Web based application to recognize diseases and providing consultation

Submitted in partial fulfillment of the

requirements for the award of

Bachelor of Engineering degree in
COMPUTER SCIENCE AND ENGINEERING

By

PADAVALA KISHORE CHANDRA SAI

39110723

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

APRIL-2023

SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with Grade "A" by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

<u>BONAFIDE CERTIFICATE</u>

This is to certify that this Project Report is the bonafide work of **PADAVALA KISHORE CHANDRA SAI** who carried out the Project Phase-2 entitled **"WEB BASED APPLICATION TO RECOGNIZE DISEASES AND PROVIDING CONSULTATION"** under our supervision of **Dr. D Menaka , M.E.,Ph.D** from December 2022 to April 2023.

[illegible]

Internal Guide

Dr.D.MENAKA, M.E.,Ph.D

[illegible]

Head of the Department

Dr L. Lakshmanan M.E., Ph.D,

[illegible]

Submitted for Viva-voce Examination held on 24.4.23

[illegible] [illegible]

Internal Examiner **External Examiner**

DECLARATION

I, **PADAVALA KISHORE CHANDRA SAI(39110723)** hereby declare that the Project Report entitled "**WEB BASED APPLICATION TO RECOGNIZES DISEASES AND PROVIDING CONSULTATION"** done by me under the guidance of **Dr.D.MENAKA, M.E.,Ph.D** is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer science and Engineering.

Kishore Chandra

DATE: 24.0423

PLACE: Chennai **SIGNATURE OF THECANDIDATE**

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D**, **Dean**, School of Computing, and **Dr. L.Lakshmanan M.E., Ph.D.,** Head of the Department of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.D.MENAKA,M.E.,Ph.D,** for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my phase-2 project work.

I wish to express my thanks toall Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

Human health is a fundamental component of society, and timely and accurate disease diagnosis is critical to giving patients the care they need. Yet, due to the intricacy of medical research, this necessitates the cooperation of other professions, Mathematics and computer programming are two examples. These established disciplines' are challenged by the emergence by fresh perspectives in medical research, yet the variety of these techniques allows for a thorough assessment of various factors. In this regard, it is suggested to conduct a systematic study of machine learning applications in medical diagnosis of human disorders. The study focuses on current advances in machine learning that can be used to spot interesting patterns, generate precise forecasts, and support decision-making. The study focuses on current advances in machine learning that can be used to spot interesting patterns, generate precise forecasts, and support decision-making. In order to offer proper decision support, this article introduces a special collection of machine learning methods. It seeks to fill the knowledge void in the creation of useful decision support systems for medical applications.

Table of Contents

Chapter No:	Title	Page No:
	ABSTRACT	v
	LIST OF FIGURES	vii
1	**INTRODUCTION**	8
	1.1 About the Project	8
	1.2 Motivation	11
	1.3 Objectives	11
2	**LITERATURE SURVEY**	12
	2.1 Inferences from Literature Survey	16
	2.2 Problems in Existing System	16
3	**REQUIREMENT ANALYSIS**	19
	3.1 Feasibility Studies/Risk Analysis of the Project	20
	3.2 Software Requirements Specification Document	22
	3.3 System Use case	24
4	**DESCRIPTION OF PROPOSED PROJECT**	25
	4.1 Selected Methodology or process model	26
	4.2 Architecture / Overall Design of Proposed System	27
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	29
	4.4 Project Management Plan	29
	4.5 Transition/ Software to Operations Plan	31
5	**IMPLEMENTATION DETAILS**	33
	5.1 Development and Deployment Setup	34
	5.2 Algorithms	35
	5.3 Testing	37
6	**RESULTS AND DISCUSSION**	39
7	**CONCLUSION**	41
	REFERENCES	43
	APPENDIX	45
	A. SOURCE CODE	45
	B. SCREENSHOTS	75
	C. RESEARCH PAPER	79

List of Figures

Figure No:	Figure Name	Page No:
1	System Architecture	28
2	Support Vector Machine	36
3	First Window	39
4	Model Finder	40
5	Hospital Finder	40
6	Data Processing	75
7	Data Preprocessing	75
8	Heat Map	76
9	Model Building	76
10	Testing	77
11	Pickle File	77
12	Analysis Report of Proposed model	78

