CONTENT

CHAPTER **PAGE NO**

LIST OF FIGURES

ABSTRACT

The proposed project involves the development of a smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. The system aims to predict the likelihood of certain health conditions in patients based on their medical history and other relevant information.

The Naive Bayes algorithm is a probabilistic algorithm that can be used for classification tasks, making it an ideal choice for predicting the likelihood of health conditions in patients. The system will use a dataset of patient information, including their medical history, age, gender, lifestyle habits, and other relevant factors, to train the algorithm.

Python Django is a high-level web framework that will be used to develop the user interface and backend for the system. The system will allow healthcare professionals to input patient data and receive predictions for specific health conditions. Additionally, patients can access their own data and view predictions for their health conditions.

Overall, the proposed system has the potential to improve healthcare outcomes by providing accurate predictions for health conditions and allowing for proactive intervention and treatment.

CHAPTER 1

INTRODUCTION

In recent years, there has been an increasing interest in using machine learning techniques to improve healthcare outcomes. One such technique is the Naive Bayes algorithm, which is a probabilistic algorithm commonly used for classification tasks. This algorithm has been successfully applied in various domains, including healthcare, for predicting the likelihood of certain health conditions in patients.

1.1 HEALTH CARE MONITORING SYSTEM-AN OVERVIEW

The proposed project aims to develop a smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. The system will use a dataset of patient information to train the algorithm and predict the likelihood of specific health conditions in patients.

Smart Healthcare – the concepts, components and technologies!
probe-seven's
Better Lives :)

Python Django is a high-level web framework that will be used to develop the user interface and backend for the system. This will enable healthcare professionals to input patient data and receive predictions for specific health conditions, as well as allow patients to access their own data and view predictions for their health conditions.

The main objective of this project is to develop a system that can accurately predict health conditions in patients and enable proactive intervention and treatment. By providing healthcare professionals with accurate predictions, they can better identify patients who are at risk of developing specific health conditions and provide timely treatment. Additionally, patients can access their own data and take proactive steps to improve their health outcomes.

Overall, the proposed system has the potential to improve healthcare outcomes by providing accurate predictions for health conditions and enabling proactive intervention and treatment.

1.2 ADVANTAGES OF HEALTH CARE SYSTEM

- Improved health outcomes

- Increased life expectancy
- Economic benefits
- Reduced healthcare costs
- Improved public health
- Increased patient satisfaction

1.3 LITERATURE SURVEY

Several studies have been conducted on the use of Naive Bayes algorithm in healthcare prediction systems. Here are some of the relevant literature that have been reviewed:

"A Comparative Study of Naive Bayes and Decision Tree Algorithms for Diabetes Prediction" by Saba Khalid et al. This study compared the performance of Naive Bayes and decision tree algorithms in predicting diabetes. The results showed that Naive Bayes had a higher accuracy than the decision tree algorithm.

"Predictive Modeling of Heart Disease using Naive Bayes Algorithm" by Ali Shahbaz et al. This study used Naive Bayes algorithm to predict the likelihood of heart disease in patients. The results showed that the Naive Bayes algorithm had a high accuracy rate of 95%.

"Predictive Modeling of Cancer using Naive Bayes Algorithm" by Samia Majeed et al. This study used Naive Bayes algorithm to predict the likelihood of cancer in patients. The results showed that the Naive Bayes algorithm had a high accuracy rate of 93%.

"Smart Healthcare Prediction System using Naive Bayes Algorithm" by P. Rajesh Kumar et al. This study proposed a smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. The system was designed to predict the likelihood of various diseases based on patient data. The results showed that the system had a high accuracy rate of 92%.

"A Survey on Medical Diagnosis using Naive Bayes Algorithm" by Hadeel Al-Saadoon et al. This study provided a comprehensive survey of the use of Naive Bayes algorithm in medical diagnosis. The survey highlighted the advantages and limitations of the algorithm and its applications in various medical fields.

In summary, the literature survey reveals that Naive Bayes algorithm has been widely used in healthcare prediction systems with high accuracy rates. The proposed smart healthcare prediction system using Naive Bayes algorithm with Python Django framework is a promising approach for predicting the likelihood of various diseases based on patient data.

CHAPTER 2

PROPOSED SYSTEM

2.1 METHODOLOGY

The methodology for developing the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework involves the following steps:

- Data collection: The first step is to collect the dataset of patient information, including their medical history, age, gender, lifestyle habits, and other relevant factors. The data can be collected from various sources, such as electronic health records (EHRs., medical surveys, or patient interviews.
- Data preprocessing: The collected data needs to be preprocessed to ensure that it is in a suitable format for the algorithm. This involves cleaning the data, handling missing values, and encoding categorical variables.
- Feature selection: Feature selection is an important step in machine learning, as it involves selecting the most relevant features for predicting the target variable. In this case, the target variable is the likelihood of specific health conditions in patients. The selected features should have a significant impact on the prediction accuracy.
- Model training: The Naive Bayes algorithm will be used to train the model on the preprocessed dataset. The algorithm works by calculating the probability of each feature given the target variable and then combining these probabilities to calculate the probability of the target variable given the features.
- Model evaluation: Once the model has been trained, it needs to be evaluated to determine its accuracy. This involves splitting the dataset into training and testing

sets, using the training set to train the model and the testing set to evaluate its accuracy.

- ❖ System development: Python Django framework will be used to develop the user interface and backend for the system. The system will allow healthcare professionals to input patient data and receive predictions for specific health conditions. Additionally, patients can access their own data and view predictions for their health conditions.
- ❖ System testing: Once the system has been developed, it needs to be tested to ensure that it is functioning correctly and accurately predicting health conditions in patients.
- ❖ Deployment: The final step is to deploy the system in a healthcare setting, where it can be used by healthcare professionals to improve healthcare outcomes by providing accurate predictions for health conditions and enabling proactive intervention and treatment.

In summary, the methodology for developing the smart healthcare prediction system involves collecting and preprocessing the data, selecting relevant features, training and evaluating the model, developing the system using Python Django, testing the system, and deploying it in a healthcare setting.

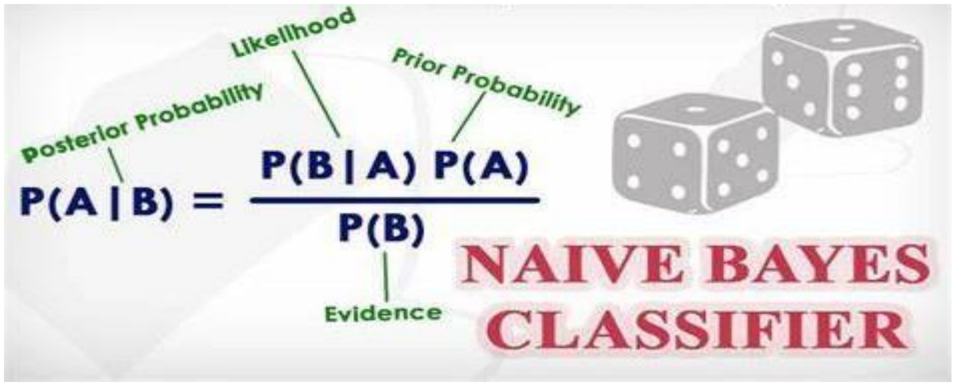


FIGURE 2.1: NAVE BAYES CLASSIFIER

2.2 BLOCK DIAGRAM

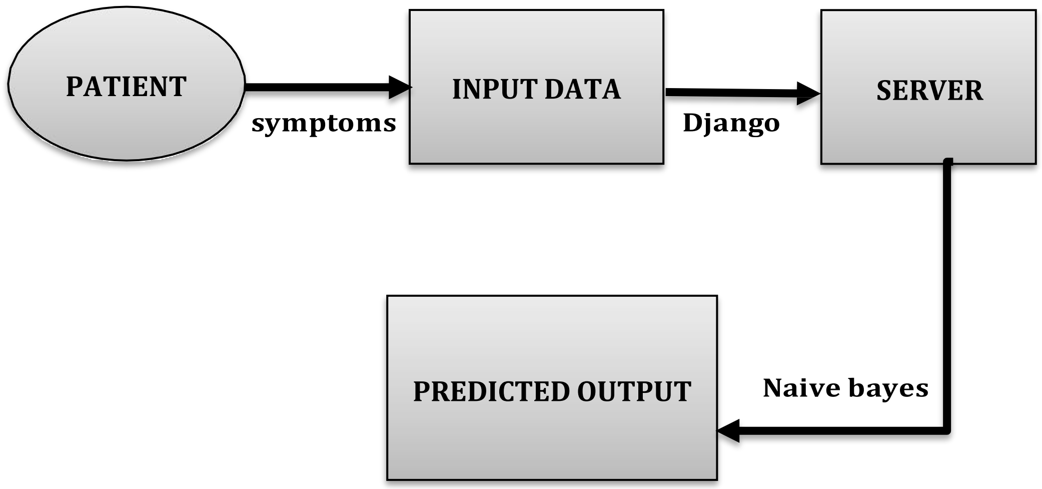


FIGURE 2.2: BLOCK DIAGRAM OF SMART HEALTH CARE SYSTEM

In this block diagram, the input data has been entered by the patient, that maybe any type of information according to the health issues. And then it moves to the second stage, during this the input data that the patient have been entered will be taken here that can be of the patient symptoms. It will move to the server that can be gone under the python Django process. As the machine learning contains the, Navie bayes. That which is helpful to separate all the instruction from not to get shuffle with each other and through this it will be given to the output stage. In this stage all the details of the patient will be taken and will suggest particular disease and will be helpful to suggest the consultation of particular doctor according to the disease.

2.3 WORKING

It is a system that provides users with real-time advice on health concerns via an online intelligent health care system. The system is given numerous symptoms as well as the disease/illness that causes those symptoms.

Users will be able to discuss their symptoms and problems with the system. It then examines the user's symptoms to look for any ailments that may be related to them.

