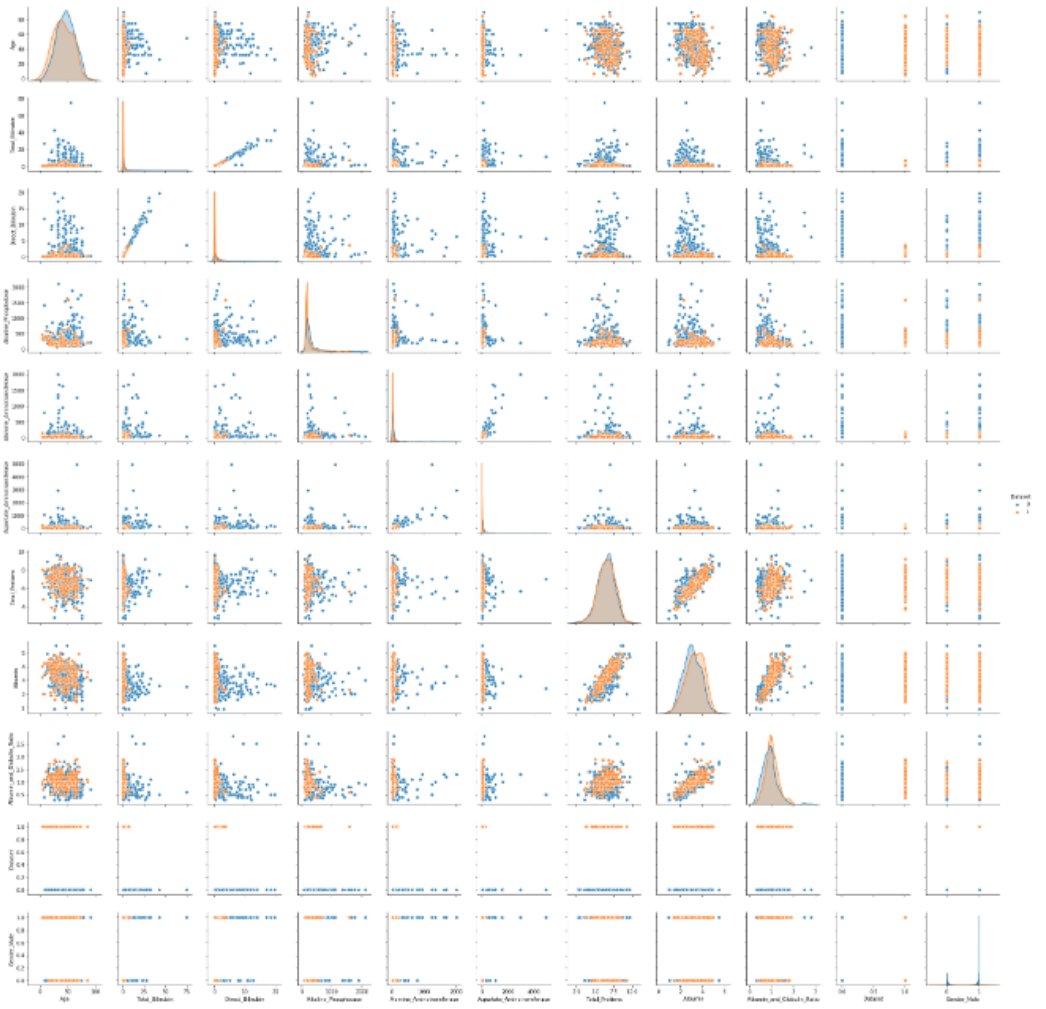
In [15]:
```
plt.figure(figsize = (20,20))
sns.heatmap(data.corr(), annot = True)
```

Out[15]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x28b0b2d8708>
```



In [16]:
```
sns.pairplot(data, hue = 'Dataset')
```

Out[16]: <seaborn.axisgrid.PairGrid at 0x28b0b40fb48>



In [17]:
```
data.corr()
```

Out[17]:

	Age	Total_Bilirubin	Direct_Bilirubin	Alkaline_Phosphotase	Alamine_Aminotransferase	Aspartate_Aminotransferase	Total_Protie
Age	1.000000	0.011763	0.007529	0.080425	-0.086883	-0.019910	-0.1874
Total_Bilirubin	0.011763	1.000000	0.874618	0.206669	0.214065	0.237831	-0.0080
Direct_Bilirubin	0.007529	0.874618	1.000000	0.234939	0.233894	0.257544	-0.0001
Alkaline_Phosphotase	0.080425	0.206669	0.234939	1.000000	0.125680	0.167196	-0.0285
Alamine_Aminotransferase	-0.086883	0.214065	0.233894	0.125680	1.000000	0.791966	-0.0425
Aspartate_Aminotransferase	-0.019910	0.237831	0.257544	0.167196	0.791966	1.000000	-0.0256
Total_Protiens	-0.187461	-0.008099	-0.000139	-0.028514	-0.042518	-0.025645	1.0000
Albumin	-0.265924	-0.222250	-0.228531	-0.165453	-0.029742	-0.085290	0.7840

In [18]:
```
# X = data[['Albumin_and_Globulin_Ratio', 'Albumin', 'Total_Protiens', 'Aspartate_Aminotransferase', 'Alamine_Aminotransferase',
X = data.drop('Dataset', axis = 1)
y = data['Dataset']
```

In [19]:
```
X.columns
```

Out[19]:
```
Index(['Age', 'Total_Bilirubin', 'Direct_Bilirubin', 'Alkaline_Phosphotase',
       'Alamine_Aminotransferase', 'Aspartate_Aminotransferase',
       'Total_Protiens', 'Albumin', 'Albumin_and_Globulin_Ratio',
       'Gender_Male'],
      dtype='object')
```

In [20]:
```
# from imblearn.combine import SMOTETomek
# smk = SMOTETomek(random_state = 42)
# X, y = smk.fit_sample(X,y)
# X.shape, y.shape
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [22]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.1, random_state = 42)
```

In [23]:
```
print("Train Set: ", X_train.shape, y_train.shape)
print("Test Set: ", X_test.shape, y_test.shape)
```