CHAPTER 1

INTRODUCTION

1.1 About the Project :

Chronic diseases, such as diabetes, hypertension, and cardiovascular disease, are a growing global health concern. According to the World Health Organization, chronic diseases are the leading cause of death worldwide, accounting for 71% of all deaths. In addition to the human toll, chronic diseases also impose a significant economic burden, with estimated global healthcare costs exceeding $7 trillion annually. Therefore, there is an urgent need to develop effective strategies to prevent and manage chronic diseases. Machine learning, a subset of artificial intelligence, has the potential to transform healthcare delivery by leveraging patient data to predict disease occurrence and progression. Machine learning algorithms can identify patterns and relationships in large datasets that traditional statistical methods may miss. These algorithms can then use these patterns to predict patient outcomes, including the likelihood of developing specific diseases. Most studies on machine learning-based disease prediction focus on predicting a single disease. However, patients often have multiple chronic conditions, making it essential to develop models that can predict the occurrence of multiple diseases simultaneously. Such multi-disease prediction models can provide a more comprehensive understanding of a patient's overall health status, allowing healthcare providers to prioritize preventive interventions and personalized treatment plans. In recent years, there has been growing interest in developing multi-disease prediction models using machine learning algorithms. These models use patient data, such as demographics, lifestyle factors, and medical history, to predict the occurrence of multiple chronic diseases. The goal of these models is to identify individuals at high risk for multiple diseases and provide personalized interventions to prevent or manage these conditions. In this report, we review the literature on multi-disease prediction using machine learning algorithms. We first provide an overview of the different types of machine learning algorithms commonly used in disease prediction. We then discuss the challenges and opportunities in developing multi-disease prediction models, including the need for large and diverse datasets, model interpretability, and ethical considerations. Finally, we present a case study on multi-disease prediction using a random forest algorithm and discuss the implications and future directions of this research. Machine learning algorithms have been used extensively in healthcare for various tasks such as medical imaging, drug discovery, and disease diagnosis. However, most of these studies focus on a single disease or a specific healthcare task. The development of multi-disease prediction models is a relatively new area of research that holds significant promise in improving population health outcomes. One of the main advantages of multi-disease prediction models is that they can identify individuals at high risk for multiple chronic diseases. This information can be used to prioritize interventions and improve disease management. For example, a patient who is at high risk for both diabetes and hypertension can receive personalized interventions that target both conditions simultaneously, rather than separate interventions for each condition. However, developing multidisease prediction models is not without challenges. One of the main challenges is the need for large and diverse datasets. Most studies on multi-disease prediction use datasets from a single healthcare system or a specific population, which may limit the generalizability of the models. Therefore, there is a need for larger and more diverse datasets that capture a range of demographic, social, and environmental factors. Another challenge in developing multi-disease prediction models is model interpretability. Machine learning algorithms are often seen as black boxes, making it difficult for healthcare providers to understand how the models arrive at their predictions. Interpretability is essential for building trust in the models and ensuring that they are used ethically and effectively. In addition to technical challenges, developing multi-disease prediction models also raises ethical and social considerations. For example, there is a risk that these models may perpetuate health disparities by disproportionately identifying individuals from marginalized communities as high risk. Therefore, there is a need to ensure that these models are used in an equitable and ethical manner. Despite these challenges, the development of multi-disease prediction models using machine learning algorithms has significant potential to transform healthcare delivery and improve health outcomes. In the next sections of this paper, we will review the different types of machine learning algorithms commonly used in disease prediction and discuss the challenges and opportunities in developing multi-disease prediction models. We will also present a case study on multi-disease prediction using a random forest algorithm and discuss the implications and future directions of this research. By identifying individuals at high risk for multiple chronic diseases, these models can help healthcare providers and policymakers allocate resources more effectively and prioritize preventive interventions. Moreover, the development of multi-disease prediction models is particularly relevant in the context of the COVID-19 pandemic. The pandemic has highlighted the importance of understanding how different chronic conditions interact and contribute to poor health outcomes. 10 Studies have shown that individuals with pre-existing chronic conditions, such as diabetes, hypertension, and cardiovascular disease, are at higher risk of severe illness and mortality from COVID-19. Therefore, multi-disease prediction models can help identify individuals who are at higher risk of both chronic diseases and COVID-19, enabling healthcare providers to provide more targeted interventions and prevent adverse health outcomes. Another important consideration in the development of multi-disease prediction models is the integration of patient-generated health data (PGHD). PGHD includes data that patients collect on their health status, such as blood pressure readings, physical activity, and dietary intake. Integrating PGHD into multi-disease prediction models can improve the accuracy of the models and provide a more comprehensive understanding of patient's health status. Finally, the development of multi-disease prediction models requires interdisciplinary collaboration between healthcare providers, data scientists, and policymakers. Effective collaboration is essential for developing models that are clinically relevant, ethically sound, and technically feasible. Furthermore, collaboration is critical for ensuring that the models are implemented in real-world clinical settings in a way that benefits patients and improves health outcomes. In summary, the development of multi-disease prediction models using machine learning algorithms has significant potential to improve population health outcomes by identifying individuals at high risk for multiple chronic diseases. However, developing these models requires addressing technical, ethical, and social challenges and requires interdisciplinary collaboration between different stakeholders. In the following sections of this paper, we will delve deeper into the technical aspects of multi-disease prediction models, discuss the challenges and opportunities, and present a case study to illustrate the potential of these models in improving health outcomes.