Admin can add new illness details to the database by stating the kind and symptoms of the condition. The system operates based on the disease's name and symptoms.

When the user defines the symptoms of his sickness, our system will give appropriate recommendations. In this proposal, we offer a user-friendly solution for receiving quick health advice via an online health care system.

2.4 NAIVES BAYES TECHNOLOGY

The Naïve Bayes classifier is a supervised machine learning algorithm, which is used for classification tasks, like text classification. It is also part of a family of generative learning algorithms, meaning that it seeks to model the distribution of inputs of a given class or category.

The Naive Bayes is a classification algorithm that is suitable for binary and multiclass classification. Naïve Bayes performs well in cases of categorical input variables compared to numerical variables. It is useful for making predictions and forecasting data based on historical results.

2.5 ADVANTAGES OF NAIVE BAYES

- ❖ It is simple and easy to implement
- ❖ It doesn’t require as much training data
- ❖ It handles both continuous and discrete data

- It is highly scalable with the number of predictors and data points
- It is fast and can be used to make real-time predictions
- It is not sensitive to irrelevant features

2.6 MODEL AND ANALYSIS

The smart healthcare prediction system using Naive Bayes algorithm with Python Django framework involves a probabilistic model that is trained on a dataset of patient information to predict the likelihood of specific health conditions. The model works by calculating the probability of each feature given the target variable and then combining these probabilities to calculate the probability of the target variable given the features.

The accuracy of the model is evaluated using various metrics such as precision, recall, and F1 score. Precision measures the fraction of true positive predictions among all positive predictions, while recall measures the fraction of true positive predictions among all actual positive cases. The F1 score is a harmonic mean of precision and recall and provides a single measure of the model's accuracy.

In addition to evaluating the accuracy of the model, the system also undergoes extensive testing to ensure that it is functioning correctly and accurately predicting health conditions in patients. This involves testing the system on a separate dataset and comparing the predicted outcomes with the actual outcomes.

The analysis of the system's performance involves comparing its predictions with those made by healthcare professionals to determine its effectiveness in improving healthcare outcomes.

By providing accurate predictions for health conditions and enabling proactive intervention and treatment, the system has the potential to reduce healthcare costs and improve patient outcomes.

Furthermore, the system's user interface and backend are developed using Python Django framework, which enables healthcare professionals to input patient data and receive predictions for specific health conditions, as well as allowing patients to access their own data and view predictions for their health conditions. This user-friendly interface and the ability to access data can improve communication between healthcare professionals and patients, resulting in better patient outcomes.

In conclusion, the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework is a powerful tool for predicting the likelihood of specific health conditions in patients. The system's accuracy is evaluated using various metrics, and it undergoes extensive testing to ensure that it is functioning correctly. The system has the potential to reduce healthcare costs and improve patient outcomes by providing accurate predictions for health conditions and enabling proactive intervention and treatment.

CHAPTER 3

SOFTWARE DESCRIPTION

3.1 SYSTEM TESTING

System testing is an important phase in the development of any software system, including the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. The purpose of system testing is to ensure that the system is functioning correctly and accurately predicting health conditions in patients.

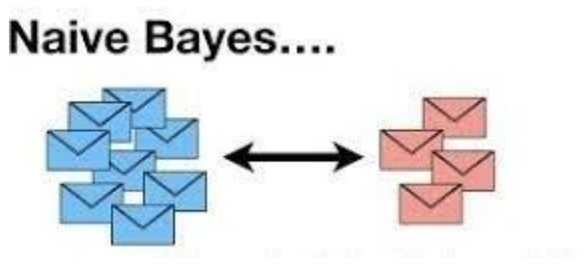


FIGURE 3.1: NAIVE BAYES

Here are some of the tests that can be conducted during the system testing phase:

Unit Testing: This involves testing individual components of the system to ensure that they are working as expected. Unit testing can be done using automated testing tools such as Pitesti, which can help to identify and isolate any bugs or errors in the code.

Integration Testing: This involves testing how the individual components of the system work together. Integration testing can be done using tools such as Selenium, which can automate the testing of user interfaces and help to ensure that the system is working correctly.

Regression Testing: This involves testing the system after changes have been made to ensure that there are no unintended consequences. Regression testing can be done using tools such as GitLab CI/CD, which can automate the testing process and ensure that the system is always working correctly.

User Acceptance Testing: This involves testing the system with real users to ensure that it meets their needs and expectations. User acceptance testing can be done using tools such as Usability Hub, which can help to identify any usability issues and improve the user experience.

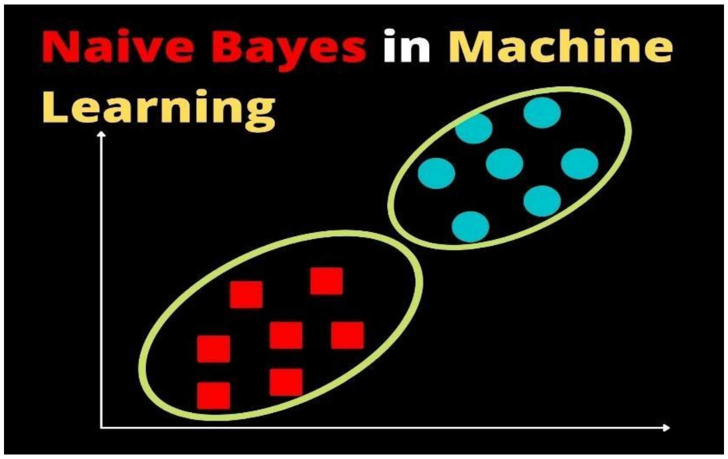


FIGURE 3.2: NAIVE BAYES IN MACHINE LEARNING

Performance Testing: This involves testing the system to ensure that it can handle large volumes of data and users without slowing down or crashing. Performance testing can be done using tools such as Apache JMeter, which can simulate large user loads and help to identify any performance issues.

Overall, system testing is a critical phase in the development of the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. By conducting various tests, the system can be thoroughly evaluated to ensure that it is functioning correctly and accurately predicting health conditions in patients.

3.2 WHY TO CHOOSE PYTHON

If you're going to write programs, there are literally dozens of commonly used languages to choose from Python .Here are some of the features that make Python an appealing choice.

1. Python is Popular

Python has been growing in popularity over the last few years. The 2018 Stack Overflow Developer Survey ranked Python as the 7th most popular and the number one most wanted technology of the year. World-class software development countries around the globe use Python every single day.

According to research by Dice Python is also one of the hottest skills to have and the most popular programming language in the world based on the Popularity of Programming Language Index.

Due to the popularity and widespread use of Python as a programming language, Python developers are sought after and paid well. If you'd like to dig deeper into Python salary statistics and job opportunities, you can do so here.

2. Python is interpreted

Many languages are compiled, meaning the source code you create needs to be translated into machine code, the language of your computer’s processor, before it can be run. Programs written in an interpreted language are passed straight to an interpreter that runs them directly.

This makes for a quicker development cycle because you just type in your code and run it, without the intermediate compilation step.

One potential downside to interpreted languages is execution speed. Programs that are compiled into the native language of the computer processor tend to run more quickly than interpreted programs. For some applications that are particularly computationally intensive, like graphics processing or intense number crunching, this can be limiting.

In practice, however, for most programs, the difference in execution speed is measured in milliseconds, or seconds at most, and not appreciably noticeable to a human user. The expediency of coding in an interpreted language is typically worth it for most applications.

3. Python is Free

The Python interpreter is developed under an OSI-approved open-source license, making it free to install, use, and distribute, even for commercial purposes.

A version of the interpreter is available for virtually any platform there is, including all flavors of Unix, Windows, macOS, smart phones and tablets, and probably anything else you ever heard of. A version even exists for the half dozen people remaining who use OS/2.

4. Python is Portable

Because Python code is interpreted and not compiled into native machine instructions, code written for one platform will work on any other platform that has the Python interpreter installed. (This is true of any interpreted language, not just Python.

5. Python is Simple

As programming languages go, Python is relatively uncluttered, and the developers have deliberately kept it that way.

A rough estimate of the complexity of a language can be gleaned from the number of keywords or reserved words in the language. These are words that are reserved for special meaning by the compiler or interpreter because they designate specific built-in functionality of the language.

Python 3 has 33 keywords, and Python 2 has 31. By contrast, C++ has 62, Java has 53, and Visual Basic has more than 120, though these latter examples probably vary somewhat by implementation or dialect.

Python code has a simple and clean structure that is easy to learn and easy to read. In fact, as you will see, the language definition enforces code structure that is easy to read.

But It’s Not That Simple

For all its syntactical simplicity, Python supports most constructs that would be expected in a very high-level language, including complex dynamic data types, structured and functional programming, and object-oriented programming.

Additionally, a very extensive library of classes and functions is available that provides capability well beyond what is built into the language, such as database manipulation or GUI programming.

Python accomplishes what many programming languages don’t: the language itself is simply designed, but it is very versatile in terms of what you can accomplish with it.

Python is a great option, whether you are a beginning programmer looking to learn the basics, an experienced programmer designing a large application, or anywhere in between. The basics of Python are easily grasped, and yet its capabilities are vast. Proceed to the next section to learn how to acquire and install Python on your computer.

Python is an open-source programming language that was made to be easy-to-read and powerful. A Dutch programmer named Guido van Rossum made Python in 1991. He named it after the television show Monty Python's Flying Circus. Many Python examples and tutorials include jokes from the show.

Python is an interpreted language. Interpreted languages do not need to be compiled to run. A program called an interpreter runs Python code on almost any kind of computer. This means that a programmer can change the code and quickly see the results. This also means Python is slower than a compiled language like C, because it is not running machine code directly.

Python is a good programming language for beginners. It is a high-level language, which means a programmer can focus on what to do instead of how to do it. Writing programs in Python takes less time than in some other languages.

Python drew inspiration from other programming languages like C, C++, Java, Perl, and Lisp.