```
Train Set:  (524, 10) (524,)
Test Set:  (59, 10) (59,)
```

In [24]:
```
from sklearn.ensemble import RandomForestClassifier
model = RandomForestClassifier(n_estimators=20)
model.fit(X_train, y_train)
```

Out[24]:
```
RandomForestClassifier(n_estimators=20)
```

In [25]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
```

In [26]:
```
confusion_matrix(y_test, model.predict(X_test))
```

Out[26]:
```
array([[38,  7],
       [ 6,  8]], dtype=int64)
```

In [27]:
```
print(f"Accuracy is {round(accuracy_score(y_test, model.predict(X_test))*100,2)}")
```

```
Accuracy is 77.97
```

In [1]:
```
import pickle
pickle.dump(model, open('liver.pkl', 'wb'))
```

CHAPTER 8

CONCLUSION

The Multiple Disease Prediction System project aims to develop a web-based application that predicts multiple diseases based on user inputs. The project utilizes the Flask framework, Random Forest algorithm, and Jupyter Notebook for implementation.

The project begins with collecting or preparing a suitable dataset that contains relevant features and target labels for multiple diseases. The dataset is then used to train a Random Forest model in a Jupyter Notebook. After training, the model is saved using the pickle library for future use.

The Flask application is set up to handle user interactions and predictions. It involves importing the necessary libraries, loading the trained model, defining routes and view functions, and setting up HTML templates for the user interface. The user can input their symptoms or relevant data, which is then processed and used by the loaded Random Forest model to make predictions. The results are displayed to the user through the web interface.

The project allows users to easily access the Multiple Disease Prediction System through a web browser. It provides a user-friendly interface for entering symptoms or relevant information and obtaining predictions for multiple diseases. By leveraging the power of Random Forest algorithm, the system can make accurate predictions based on the trained model.

Overall, the Multiple Disease Prediction System project demonstrates the integration of machine learning algorithms, web development frameworks, and user interface design to create a practical and accessible tool for disease prediction. It showcases the potential of using Flask, Random Forest algorithm, and Jupyter Notebook in combination to develop such predictive systems.

The Multiple Disease Prediction System project aims to address the need for an efficient and accessible tool to predict multiple diseases based on user inputs. By combining various technologies and techniques, the project provides a user-friendly web interface that allows users to input their symptoms or relevant data and receive predictions for various diseases.

The project begins by preparing a suitable dataset that contains features and target labels related to multiple diseases. This dataset is essential for training the Random Forest model, which is a powerful machine learning algorithm capable of handling complex classification tasks.

Using a Jupyter Notebook, the dataset is loaded and split into features (X) and target labels (y). Any necessary preprocessing steps, such as data normalization or handling missing values, are applied to ensure the dataset's quality. The Random Forest Classifier from the scikit-learn library is then imported, and the model is trained using the dataset.

After training the model, it is saved using the pickle library. This allows the model to be loaded and used later in the Flask application without the need for retraining.

The Flask framework is utilized to develop the web interface for the Multiple Disease Prediction System. Flask provides a lightweight and flexible environment for handling web requests, rendering HTML templates, and managing routes.

In the Flask application, the necessary libraries, including Flask and pickle, are imported. The trained Random Forest model is loaded using the pickle library. Routes and associated view functions are defined to handle different URLs and user interactions.

HTML templates are created to structure the web pages, including the home page, prediction form, and result display. The Flask render_template function is used to render the appropriate templates and pass data to them.

When a user interacts with the application by entering symptoms or relevant data, the form submission is handled by the corresponding view function. The input data is preprocessed if required, and then passed to the loaded Random Forest model for prediction. The model generates predictions based on the input, and the results are displayed to the user through the web interface.