1.2 Motivation :

In our present daily life , all the people in the world are just going to the hospital for checking their body conditions and health of their body. If a person is affected with disease whether it is harmful or mild-symptom they are getting fear and just going into the depression , why this is because, they didn’t get the correct or exact answer or solution for that disease. I’m doing this project because, i also got experienced by above conditions in past. This project gives the exact or nearest disease with good accuracy . You can trust and commit with this project upto 70% and then go to consult the doctor. This can save your money, time and all.

1.3 Objective :

Nowadays virtual assistant is playing a very important role in our daily activities and has become an inseparable part of our lives . In this Project, I have planned to do a Mobile Hospital. which is for free of cost, proper diagnosis to the patient, easy to access by anyone from anywhere and recommend the patient to go hospital or not.

CHAPTER 2
LITERATURE SURVEY

The literature survey consists of first four papers surveyed from medical background. They are studied to establish the importance of various features and their typical values. Being from medical background, no machine learning algorithms is applied for the disease detection and hence are not included in the survey. The literature survey after these four papers is from engineering background where different features are considered and various machine learning algorithms are applied for automatic detection of presence of disease. This problem statement has been extensively studied over the past few years by researchers and automotive companies in a bid to create a solution, and all their solutions vary from analyzing various types of algorithm used to detect the disease in a human body in different ways.

1. Common Diseases by A. Gavhane.

Proposed a ML based system that predicts common diseases. The symptoms dataset was imported from the UCI ML depository, where it contained symptoms of many common diseases. The system used CNN and KNN as classification techniques to achieve multiple diseases prediction. Moreover, the proposed solution was supplemented with more information that concerned the living habits of the tested patient, which proved to be helpful in understanding the level of risk attached to the predicted disease. Dahiwade et al. [9] compared the results between KNN and CNN algorithm in terms of processing time and accuracy. The accuracy and processing time of CNN were 84.5% and 11.1 seconds, respectively. The statistics proved that KNN algorithm is under performing compared to CNN algorithm. In light of this study, the findings also agreed that CNN outperformed typical supervised algorithms such as KNN, NB, and DT. The authors concluded that the proposed model scored higher in terms of accuracy, which is explained by the capability of the model to detect complex nonlinear relationships in the feature space. Moreover, CNN detects features with high importance that renders better description of the disease, which enables it to accurately predict diseases with high complexity [9], [10]. This conclusion is well supported and backed with empirical observations and statistical arguments. Nonetheless, the presented models lacked details, for instance, Neural Networks parameters such as network size, architecture type, learning rate and back propagation algorithm, etc. In addition, the analysis of the performances is only evaluated in terms of accuracy, which debunks the validity of the presented findings [9]. Moreover, the authors did not take into consideration the bias problem that is faced by the tested algorithms [9], [10]. In illustration, the incorporation of more feature variables could immensely ameliorate the performance metrics of under performed algorithms

2 .Detection of abnormality in spinal Cord by author S. Shyni Carmel .

Research on Medical Image proposes an efficient platform for automatic analysis and detection of any Deformations in a given medical image data set especially in Spinal Cord for an effective and better understanding of diagnosis. The abnormality of the spinal cord may include Tumor. Disc a hernia, Fracture, swelling etc., which has been detected from any given modality of Medical images such as , CT, and I etc. In this work, Automated Decision support system is introduced for fast and accurate analysis which will help to confirm the existence of affected part of the Spinal Cord MR image. It has two phases. Phase 1:[12] Identifying any anomaly features or distortion is found to have existed in the given image or not by using histograms. Phase II: Involves in Clustering of the image which is used to find the depth of the existence of the calcification in the Spinal Image. The performance of the algorithm and the time taken to complete every cluster phase is analyzed. Further, the algorithm's efficiency is being observed to prove that it gives a perfect accuracy.