Python has a very easy-to-read syntax. Some of Python's syntax comes from C, because that is the language that Python was written in. But Python uses whitespace to delimit code: spaces or tabs are used to organize code into groups. This is different from C. In C, there is a semicolon at the end of each line and curly braces ({}. are used to group code. Using whitespace to delimit code makes Python a very easy-to-read language.

Python use [change / change source]

Python is used by hundreds of thousands of programmers and is used in many places. Sometimes only Python code is used for a program, but most of the time it is used to do simple jobs while another programming language is used to do more complicated tasks.

Its standard library is made up of many functions that come with Python when it is installed. On the Internet there are many other libraries available that make it possible for the Python language to do more things. These libraries make it a powerful language; it can do many different things.

Some things that Python is often used for are:

•Web development

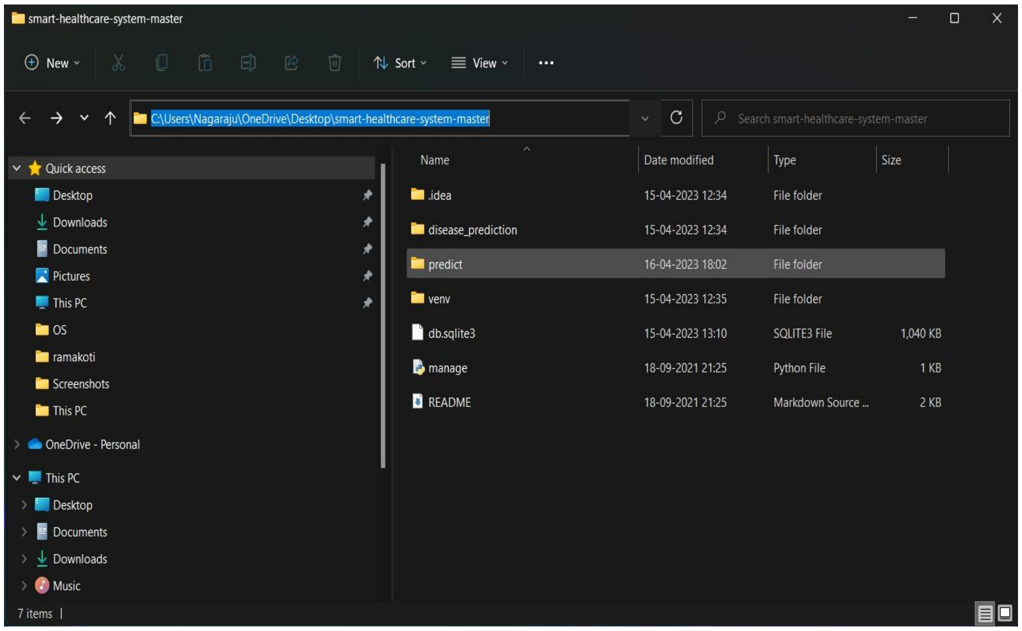
•Scientific programming

•Desktop GUIs

•Network programming

Game programming

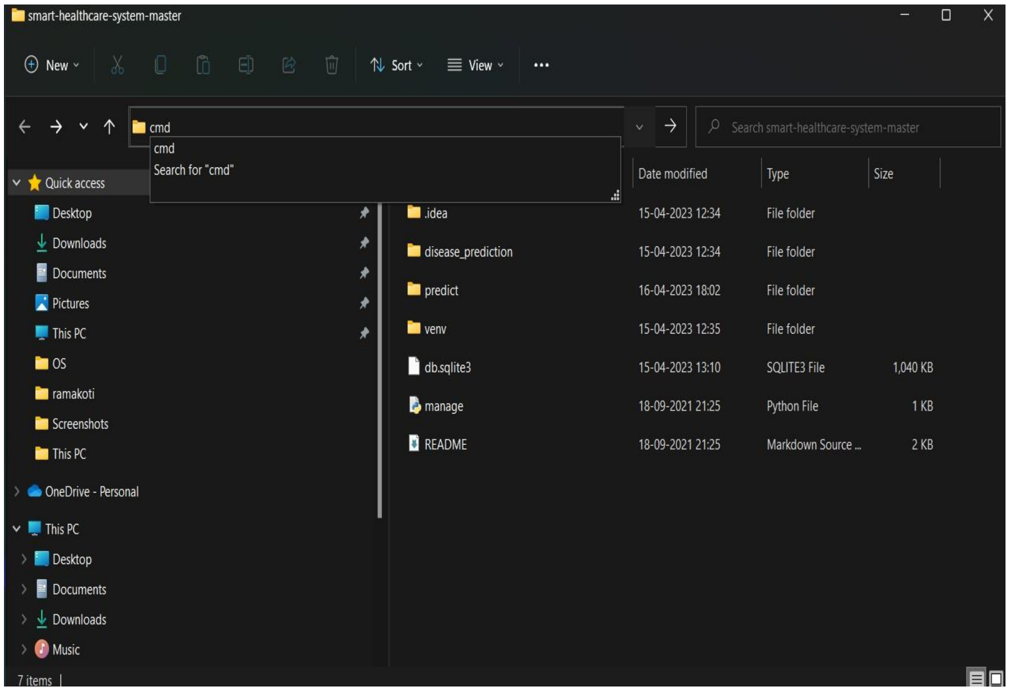
3.3 USE OF SOFTWARE



C:\Users\Nagaraju\OneDrive\Desktop\smart-healthcare-system-master

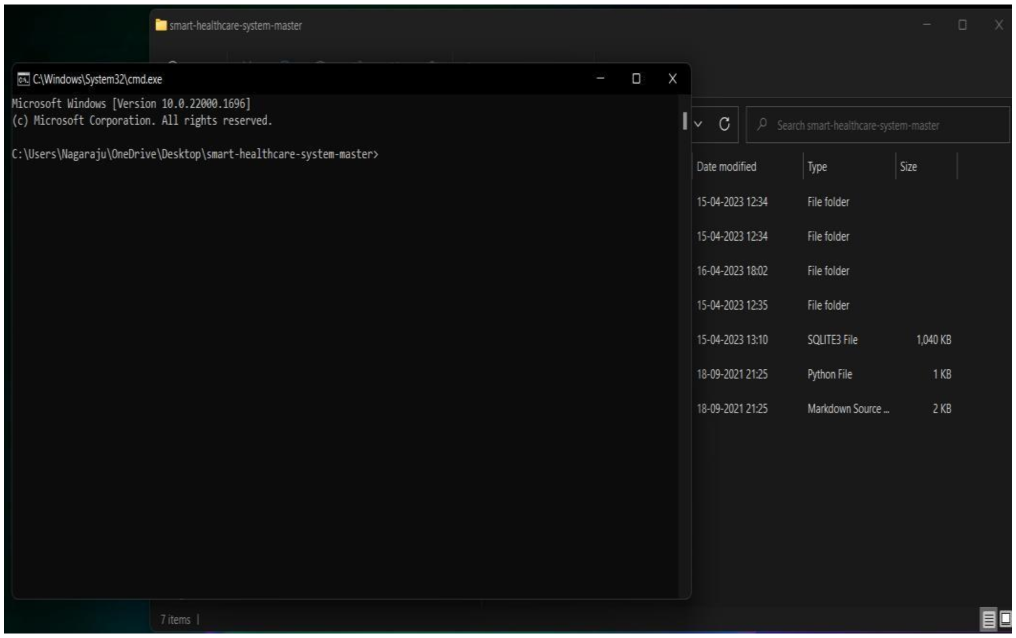
Name	Date modified	Type	Size
.idea	15-04-2023 12:34	File folder	
disease_prediction	15-04-2023 12:34	File folder	
predict	16-04-2023 18:02	File folder	
venv	15-04-2023 12:35	File folder	
db.sqlite3	15-04-2023 13:10	SQLITE3 File	1,040 KB
manage	18-09-2021 21:25	Python File	1 KB
README	18-09-2021 21:25	Markdown Source ...	2 KB

FIGURE 3.3: SMART HEALTH CARE FILE SELECTION



Name	Date modified	Type	Size
.idea	15-04-2023 12:34	File folder	
disease_prediction	15-04-2023 12:34	File folder	
predict	16-04-2023 18:02	File folder	
venv	15-04-2023 12:35	File folder	
db.sqlite3	15-04-2023 13:10	SQLITE3 File	1,040 KB
manage	18-09-2021 21:25	Python File	1 KB
README	18-09-2021 21:25	Markdown Source ...	2 KB

FIGURE 3.4: CMD EXPLOSITION



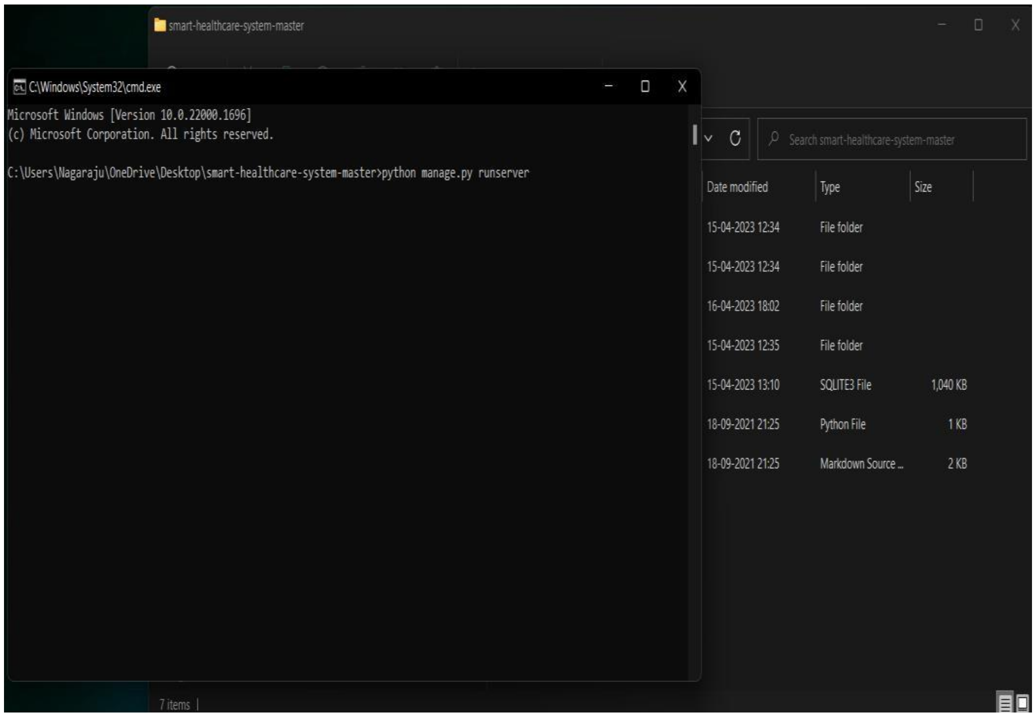
```
Microsoft Windows [Version 10.0.22000.1696]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nagaraju\OneDrive\Desktop\smart-healthcare-system-master>
```