The Flask application is run, and the Multiple Disease Prediction System becomes accessible through a web browser. Users can input their symptoms or relevant data, and the system utilizes the Random Forest model to predict multiple diseases based on the trained model.

The project demonstrates the integration of machine learning algorithms, web development frameworks, and user interface design to create a practical and user-friendly system for disease prediction. It showcases the effectiveness of Random Forest algorithm in making accurate predictions and the versatility of Flask in building web-based interfaces.

In conclusion, the Multiple Disease Prediction System project addresses the need for a reliable and accessible tool for disease prediction. It combines the power of machine learning with the flexibility of web development to provide an efficient and user-friendly solution.

CHAPTER 9

FUTURE ENHANCEMENTS

There are several potential future enhancements that can be considered for the Multiple Disease Prediction System project:

1. Expansion of Disease Coverage: Currently, the system predicts multiple diseases based on the available dataset. However, the system can be enhanced by incorporating additional datasets or expanding the existing dataset to cover a broader range of diseases. This would provide users with a more comprehensive disease prediction capability.

2. Integration of Advanced Machine Learning Techniques: While the Random Forest algorithm is effective, exploring and integrating other advanced machine learning techniques could improve the prediction accuracy of the system. Techniques such as deep learning or ensemble methods could be explored to enhance the performance of the disease prediction model.

3. Real-time Data Integration: Enhancing the system to integrate real-time data sources, such as medical databases or APIs, would enable the system to provide up-to-date and accurate predictions. This could involve collecting real-time patient data, medical research data, or symptom data from wearable devices, and incorporating it into the prediction model.

4. Incorporation of User Feedback and Refinement: Gathering user feedback on the predictions and incorporating it into the system can help refine and improve the accuracy of the predictions. This can be achieved through user feedback mechanisms, allowing users to provide information on the correctness of the predictions and adjusting the model accordingly.

5. Integration of Explainable AI: Incorporating explainable AI techniques can help provide users with insights into how the predictions are made. This can involve techniques such as feature importance analysis, decision rule extraction, or model interpretability methods. Providing explanations for the predictions can enhance user trust and understanding of the system.

6. Mobile Application Development: Developing a mobile application version of the Multiple Disease Prediction System would make it more accessible to a wider audience. A mobile app would allow users to access the system from their smartphones or tablets, enabling convenient and on-the-go disease predictions.

7. Integration of Additional Features: In addition to symptom-based predictions, the system could be enhanced to include additional features such as risk factor assessment, preventive measures, or recommended treatments for specific diseases. This would provide users with more comprehensive information and support for disease management.

8. Privacy and Security Enhancements: Strengthening the privacy and security aspects of the system is crucial, especially when dealing with sensitive medical data. Implementing robust security measures, encryption techniques, and compliance with data protection regulations will ensure the confidentiality and integrity of user data.

9. Collaboration with Healthcare Professionals: Collaborating with healthcare professionals, such as doctors or medical researchers, can provide valuable insights and domain expertise. Involving medical experts in the development process can help validate the predictions, refine the model, and ensure the system aligns with medical standards and best practices.

10. Continuous Model Improvement and Updating: Regularly updating the prediction model with new data and retraining it can help improve the accuracy and reliability of the system over time. Staying up-to-date with the latest research and medical advancements in disease prediction can contribute to the continuous improvement of the system.

These future enhancements would contribute to the evolution and refinement of the Multiple Disease Prediction System, making it more accurate, reliable, user-friendly, and aligned with the needs of both healthcare professionals and users.

REFERENCES

1. Multiple Disease Prediction and Doctor Recommendation System by www.irjet.net
2. Disease Prediction Based on Prior Knowledge by www.hcup- us.ahrq.gov/nisoverview.jsp
3. GDPS - General Multiple Disease PredictionSystem by www.irjet.net
4. Multiple Disease PredictionUsing Machine Learning by International Research Journal of Engineering and Technology (IRJET).
5. Kaveeshwar, S.A., and Cornwall, J., 2014, "The current state of disease mellitus in India". AMJ, 7(1), pp. 45-48.
6. Dean, L., McEntyre, J., 2004, "The Genetic Landscape of Disease [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); Chapter 1, Introduction to Disease. 2004 Jul 7.
7. Machine Learning Methods Used in Disease by www.wikipedia.com
8. https://www.researchgate.net/publication/325116774_disease_prediction_using_machine_learning_techniques
9. https://ieeexplore.ieee.org/document/8819782/disease_prediction
10. Algorithms Details from www.dataspirant.com
11. https://www.youtube.com/disease_prediction
12. https://www.slideshare.com/disease_prediction
13. https:/ en.wikipedia.org/machine_learning_algorithms
14. https://en.wikipedia.org/wiki/Python_(programming_language)
15. https://wiki.python.org/TkInter
16. https://creately.com/lp/uml-diagram-tool/
17. https://app.diagrams.net/