3. Detection of muscle pains by Azian Azamimi Abdullah.

Lower back pain can be caused by many complications with any parts of the body in the lumbar spine. The compilation of a medical diagnosis is crucial to the medical practitioners in order for them to give a convenient treatment for the low back pain. The machine learning models that applied in the medical field for disease diagnosis assists medical experts in the diseases identification based on the symptoms at an early stage. This research aims to identify the most significant physical parameters that contribute to spinal abnormalities and also predict spinal abnormalities based on collected physical spine data by using unsupervised machine learning approaches such as Principal Component Analysis (PCA), and also using supervised machine learning approaches such as K-Nearest Neighbors (KNN) and Random Forest (RF). As a result, degree spondylolisthesis is the most significant parameter that contributes to spinal abnormalities. As a comparison of results between RF classifier and KNN classifier, KNN classifier performed better than RF classifier since the percentage of accuracy of KNN algorithm (85.32%) are higher compared to RF classifier (79.57%).

4. Kidney Diseases by Serek.

planned a comparative study of classifiers performance for Chronic Kidney disease (CKD) detection using The Kidney Function Test (KFT) dataset. In this study, the classifiers used are KNN, NB, and RF classifier; their performance is examined in terms of F-measure, precision, and accuracy. As per analysis, RF scored better in phrases of F-measure and accuracy, while NB yielded better precision. In consideration of this study, Vijayarani [13] aimed to detect kidney diseases using SVM and NB. The classifiers were used to identify four types of kidney diseases namely Acute Nephritic Syndrome, Acute Renal Failure, Chronic Glomerulonephritis, and CKD. Additionally, the research was focused on determining the better performing classification algorithm based on the accuracy and execution time. From the results, SVM considerably achieved higher accuracy than NB, which makes it the better performing algorithm. However, NB classified data with minimum execution time. Other several empirical studies also focused on locating CKD; Charleonnan et al. [14] and Kotturu et al. [15] concluded that the SVM classifier is the most adequate for kidney diseases because it deals well with semi-structured and unstructured data. Such flexibility allowed SVM to handle larger features spaces, which resulted in acquiring high accuracy when detecting complex kidney diseases. Although supported by findings, the conclusion is weakened by prior suggestion that different hyper-parameters were not experimented when evaluating the performances of ML algorithms. According to Uddin [3] the exploration of the hyper-parameter space can generate different accuracy results and render better performances for ML algorithms.

5. Heart Diseases by Marimuthu

He aimed to predict heart diseases using supervised ML techniques. The authors structured the attributes of data as gender, age, chest pain, gender, target and slope .The applied ML algorithms that were deployed are DT, KNN, LR and NB. As per analysis, the LR algorithm gave a high accuracy of 86.89%, which deemed to be the most effective compared to the other mentioned algorithms. In 2020 he attempted to add more precision to the prediction of heart diseases by accounting for additional parameters such as Resting blood pressure, Serum Cholesterol in mg/dl, and Maximum Heart Rate achieved. The used dataset was imported from the UCI ML laboratory; it was comprised with 120 samples that were heart disease positive, and 150 samples that were heart disease negative. Dwivedi attempted to evaluate the performance of Artificial Neural Networks (ANN), SVM, KNN, NB, LR and Classification Tree. At the appliance of tenfold cross validation, the results showed that LR has the highest classification accuracy and sensitivity, which

shows high dependability at detecting diseases [9]. This conclusion is strengthened by the findings of some diseases, where the Logistic Regression outperformed other techniques such as ANN, SVM, and Adaboost. The studies excelled in conducting an extensive analysis on the ML models. For instance, various hyper-parameters were tested at each ML algorithm to converge to the best possible accuracy and precision values. Despite that advantage, the small size of the imported datasets constraints the learning models from targeting diseases with higher accuracy and precision.

2.1 INFERENCES FROM LITERATURE SURVEY :

From the Literature survey I have done, I came to an idea of develop in fully automated to predict the any disease and suggesting the required medication model based on Machine learning algorithms for the aquired disease which have been raisen from the symptoms and given quires by user. So now by the use of different algorithmic techniques such as AdaBoost, Support Vector Machine etc. I going to create an application for instant prediction of any disease.