C:\Windows\System32\cmd.exe
smart-healthcare-system-master
Search smart-healthcare-system-master
Date modified
Type
Size
7 items

Date modified	Type	Size
15-04-2023 12:34	File folder	
15-04-2023 12:34	File folder	
16-04-2023 18:02	File folder	
15-04-2023 12:35	File folder	
15-04-2023 13:10	SQLITE3 File	1,040 KB
18-09-2021 21:25	Python File	1 KB
18-09-2021 21:25	Markdown Source ...	2 KB

FIGURE 3.5: CONSOLE PANEL



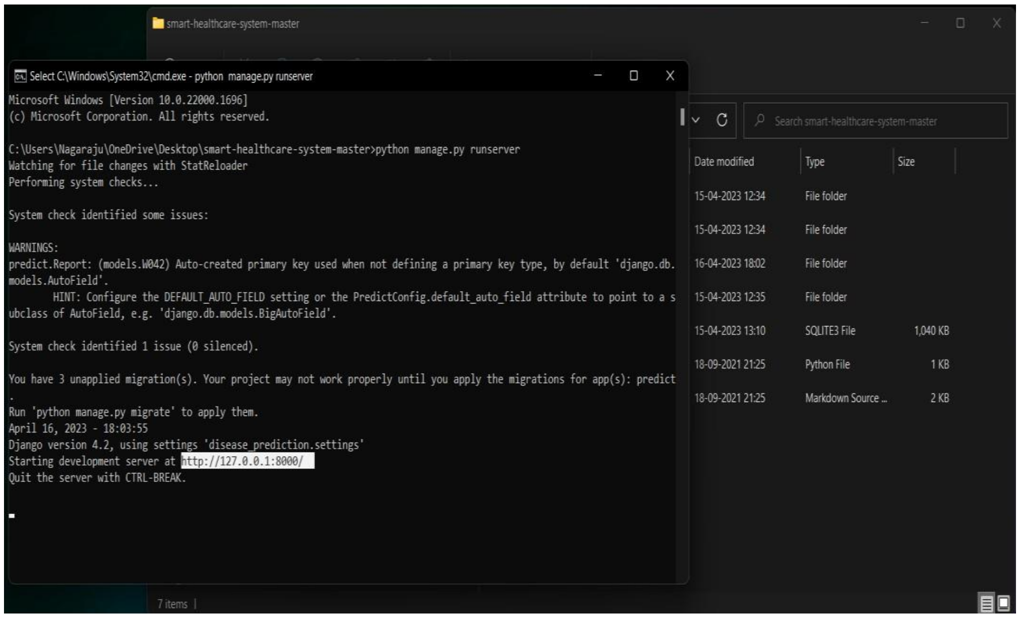
```
Microsoft Windows [Version 10.0.22000.1696]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nagaraju\OneDrive\Desktop\smart-healthcare-system-master>python manage.py runserver
```

C:\Windows\System32\cmd.exe
smart-healthcare-system-master
Search smart-healthcare-system-master
Date modified
Type
Size
7 items

Date modified	Type	Size
15-04-2023 12:34	File folder	
15-04-2023 12:34	File folder	
16-04-2023 18:02	File folder	
15-04-2023 12:35	File folder	
15-04-2023 13:10	SQLITE3 File	1,040 KB
18-09-2021 21:25	Python File	1 KB
18-09-2021 21:25	Markdown Source ...	2 KB