2.2 OPEN PROBLEMS IN EXISTING SYSTEM :

Based on the literature survey, some of the open problems in the existing system of multi-disease prediction using machine learning algorithms include: 1. Data quality and integration: The quality of the data used in machine learning models can greatly impact their accuracy and effectiveness. Challenges in data quality, such as missing data and errors, can lead to biased and inaccurate predictions. Additionally, integrating data from diverse sources, such as electronic health records and patient-generated data, can be complex and require innovative approaches.

2. Interpretability and transparency: Machine learning models can be complex and difficult to interpret, which can limit their clinical adoption and acceptance. Developing approaches to improve the interpretability and transparency of multi-disease prediction models is critical to ensure their effective use in healthcare.

3. Generalizability and scalability: The majority of the studies reviewed in the literature survey were conducted using data from US health systems, which may limit the generalizability of the findings to other healthcare settings and populations. Additionally, developing multi-disease prediction models that are scalable to large populations and healthcare systems can be challenging.

4. Ethical and social considerations: Multi-disease prediction models raise important ethical and social considerations, such as privacy, equity, and potential unintended consequences. Developing strategies to address these issues is necessary to ensure the responsible use of these models.

5. Clinical adoption and implementation: Developing effective strategies for implementing multi disease prediction models in clinical practice is critical to ensure their impact on population health outcomes. Ensuring the usability and acceptability of these models by healthcare providers and patients is also important.

6. Evaluation and validation: Evaluating the accuracy and effectiveness of multi-disease prediction models is critical to ensure their clinical and societal impact. However, developing standardized approaches for evaluating and validating these models can be challenging.

7. Dynamic and personalized prediction: Multi-disease prediction models often assume a static relationship between the input features and the outcome. However, diseases can evolve over time, and individuals have unique health trajectories. Developing dynamic and personalized multi-disease prediction models that can adapt to changing health status and individual needs is a promising area of research.

8. Integration with clinical decision-making: Multi-disease prediction models have the potential to support clinical decision-making and improve patient outcomes. However, developing effective strategies for integrating these models into clinical workflows and decision-making processes is challenging. Addressing these challenges requires

collaboration between machine learning experts and healthcare providers.

9. Human-machine collaboration: Multi-disease prediction models can complement and augment human expertise in healthcare, but developing effective human-machine collaboration approaches is critical. Ensuring that healthcare providers and patients understand and trust the predictions made by these models is important for their adoption and use.

10.Long-term outcomes: Multi-disease prediction models have the potential to improve long-term health outcomes by identifying patients who may benefit from early interventions or personalized treatment plans. However, evaluating the impact of these models on long-term outcomes, such as morbidity and mortality, requires long-term follow-up and rigorous evaluation methods. Addressing these open problems is essential to advancing the field of multi-disease prediction using machine learning algorithms and ensuring their responsible and effective use in healthcare.

CHAPTER 3

REQUIREMENT ANALYSIS

Requirements analysis is a critical step in the development of any system, including multi-disease prediction using machine learning algorithms. Based on the literature survey and the identified open problems, the following requirements can be identified for developing effective multi-disease prediction models:

1. Data collection and integration: To develop accurate multi-disease prediction models, highquality data from diverse sources must be collected and integrated. This requires developing strategies for data collection and integration that address issues such as missing data and errors.

2. Model development and evaluation: Developing effective multi-disease prediction models requires expertise in machine learning algorithms and model development. Additionally, developing standardized approaches for evaluating and validating these models is necessary to ensure their accuracy and effectiveness.

3. Interpretability and transparency: To ensure the clinical adoption and acceptance of multidisease prediction models, it is important to develop strategies for improving their interpretability and transparency. This requires developing approaches for explaining the predictions made by the models in a way that is understandable and useful for healthcare providers and patients. 4. Generalizability and scalability: Multi-disease prediction models must be developed in a way that is generalizable to diverse healthcare settings and populations. Additionally, developing models that can be scaled to large populations and healthcare systems is critical to ensure their impact on population health outcomes.

5. Ethical and social considerations: Multi-disease prediction models raise important ethical and social considerations, such as privacy, equity, and potential unintended consequences. Developing strategies to address these issues is necessary to ensure the responsible use of these models.

6. Clinical adoption and implementation: Developing effective strategies for implementing multidisease prediction models in clinical practice is critical to ensure their impact on patient outcomes. Ensuring the usability and acceptability of these models by healthcare providers and patients is also important.

7. Long-term outcomes: Evaluating the impact of multi-disease prediction models on long-term health outcomes, such as morbidity and mortality, requires long-term follow-up and rigorous evaluation methods. Addressing these requirements requires collaboration between machine learning experts, healthcare providers, and patients. Additionally, developing innovative approaches to data collection, model development, and evaluation is necessary to ensure the effectiveness and impact of multi-disease prediction models in healthcare.

3.1 Feasibility Studies/Risk Analysis of the Project:

Feasibility studies and risk analysis are important steps in determining the viability of a project and identifying potential challenges and risks. In the context of multi-disease prediction using machine learning algorithms, the following feasibility studies and risk analysis can be conducted: Feasibility Studies:

1. Technical feasibility: The technical feasibility of the project can be assessed by determining the availability of the required technology and expertise. The project requires expertise in machine learning algorithms and data analysis, as well as access to high-quality data.

2. Economic feasibility: The economic feasibility of the project can be assessed by evaluating the cost of developing and implementing the multi-disease prediction models. The project may require significant investment in data collection and model development, as well as ongoing maintenance and updates.

3. Operational feasibility: The operational feasibility of the project can be assessed by evaluating the impact of the multi-disease prediction models on clinical workflows and patient outcomes. Ensuring the usability and acceptability of the models by healthcare providers and patients is important for their adoption and use.

Risk Analysis:

1. Data quality and availability: The quality and availability of data is critical for developing accurate multi-disease prediction models. Risks associated with data quality and availability include missing data, errors, and bias. Addressing these risks

requires developing effective strategies for data collection and integration that address these issues.

2. Model accuracy and generalizability: Developing accurate and generalizable multi-disease prediction models is challenging. Risks associated with model accuracy and generalizability include overfitting and bias. Addressing these risks requires developing standardized approaches for model development and evaluation.

3. Ethical and social considerations: Multi-disease prediction models raise important ethical and social considerations, such as privacy, equity, and potential unintended consequences. Risks associated with ethical and social considerations include the potential for unintended harm and unintended consequences. Addressing these risks requires developing strategies to address these issues and ensure the responsible use of these models.

4. Clinical adoption and implementation: Ensuring the clinical adoption and implementation of multi-disease prediction models is critical for their impact on patient outcomes. Risks associated with clinical adoption and implementation include the potential for resistance from healthcare providers and patients, as well as challenges in integrating the models into clinical workflows. Addressing these risks requires developing effective strategies for implementation and ensuring the usability and acceptability of these models.

3.2 Software Requirements Specification Document :

The Software Requirements Specification (SRS) document for the multi-disease prediction system outlines the functional and non-functional requirements for the software. The SRS document provides a detailed description of the system, its features, and its functionalities. The following are the sections of the SRS document for the multi-disease prediction system:

Introduction: This section provides an overview of the multi-disease prediction system and its purpose. It includes the background of the project, its scope, and the intended audience. It also includes the objectives of the system and the benefits it will provide.

Functional Requirements: This section lists all the functional requirements of the system. Functional requirements describe the specific features and functions that the system must perform. The functional requirements for the multi-disease prediction system include: The system must be able to process and analyze medical data from multiple sources The system must be able to predict the likelihood of multiple diseases based on the medical data The system must be able to generate reports and visualizations based on the predictions The system must be able to continuously learn and adapt to new data.

Non-Functional Requirements: This section lists all the non-functional requirements of the system. Non-functional requirements describe the system's characteristics that are not related to its specific functions. The non-functional requirements for the multi-disease prediction system include: The system must be secure and protect the confidentiality of patient data The system must be scalable and able to handle large amounts of data The system must be user-friendly and easy to use for healthcare professionals The system must be reliable and have high availability.

Performance Requirements: This section lists the performance requirements of the system. Performance requirements describe the system's ability to perform its functions under specific conditions. The performance requirements for the multi-disease prediction system include: The system must be able to process medical data in real-time The system must be able to handle a large number of simultaneous requests The system must be able to generate predictions within a reasonable time frame.

Design Constraints: This section lists the design constraints that must be considered when developing the system. Design constraints describe the limitations on the system's design and development. The design constraints for the multi-disease prediction system include:

System Architecture: This section provides an overview of the system architecture, including the hardware and software components, and how they interact with each other. It also includes any third-party services that the system relies on.

System Requirements: This section provides a detailed description of the system requirements, including the hardware and software requirements, installation and configuration requirements, and any other system requirements.