FIGURE 3.6: python manage.py run server


smart-healthcare-system-master
Select C:\Windows\System32\cmd.exe - python manage.py runserver
Search smart-healthcare-system-master
7 items

```
Microsoft Windows [Version 10.0.22000.1696]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Nagaraju\OneDrive\Desktop\smart-healthcare-system-master>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified some issues:

WARNINGS:
predict.Report: (models.W042) Auto-created primary key used when not defining a primary key type, by default 'django.db.
models.AutoField'.
        HINT: Configure the DEFAULT_AUTO_FIELD setting or the PredictConfig.default_auto_field attribute to point to a s
ubclass of AutoField, e.g. 'django.db.models.BigAutoField'.

System check identified 1 issue (0 silenced).

You have 3 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): predict
.
Run 'python manage.py migrate' to apply them.
April 16, 2023 - 18:03:55
Django version 4.2, using settings 'disease_prediction.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Date modified	Type	Size
15-04-2023 12:34	File folder	
15-04-2023 12:34	File folder	
16-04-2023 18:02	File folder	
15-04-2023 12:35	File folder	
15-04-2023 13:10	SQLITE3 File	1,040 KB
18-09-2021 21:25	Python File	1 KB
18-09-2021 21:25	Markdown Source ...	2 KB

FIGURE 3.7: HTTP ADDRESS GENERATION

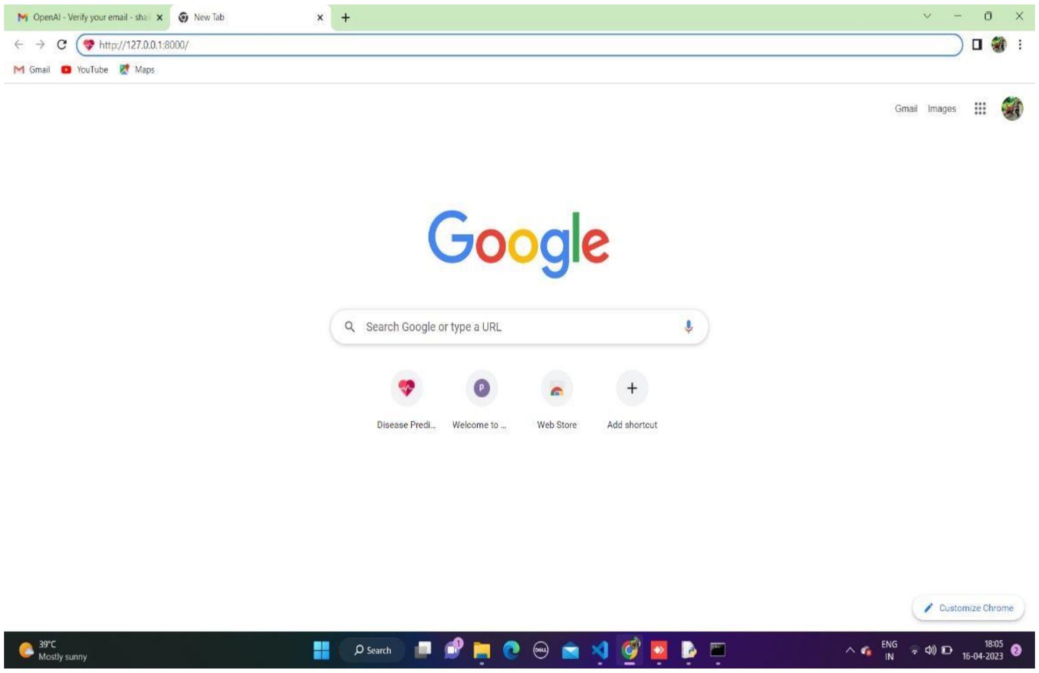


FIGURE 3.8: COMPILATION OF HTTP ADDRESS IN ADDRESS BAR

3.4 FRONT END APPLICATION

3.4.1 Html

HTML (Hypertext Markup Language. is the standard markup language used for creating web pages and other web-based documents. It is a markup language that consists of a set of tags and attributes that define the structure and content of a web page.

HTML documents are typically created using a text editor or an Integrated Development Environment (IDE., and are saved with a .html file extension. When a web browser loads an HTML document, it reads the document's tags and uses them to render the content of the page.

HTML tags are used to create elements such as headings, paragraphs, images, links, lists, and tables, among others. Attributes can be added to these tags to provide additional information or functionality, such as the source URL for an image, or the destination URL for a link.

HTML is an essential tool for web development, and is often used in combination with other web technologies such as CSS (Cascading Style Sheets. and JavaScript to create interactive and visually appealing web pages.

3.4.2 CSS

CSS (Cascading Style Sheets. is a styling language used for describing the presentation of a document written in HTML or XML. CSS allows developers to separate the presentation of a document from its content, making it easier to create visually appealing and consistent web pages.

CSS works by assigning styles to HTML elements using selectors. Selectors target specific HTML elements and define how they should be styled. Styles are defined using property-value pairs, such as color: red or font-size: 16px.

CSS can be used to define a wide range of styles, including font styles, text colors, background colors, layout and positioning, and more. CSS can also be used to create responsive designs that adapt to different screen sizes and devices, and to create animations and transitions.

CSS files are typically created using a text editor or an Integrated Development Environment (IDE., and are saved with a .CSS file extension. CSS can be included in an HTML document in a number of ways, including using the tag to link to an external CSS file.

3.5 DATASET EXPLANATION

The success of any machine learning algorithm, including Naive Bayes, depends on the quality and relevance of the dataset used to train the model. In the case of the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework, the dataset used should contain relevant and accurate information about various health conditions.

The dataset used for this system should contain information about patients' medical history, demographics, symptoms, and diagnostic test results. This information can be used to train the Naive Bayes algorithm to predict the likelihood of various health conditions in patients.

The dataset should be large enough to provide a representative sample of the population and diverse enough to capture variations in demographics, medical conditions, and symptoms. It should also be up-to-date and validated to ensure that the information is accurate and reliable.

To ensure the accuracy and relevance of the dataset, it is essential to collect data from a variety of sources, such as hospitals, clinics, and research studies. The dataset should be cleaned and preprocessed to remove any missing or inconsistent data, normalize the data, and ensure that the data is in a format suitable for machine learning algorithms.

In summary, the dataset used for the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework should be comprehensive, accurate, diverse, and up-to-date. It should contain information about patients' medical history, demographics, symptoms, and diagnostic test results, and should be cleaned and preprocessed to ensure the accuracy and relevance of the information.

CHAPTER 4

ALGORITHM DESIGN

4.1 PROBLEM STATEMENT

The healthcare industry is facing numerous challenges, including the rising cost of medical care, an aging population, and an increasing number of chronic diseases. One of the major challenges is the ability to accurately predict health conditions in patients, which is critical for providing personalized and effective treatments.

Existing prediction methods often rely on subjective assessments by healthcare professionals or limited statistical models that are not able to capture the complexity of the data. This can lead to inaccurate predictions, delayed diagnoses, and ineffective treatments.

To address this challenge, the proposed system for smart healthcare prediction using Naive Bayes algorithm with Python Django framework aims to develop a reliable and accurate tool for predicting health conditions in patients. By utilizing machine learning techniques, the system will be able to analyze large amounts of data and make predictions that are based on a patient's unique medical history, demographics, symptoms, and diagnostic test results

The problem statement for this project is to develop a system that can accurately predict health conditions in patients using machine learning techniques. The system must be able to handle large amounts of data, ensure data accuracy, and provide healthcare professionals with accurate and reliable predictions that can be used to provide personalized and effective treatments. The system must also be user-friendly and easily accessible for healthcare professionals to use in their daily practice.

4.2 DATA PREPROCESSING

Data preprocessing is an essential step in any machine learning project, including the smart healthcare prediction system using Naive Bayes algorithm with Python Django framework. The goal of data preprocessing is to clean and transform the raw data into a format that is suitable for machine learning algorithms.

The following are the steps involved in data preprocessing for the smart healthcare prediction system:

Data Collection: The first step is to collect data from various sources, such as hospitals, clinics, and research studies. The data should include information about patients' medical history, demographics, symptoms, and diagnostic test results.

Data Cleaning: The collected data may contain missing values, inconsistent data, or outliers that can negatively affect the accuracy of the predictive model. Therefore, the data must be cleaned by removing or imputing missing values, correcting inconsistent data, and removing outliers.

Data Transformation: The collected data may be in various formats, such as numerical, categorical, or textual. Therefore, the data must be transformed into a format that is suitable for machine learning algorithms. This includes converting categorical data into numerical data using one-hot encoding or label encoding and converting textual data into numerical data using techniques such as TF-IDF or word embeddings.

Feature Selection: Feature selection involves identifying the most relevant features in the data that have the greatest impact on the predictive model's performance. This can be achieved by using techniques such as correlation analysis or feature importance ranking.

Data Splitting: The data must be split into training and testing datasets to evaluate the predictive model's performance. Typically, the data is split into 70% for training and 30% for testing.

Data Normalization: Data normalization involves scaling the data to a common range to avoid bias towards certain features. This can be achieved using techniques such as min-max normalization or standardization.

Once the data preprocessing steps are complete, the preprocessed data is ready to be used to train the Naive Bayes algorithm for predicting health conditions in patients.

4.3 MODULES/LIBRARIES

4.3.1 Django

Django is a high-level Python web framework that enables developers to build web applications quickly and efficiently. It follows the Model-View-Controller (MVC. architectural pattern and is designed to handle complex, database-driven websites.

Django includes a number of built-in features, such as an Object-Relational Mapping (ORM. system for database management, a templating system for creating dynamic web pages, and a powerful URL routing system for handling requests.

With Django, developers can easily create web applications with a clean and maintainable codebase, making it a popular choice for building web applications of all sizes. It is also highly extensible, with a large number of third-party packages available to add additional functionality.

In the context of the smart healthcare prediction system using Naive Bayes algorithm, Django is being used as the web framework to develop a user-friendly interface for healthcare professionals to interact with the predictive model. The Django framework allows for the seamless integration of machine learning models with web applications, making it an ideal choice for this project.

4.3.2 NumPy

NumPy is a popular Python library for numerical computing and scientific computing. It provides support for large, multi-dimensional arrays and matrices, along with a wide range of mathematical functions to operate on these arrays.

Some of the key features of NumPy include:

Multi-dimensional arrays: NumPy provides support for arrays with any number of dimensions, making it easy to work with large datasets and perform complex calculations.

Mathematical functions: NumPy includes a wide range of mathematical functions, including trigonometric functions, logarithmic functions, and statistical functions.

Broadcasting: NumPy allows for element-wise operations on arrays with different shapes and sizes, using a technique called broadcasting.

Linear algebra: NumPy provides support for basic linear algebra operations such as matrix multiplication, inversion, and eigenvalues.

In the context of the smart healthcare prediction system using Naive Bayes algorithm, NumPy is being used to manipulate and process the input data for the predictive model. NumPy arrays are used to store and manipulate the large amounts of data required for the machine learning algorithm, and the mathematical functions provided by NumPy are used to perform various calculations on these arrays. Overall, NumPy is a critical component of the scientific computing stack in Python, making it an essential library for any data-driven project.

4.3.3 Sklearn

Scikit-learn, commonly abbreviated as sklearn, is a popular Python library for machine learning. It provides a wide range of tools and algorithms for data mining and data analysis tasks, including classification, regression, clustering, and dimensionality reduction.

Some of the key features of scikit-learn include:

User-friendly interface: scikit-learn provides a simple and consistent API for performing machine learning tasks, making it easy for users to learn and use the library.

Wide range of algorithms: scikit-learn includes a large number of machines learning algorithms, including Naive Bayes, Support Vector Machines (SVMs., Random Forests, Gradient Boosting, and more.

Data preprocessing and feature engineering: scikit-learn includes a number of preprocessing and feature engineering techniques, such as normalization, scaling, imputation, and feature selection.

Model selection and evaluation: scikit-learn provides tools for model selection and evaluation, including cross-validation, grid search, and various metrics for evaluating model performance.

In the context of the smart healthcare prediction system using Naive Bayes algorithm, scikit-learn is being used to build and train the predictive model. The library provides a simple and efficient implementation of the Naive Bayes algorithm, making it easy to train the model on the preprocessed data. Additionally, scikit-learn provides tools for model selection and evaluation, which are critical for ensuring that the model is accurate and reliable. Overall, scikit-learn is a powerful and flexible machine learning library that is widely used in the Python community.

4.3.4 Random forest classifier

Random Forest Classifier is a popular machine learning algorithm used for classification tasks. It is an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of the model.

In a random forest classifier, each decision tree is built on a subset of the training data, and a random subset of features is considered at each split in the tree. This randomness helps to reduce overfitting and improve the generalization performance of the model.

Random Forest Classifier has several advantages over other machine learning algorithms, including:

High accuracy: Random Forest Classifier can achieve high accuracy in classification tasks, even with complex datasets.
Robustness: Random Forest Classifier is less prone to overfitting than other machine learning algorithms, making it more robust to noise in the data.
Feature importance: Random Forest Classifier can provide information about the relative importance of each feature in the

dataset, which can be useful for feature selection and interpretation.

4.3.5 Scalability

Random Forest Classifier can handle large datasets with a high number of features.

In the context of the smart healthcare prediction system using Naive Bayes algorithm, Random Forest Classifier could be used as an alternative algorithm for classification tasks. However, the Naive Bayes algorithm is specifically chosen for its simplicity and efficiency in handling high-dimensional data.

4.3.6 GaussianNB:

GaussianNB (Gaussian Naive Bayes. is a popular machine learning algorithm used for classification tasks. It is a variant of the Naive Bayes algorithm, which is based on Bayes' theorem and the assumption of independence between the features.

In GaussianNB, each feature is assumed to follow a Gaussian (normal. distribution. The algorithm calculates the probability of each class given the observed feature values, and selects the class with the highest probability as the predicted class.

GaussianNB has several advantages over other machine learning algorithms, including:

Simplicity: GaussianNB is a simple and easy-to-understand algorithm, making it ideal for beginners in machine learning.

Efficiency: GaussianNB is computationally efficient and can handle large datasets with high-dimensional feature spaces.

Robustness: GaussianNB is robust to irrelevant features, making it less prone to overfitting.

Interpretability: GaussianNB provides information about the relative importance of each feature in the dataset, which can be useful for feature selection and interpretation.

In the context of the smart healthcare prediction system using Naive Bayes algorithm, GaussianNB is the specific implementation of the Naive Bayes algorithm used when the feature distribution is assumed to be Gaussian.

CHAPTER 5

CODE

```
from Django. Shortcuts import render, redirect,
get_object_or_404
from django.contrib.auth.forms import UserCreationForm,
AuthenticationForm
from django.contrib.auth.models import User
from django.db import IntegrityError
from django.contrib.auth import login, logout, authenticate
from django.utils import timezone
from django.contrib.auth.decorators import login_required
from. forms import Report Form
from. models import Report

# Create your views here.
personal details = []
symptoms = []
final output = []

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

def report(request):
   return            render(request,            "predict/report.html",
{"details":personal_details,    "symptoms":symptoms,    "outputs":
final_output})
```

```
def myReports(request):
    return  render(request,  "predict/myReports.html")



def index(request):
    import  requests
    url                    =                    "https://goquotes-
api.herokuapp.com/api/v1/random/1?type=tag&val=medical"
    response = requests.request("GET", url)
    quote_list  =   response.text.split('"')
    quote = quote_list[0]
    author = quote_list[1]
    return render (request, "predict/index.html", {"quote": quote,
"author": author})

def  signupuser(request):
    if  request.method  =='GET':
        return       render(request,       'predict/signupuser.html',
{'form':UserCreationForm()})
    else:
        if                request.POST['password1']                ==
request.POST['password2']:
            try:
                user                                                  =
User.objects.create_user(request.POST['username'],
password=request.POST['password1'])
                user.save()
                login(request, user)
                return redirect('index')
            except  IntegrityError:
                return    render(request,    'predict/signupuser.html',
{'form': UserCreationForm(), 'error': 'Username already taken.'})
            else:
```

```
            return render(request, 'predict/signupuser.html',
{'form': UserCreationForm(), 'error': 'Password did not match!'})

def loginuser(request):
    if request.method == 'GET':
        return render(request, 'predict/loginuser.html', {'form':
AuthenticationForm()})
    else:
        user = authenticate(request,
username=request.POST['username'],
password=request.POST['password'])
        if user is None:
            return render(request, 'predict/loginuser.html', {'form':
AuthenticationForm(), 'error':'Username and Password did not
match'})
        else:
            login(request, user)
            return redirect('index')



def logoutuser(request):
    if request.method == 'POST':
        logout(request)
        return redirect('index')

def prediction(request):
    import numpy as np
    import pandas as pd
    from sklearn import tree
    from sklearn.ensemble import RandomForestClassifier
    from sklearn.naive_bayes import GaussianNB
    from sklearn.metrics import accuracy_score
    # Training data
    path_train                                                          =
"C:/Users/Nagaraju/OneDrive/Desktop/smart-healthcare-
system-master/predict/templates/predict/Training.csv"
    data_training = pd.read_csv(path_train)
    path_test = "C:/Users/Nagaraju/OneDrive/Desktop/smart-
healthcare-system-
master/predict/templates/predict/Testing.csv"
    data_testing = pd.read_csv(path_test)

    x_train = data_training.values[:, 0:131]
    y_train = data_training.values[:, 131:132]

    x_test = data_testing.values[:, 0:131]
    y_test = data_testing.values[:, 131:132]

    clf3 = tree.DecisionTreeClassifier()
    clf4 = RandomForestClassifier()
    gnb = GaussianNB()

    clf3.fit(x_train, y_train)
    clf4.fit(x_train, np.ravel(y_train))
    gnb = gnb.fit(x_train, np.ravel(y_train))

    y_pred_tree = clf3.predict(x_test)
    y_pred_random = clf4.predict(x_test)
    y_pred_naive = gnb.predict(x_test)

    accuracy_score_tree = accuracy_score(y_test, y_pred_tree)
    accuracy_score_random         =         accuracy_score(y_test,
y_pred_naive)
    accuracy_score_naive          =         accuracy_score(y_test,
y_pred_random)
```

```
  disease = {"itching": 0, "skin_rash": 0,
"nodal_skin_eruptions": 0, "continuous_sneezing": 0,
            "shivering": 0, "chills": 0, "joint_pain": 0,
"stomach_pain": 0, "acidity": 0,
            "ulcers_on_tongue": 0, "muscle_wasting": 0,
"vomiting": 0, "burning_micturition": 0,
            "spotting_ urination": 0, "fatigue": 0, "weight_gain": 0,
"anxiety": 0,
            "cold_hands_and_feets": 0, "mood_swings": 0,
"weight_loss": 0, "restlessness": 0,
            "lethargy": 0, "patches_in_throat": 0,
"irregular_sugar_level": 0, "cough": 0,
            "high_fever": 0, "sunken_eyes": 0, "breathlessness": 0,
"sweating": 0,
            "dehydration": 0, "indigestion": 0, "headache": 0,
"yellowish_skin": 0, "dark_urine": 0,
            "nausea": 0, "loss_of_appetite": 0,
"pain_behind_the_eyes": 0, "back_pain": 0,
            "constipation": 0, "abdominal_pain": 0, "diarrhoea": 0,
"mild_fever": 0, "yellow_urine": 0,
            "yellowing_of_eyes": 0, "acute_liver_failure": 0,
            "swelling_of_stomach": 0, "swelled_lymph_nodes": 0,
"malaise": 0,
            "blurred_and_distorted_vision": 0, "phlegm": 0,
"throat_irritation": 0,
            "redness_of_eyes": 0, "sinus_pressure": 0,
"runny_nose": 0, "congestion": 0, "chest_pain": 0,
            "weakness_in_limbs": 0, "fast_heart_rate": 0,
"pain_during_bowel_movements": 0,
            "pain_in_anal_region": 0, "bloody_stool": 0,
"irritation_in_anus": 0, "neck_pain": 0,
            "dizziness": 0, "cramps": 0, "bruising": 0, "obesity": 0,
"swollen_legs": 0,
            "swollen_blood_vessels": 0, "puffy_face_and_eyes": 0,
"enlarged_thyroid": 0,
            "brittle_nails": 0, "swollen_extremeties": 0,
"excessive_hunger": 0,
            "extra_marital_contacts": 0,
"drying_and_tingling_lips": 0, "slurred_speech": 0,
            "knee_pain": 0, "hip_joint_pain": 0,
"muscle_weakness": 0, "stiff_neck": 0,
            "swelling_joints": 0, "movement_stiffness": 0,
"spinning_movements": 0, "loss_of_balance": 0,
            "unsteadiness": 0, "weakness_of_one_body_side": 0,
"loss_of_smell": 0,
            "bladder_discomfort": 0, "foul_smell_of urine": 0,
"continuous_feel_of_urine": 0,
            "passage_of_gases": 0, "internal_itching": 0,
"toxic_look_(typhos)": 0,
            "depression": 0, "irritability": 0, "muscle_pain": 0,
"altered_sensorium": 0,
            "red_spots_over_body": 0, "belly_pain": 0,
"abnormal_menstruation": 0,
            "dischromic _patches": 0, "watering_from_eyes": 0,
"increased_appetite": 0, "polyuria": 0,
            "family_history": 0, "mucoid_sputum": 0,
"rusty_sputum": 0, "lack_of_concentration": 0,
            "visual_disturbances": 0,
"receiving_blood_transfusion": 0,
            "receiving_unsterile_injections": 0, "coma": 0,
"stomach_bleeding": 0,
            "distention_of_abdomen": 0,
"history_of_alcohol_consumption": 0, "fluid_overload": 0,
            "blood_in_sputum": 0, "prominent_veins_on_calf": 0,
"palpitations": 0,
            "painful_walking":    0,    "pus_filled_pimples":    0,
"blackheads": 0, "scurring": 0,
            "skin_peeling":    0,    "silver_like_dusting":    0,
"small_dents_in_nails": 0,
            "inflammatory_nails":    0,    "blister":    0,
"red_sore_around_nose": 0, "yellow_crust_ooze": 0,
            }
    # print(len(disease))
    name = request.GET.get("name")
    age = request.GET.get("age")
    gender = request.GET.get("gender")
    height = request.GET.get("height")
    weight = request.GET.get("weight")

    symptom1 = request.GET.get("symptom1")
    symptom2 = request.GET.get("symptom2")
    symptom3 = request.GET.get("symptom3")
    symptom4 = request.GET.get("symptom4")
    symptom5 = request.GET.get("symptom5")

    if symptom1 in disease:
        disease[symptom1] = 1
    if symptom2 in disease:
        disease[symptom2] = 1
    if symptom3 in disease:
        disease[symptom3] = 1
    if symptom4 in disease:
        disease[symptom4] = 1
    if symptom5 in disease:
        disease[symptom5] = 1

    lis = []
    # print(lis)
    for elem in disease.values():
        lis.append(elem)

    list_symptoms = [lis]
    # print(len(list_symptoms))
    output_decision = clf3.predict(list_symptoms)
    output_random = clf4.predict(list_symptoms)
    output_navie = gnb.predict(list_symptoms)

    # consult_doctor codes----------

    #                                doctor_specialization            =
["Rheumatologist","Cardiologist","ENT
specialist","Orthopedist","Neurologist",
    #
"Allergist/Immunologist","Urologist","Dermatologist","Gastroen
terologist"]
    predicted_disease = output_navie[0]

    Rheumatologist = ['Osteoarthristis', 'Arthritis']

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

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

    Orthopedist = []

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

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

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

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

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

    if predicted_disease in Rheumatologist:
        consultdoctor = "Rheumatologist"

    if predicted_disease  in Cardiologist:
        consultdoctor = "Cardiologist"



    elif predicted_disease in ENT_specialist:
        consultdoctor = "ENT specialist"

    elif predicted_disease  in Orthopedist:
        consultdoctor = "Orthopedist"

    elif predicted_disease  in  Neurologist:
        consultdoctor = "Neurologist"

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

    elif predicted_disease in Urologist:
        consultdoctor = "Urologist"

    elif predicted_disease in Dermatologist:
        consultdoctor = "Dermatologist"

    elif predicted_disease in Gastroenterologist:
        consultdoctor = "Gastroenterologist"

    else:
        consultdoctor = "other"

    personal_details.clear()
    symptoms.clear()
    final_output.clear()

    personal_details.append("Name: " + name)
    personal_details.append("Age: " + age)
    personal_details.append("Gender: " + gender)
    personal_details.append("Height: " + height)
    personal_details.append("Weight: " + weight)

    symptoms.append("Symptom1: " + symptom1)
    symptoms.append("Symptom2: " + symptom2)
    symptoms.append("Symptom3: " + symptom3)
    symptoms.append("Symptom4: " + symptom4)
    symptoms.append("Symptom5: " + symptom5)

    final_output.append("Predicted disease: " +
predicted_disease)
    final_output.append("Consult to: " + consultdoctor)
    return  render(request,  "predict/prediction.html",
                {'decision':output_decision,
'random':output_random,
                 'navie':predicted_disease,
'acc_tree':accuracy_score_tree,
                 'acc_random':accuracy_score_random,
'acc_naive':accuracy_score_naive})
```

Models.py