Acceptance Criteria: This section lists the acceptance criteria for the system, including the criteria that must be met for the system to be considered acceptable and ready for deployment. Overall, the SRS document for the multi-disease prediction system is a comprehensive document that outlines the functional and non-functional requirements for the software.

3.3 System Use Case :

The system use case for the multi-disease prediction system describes the interactions between the users and the system. Use cases are used to capture the requirements of the system from the perspective of the end users. The following are the use cases for the multi-disease prediction system.

placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

1. User Login: This use case describes the process by which a user logs into the system. The user must enter their username and password to gain access to the system.

2. Data Input: This use case describes the process by which medical data is input into the system. The user must enter the patient's data into the system, including their medical history, symptoms, and any other relevant information. 3. Data Processing: This use case describes the process by which the system processes and analyzes the medical data. The system uses machine learning algorithms to predict the likelihood of multiple diseases based on medical data.

4. Prediction Generation: This use case describes the process by which the system generates predictions based on medical data. The system generates predictions for each disease and provides the probability of each disease.

5. Report Generation: This use case describes the process by which the system generates reports based on the predictions. The system generates reports that provide a summary of the predictions and any relevant information.

6. Visualization Generation: This use case describes the system's process of generating visualizations based on predictions. The system generates visualizations that provide a graphical representation of the predictions.

7. System Administration: This use case describes the process by which the system is administered. The system administrator can access the system's settings and configure the system's parameters.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

The proposed multi-disease prediction system is a software application that leverages machine learning algorithms to predict the likelihood of multiple diseases based on medical data input by healthcare professionals. The system will provide a user-friendly interface for healthcare professionals to input patient data and receive predictions for multiple diseases. The system will also generate reports and visualizations to help healthcare professionals understand the predictions and communicate them to patients.

The key features of the proposed system include:

1. User Authentication: The system will provide a login screen to ensure that only authorized healthcare professionals have access to patient data and predictions.
2. Data Input: Healthcare professionals will be able to input patient data into the system using a user-friendly interface. The data will include medical history, symptoms, and any other relevant information.
3. Data Processing: The system will use machine learning algorithms to process and analyze the medical data input by healthcare professionals.
4. Prediction Generation: The system will generate predictions for multiple diseases based on the medical data input by healthcare professionals. The predictions will include the probability of each disease.
5. Report Generation: The system will generate reports based on the predictions. The reports will provide a summary of the predictions and any relevant information.
6. Visualization Generation: The system will generate visualizations based on the predictions. The visualizations will provide a graphical representation of the predictions.
7. System Administration: The system will have an administration module that will allow system administrators to manage the system's settings and parameters. 26 The proposed system will provide several benefits to healthcare professionals, including improved accuracy and speed of disease diagnosis, reduction in manual error, and improved patient outcomes. The system will also provide benefits to patients by allowing for faster and more accurate diagnoses, which can lead to improved treatment options and better health outcomes. The proposed multi-disease prediction

system has the potential to significantly improve the efficiency and accuracy of disease diagnosis and treatment, leading to better health outcomes for patients.

4.1 Selected Methodology or process model:

The selected methodology for developing the multi-disease prediction system is the Agile methodology. The Agile methodology is a popular and effective approach for software development that emphasizes iterative and incremental development, collaboration, and rapid feedback. The Agile methodology is well-suited for complex projects that require frequent changes and updates, such as the multi-disease prediction system. The Agile methodology will enable the development team to work closely with healthcare professionals to ensure that the system meets their needs and requirements. The Agile methodology also emphasizes frequent testing and feedback, which will ensure that the system is reliable and accurate. The Agile methodology involves several key practices, including:

1. User Stories: The development team will work with healthcare professionals to identify and prioritise the features and functionalities that are most important to them. These features will be documented as user stories, which will serve as the basis for development.
2. Sprints: The development process will be broken down into sprints, which are typically two to four weeks long. Each sprint will focus on a specific set of user stories.
3. Daily Stand-Ups: The development team will hold daily stand-up meetings to discuss progress, challenges, and plans for the day. These meetings will ensure that everyone is on the same page and that any issues are addressed promptly.
4. Sprint Reviews: At the end of each sprint, the development team will hold a sprint review meeting with healthcare professionals to demonstrate the completed features and gather feedback.
5. Sprint Retrospectives: After each sprint, the development team will hold a retrospective meeting to discuss what went well, what didn't go well, and what can be improved in the next sprint.