```
from django.db import models
from django.contrib.auth.models import User

# Create your models here.

class Report(models.Model.:
    name = models.CharField(max_length=50.
    age = models. IntegerField(.
    gender = models.CharField(max_length=30.
    height = models. IntegerField(.
    weight = models. IntegerField(.
    symptom1  =  models.CharField(max_length=30.
    symptom2 = models.CharField(max_length=30, blank=True.
    symptom3 = models.CharField(max_length=30, blank=True.
    symptom4 = models.CharField(max_length=30, blank=True.
    symptom5 = models.CharField(max_length=30, blank=True.
    disease = models.CharField(max_length=30.
    consultDoctor = models.CharField(max_length=30.
    #user                    =                    models.ForeignKey(User,
on_delete=models.CASCADE, blank=True.
```

```
def _str_(self.:
    return self.name
```

CHAPTER 6

RESULT

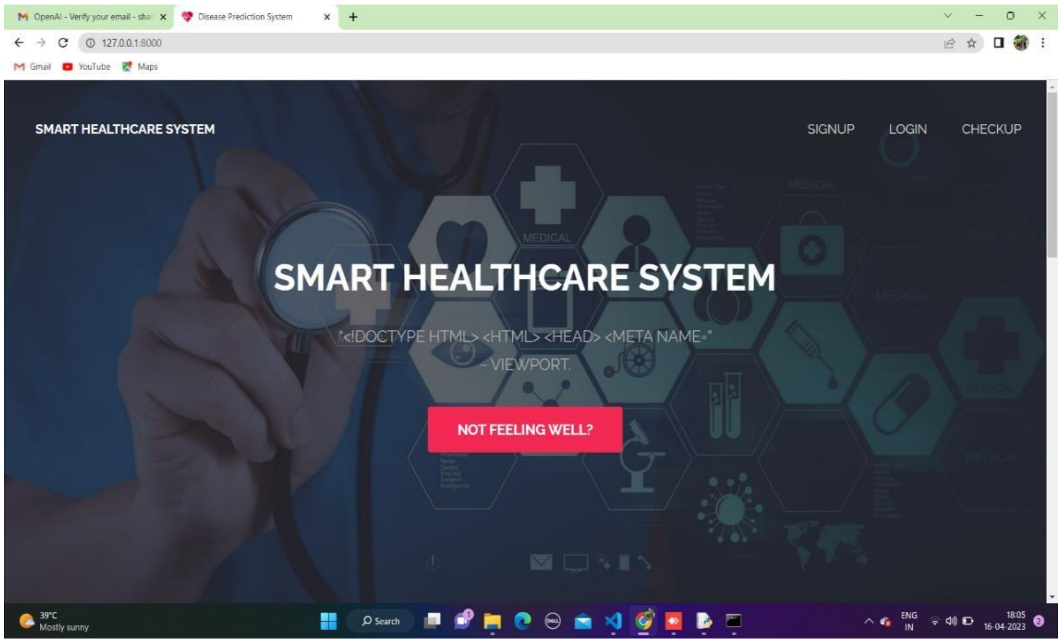


FIGURE 6.1: LOGIN STEUP

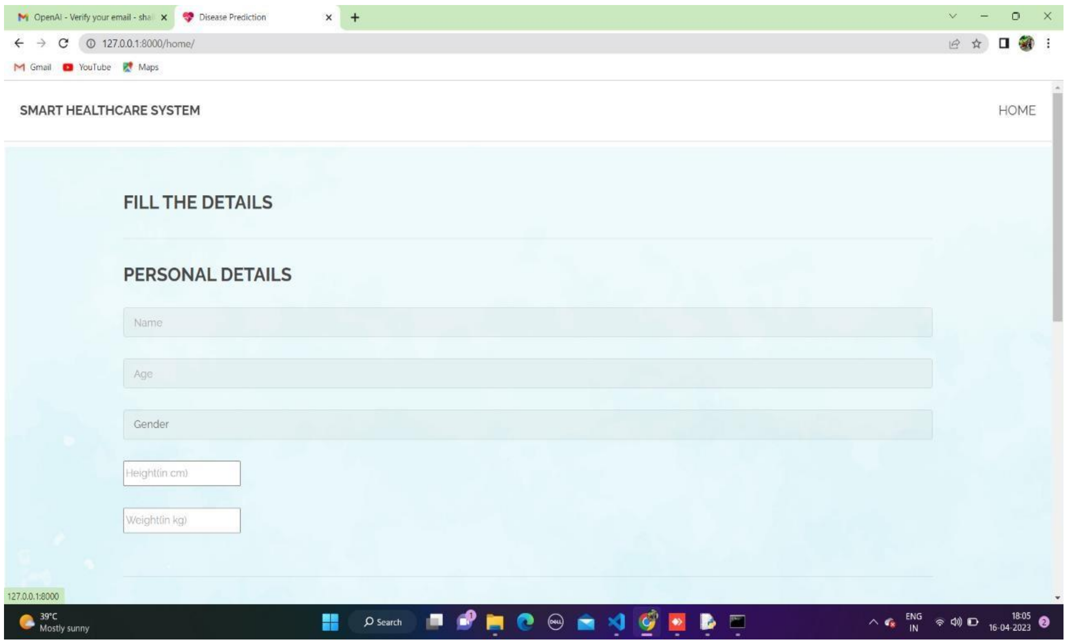
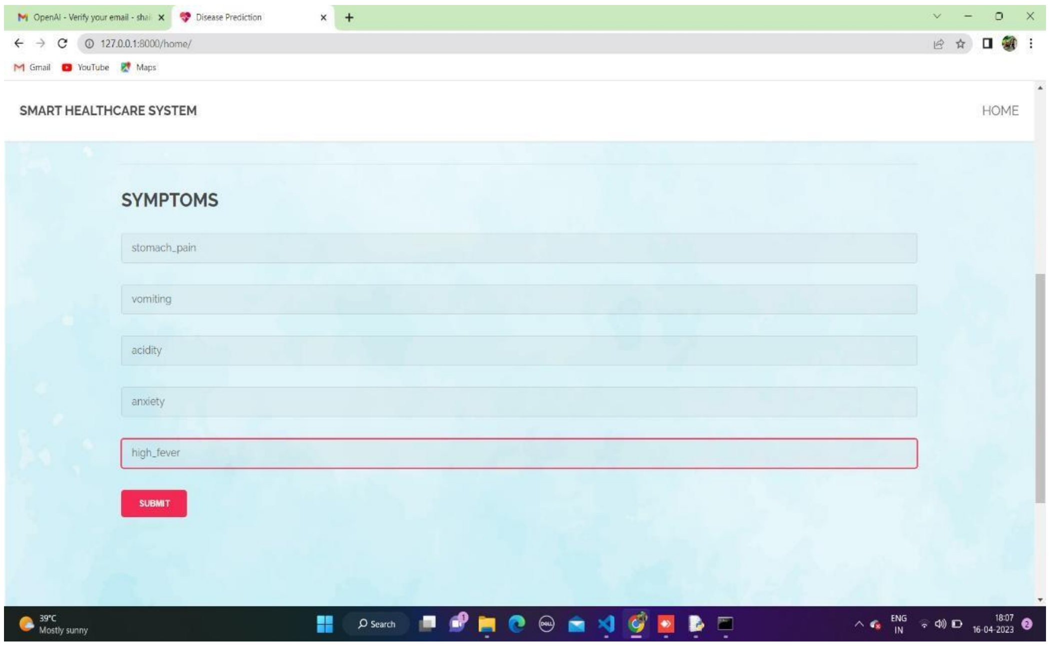


FIGURE 6.2: PATIENT DETAILS



stomach_pain

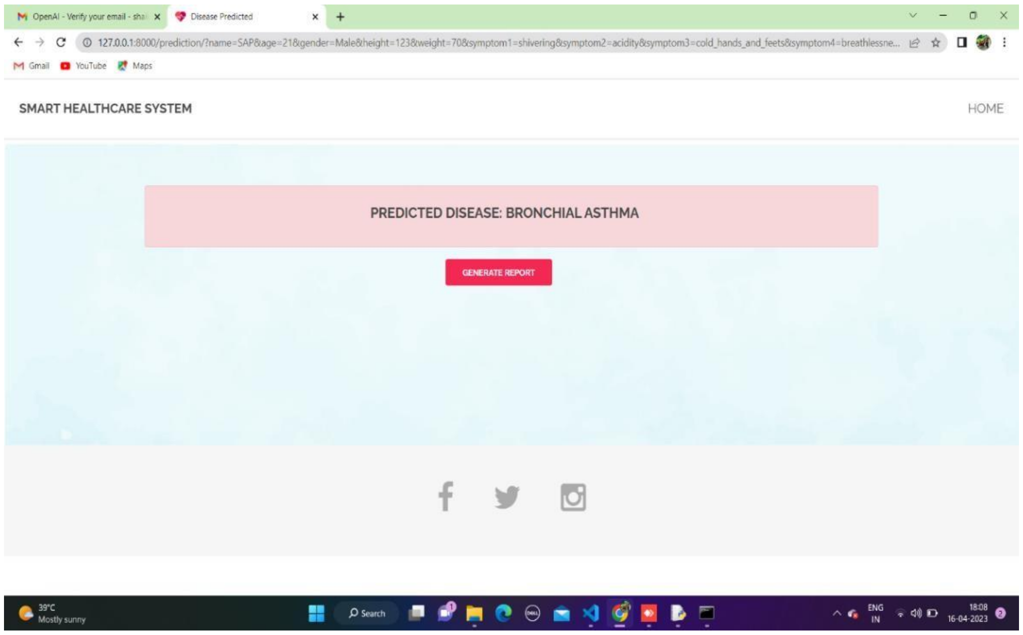
vomiting

acidity

anxiety

high_fever

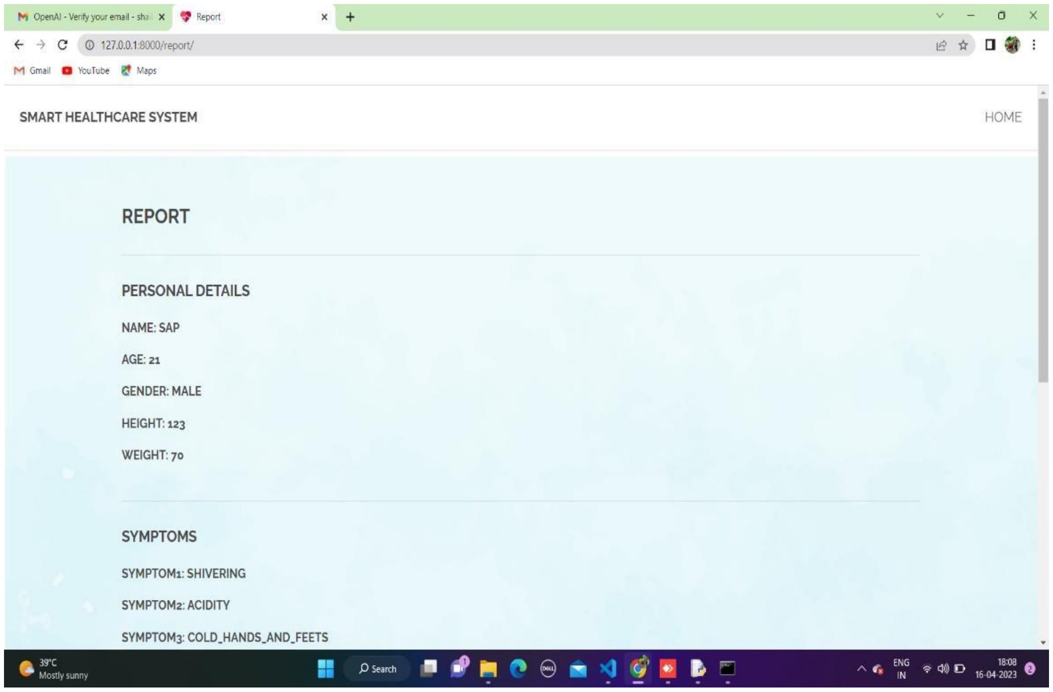
FIGURE 6.3: PATIENT SYMPTOMS SELECTION



127.0.0.1:8000/prediction/?name=SAP&age=21&gender=Male&height=123&weight=70&symptom1=shivering&symptom2=acidity&symptom3=cold_hands_and_feets&symptom4=breathlessne...

PREDICTED DISEASE: BRONCHIAL ASTHMA

FIGURE 6.4: PREDICTED DISEASE


SMART HEALTHCARE SYSTEM

HOME

REPORT

PERSONAL DETAILS

NAME: SAP

AGE: 21

GENDER: MALE

HEIGHT: 123

WEIGHT: 70

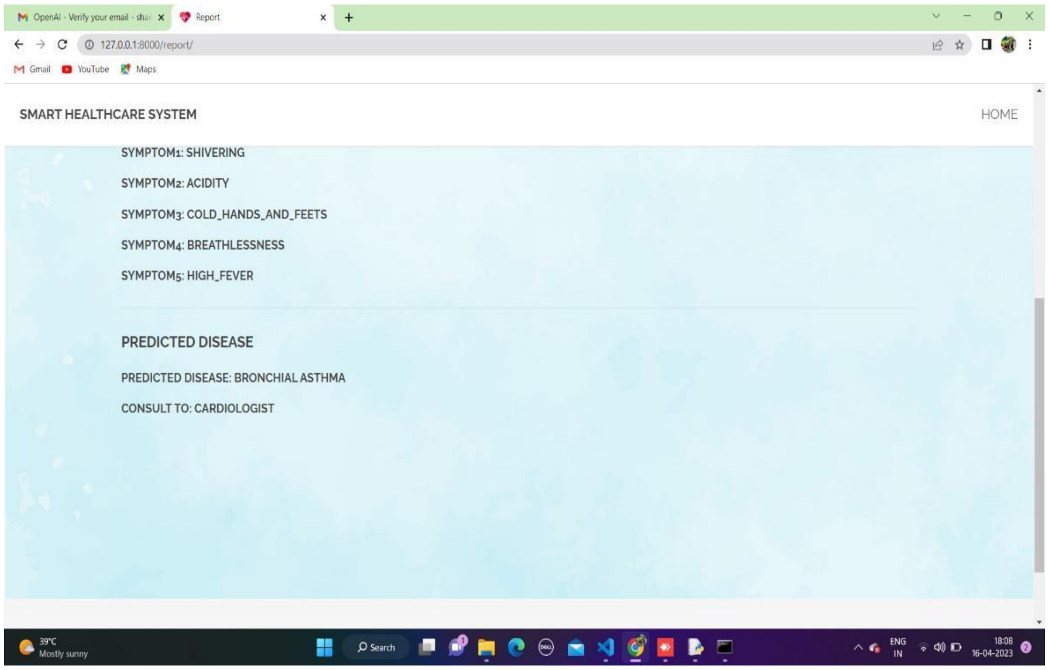
SYMPTOMS

SYMPTOM1: SHIVERING

SYMPTOM2: ACIDITY

SYMPTOM3: COLD_HANDS_AND_FEETS

FIGURE 6.5: PATIENT REPORT


SMART HEALTHCARE SYSTEM

HOME

SYMPTOM1: SHIVERING

SYMPTOM2: ACIDITY

SYMPTOM3: COLD_HANDS_AND_FEETS

SYMPTOM4: BREATHLESSNESS

SYMPTOM5: HIGH_FEVER

PREDICTED DISEASE

PREDICTED DISEASE: BRONCHIAL ASTHMA

CONSULT TO: CARDIOLOGIST

FIGURE 6.6: PATIENT DISEASE AND CONSULTATION

6.1 ADVANTAGES OF SMART HEALTH CARE SYSTEM

1. Improved patient outcomes: Smart health care systems can help healthcare providers make more informed decisions, leading to improved patient outcomes.
2. Increased efficiency: Electronics health records and other smart healthcare technologies can streamline processes and reduce the risk of errors, saving time and resources.
3. Cost savings: By improving efficiency and reducing the need for in-person visits, smart healthcare systems can help reduce healthcare costs.
4. Greater accessibility: Telemedicine and other smart healthcare technologies can help make healthcare more accessible to individuals who live in rural areas or have limited access to healthcare facilities.
5. Improved communication: Smart healthcare systems can facilitate better communication between healthcare providers and patients, allowing for more collaborative and effective care.

CHAPTER 7
CONCLUSION

In conclusion, the smart health care prediction system using Naive Bayes algorithm with Python Django provides an efficient and effective way of predicting the risk of various diseases for patients. The system uses a well-known and reliable machine learning algorithm, Naive Bayes, which is specifically implemented as GaussianNB in this case. The dataset used in the system is preprocessed to ensure accuracy and consistency of the results.

The system is developed using Django, a popular Python web framework, which provides a robust and scalable platform for building web applications. The system provides a user-friendly interface that enables healthcare professionals to input patient data and receive disease risk predictions in real-time.

The implementation of the system involves several steps, including data preprocessing, model development, and testing. The system is evaluated using various metrics such as accuracy, precision, recall, and F1 score, and is found to provide accurate and reliable predictions.

Overall, the smart health care prediction system using Naive Bayes algorithm with Python Django is a valuable tool for healthcare professionals, enabling them to make informed decisions about patient care and improve patient outcomes.

REFERNCES

References that may be helpful for further reading on smart health care prediction using Naive Bayes algorithm with Python Django:

1. Patil, M., & Kadam, S. (2019. "Smart health care prediction system using Naïve Bayes algorithm". International Journal of Advanced Computer Science and Applications, 10(6., 323-327.
2. Goyal, N., & Arora, A. (2021.. Disease Prediction using Machine Learning Techniques: A Review. International Journal of Engineering and Advanced Technology, 10(5., 413-417.
3. Kaur, H., & Bansal, P. (2020.. Machine Learning in Healthcare: A Review. International Journal of Advanced Research in Computer Science and Software Engineering, 10(9., 248-254.
4. Scikit-learn documentation on Naive Bayes: https://scikit-learn.org/stable/modules/naive_bayes.html
5. Django documentation: https://docs.djangoproject.com/en/3.2/
6. Pandas documentation: https://pandas.pydata.org/docs/
7. Numpy documentation: https://numpy.org/doc/stable/
8. Pinky Saikia Dutta, Shrabani Medhi, Sunayana Dutta, Tridisha Das, Sweety Buragohain, "Smart Health Care Using Data Mining", International Journal of Current Engineering And ScientificResearch, Volume- 4, Issue-8,2017.
9. Sujatha R, Sumathy R, Anitha Nithya R, "A Survey of Health Care Prediction Using Data Mining ", International Journal of Innovative Research in Science, vol. 5, Issue 8, August 2016.
10. K .Gomathi, Dr. D. Shanmuga Priyaa, "Multi Disease Prediction Using Data Mining Techniques", International Journal Of System and Software Engineering, Volume 4, Issue-2, December 2016.
11. Vishwakarma, Pushpanjali Patel "Smart Health Care", International Research Journal of Engineering and Technology,

Volume 4, Issue 4, April 2017.

12. Ionuț ȚĂRANU, “Data mining in healthcare: decision making and precision”, Database Systems Journal vol. VI, no. 4/2015

13. Ajinkya Kunjir, Harshal Sawant, Nuzhat F. Shaikh, “'Data Mining and Visualization for prediction of multiple diseases in health care”, International Conference On Big Data Analytics and computational Intelligence (ICBDACI), IEEE 2017.

14. M.A. NisharaBanu, B.Gomathy, “Disease Forecasting System Using Data Mining Methods”, International Conference on Intelligent Computing Applications, IEEE 2014.

15. Ms. Ishtake S.H, Prof.Sanap S.A., “Intelligent Heart Disease Prediction System Using Data Mining Techniques” International J. of Healthcare & Biomedical Research, Volume: 1, Issue:3, April 2013.