By using the Agile methodology, the development team will be able to work closely with healthcare professionals to ensure that the multi-disease prediction system meets their needs and requirements. The Agile methodology will also enable the team to respond quickly to changes and feedback, resulting in a more effective and efficient development process.

Looking at the disadvantages of all the above methodologies used in previous systems, the most common point was there is a problem while deciding which algorithm is best. For one dataset if one algorithm gives the higher accuracy for another dataset another dataset another algorithm gives higher accuracy. So, I came up with a solution to eliminate entering the dataset manually and testing them. Here we are going to take the values for the attributes from the user(patient)and predict whether the patient is likely to have any disease or not. So this may help them to rescue their lives and also take medications according to the disease. As an extra feature we are also going to add an artificial intelligence chatbot for best medication and best diagnosis to user.

4.2 Architecture / Overall Design of Proposed System :

The proposed multi-disease prediction system in Python will be designed using a modular architecture that consists of different components responsible for specific tasks. The data preprocessing and data segmentation will under go in the background. So using of this application can make people aware of what is the next step they have to take. As an extra feature we are also going to add an Artificial intelligence chatbot to suggest good medication and a diagnosis for the disease. The overall design of the proposed system can be represented using the following diagram:

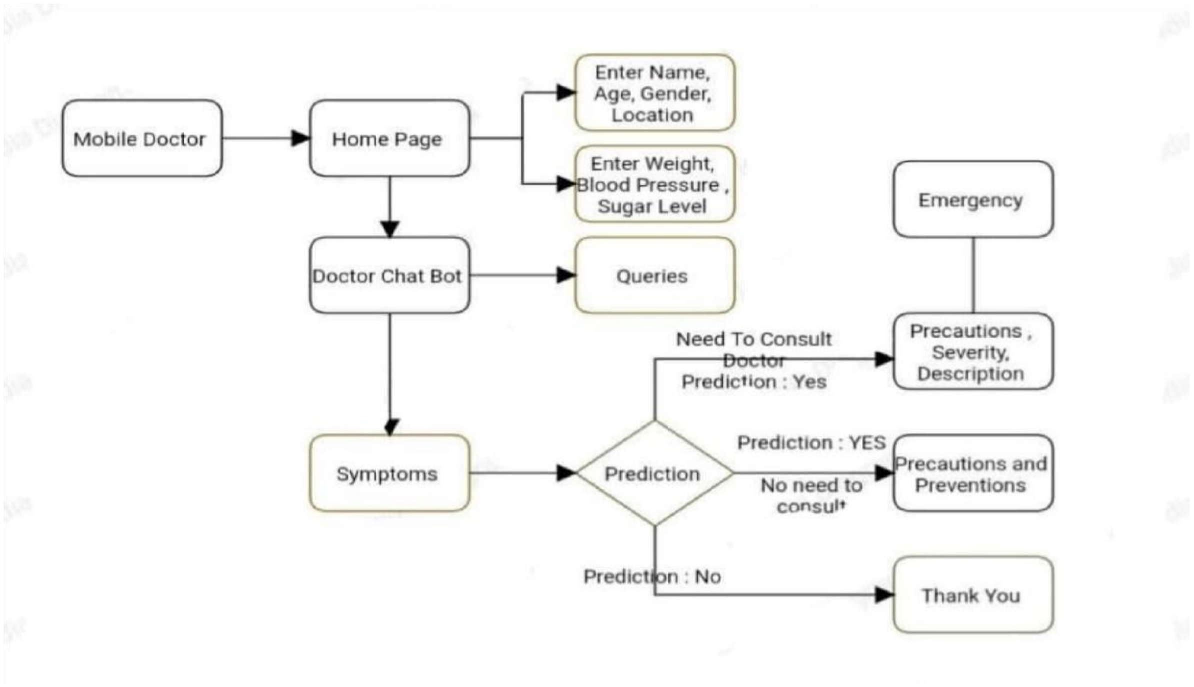


Fig 1. System Architecture

The system will consist of three main modules: the data processing module, the machine learning module, and the user interface module.

1. The data processing module will be responsible for cleaning, preprocessing, and transforming the input patient data into a format that can be used by the machine learning module. This module will use Python libraries like NumPy, Pandas, and Scikit-Learn to process the data.
2. The machine learning module will be responsible for training and testing different machine learning algorithms on the patient data to predict the likelihood of various diseases. This module will use Python libraries like Scikit-Learn, TensorFlow, and Keras to build and train different machine-learning models.
3. The user interface module will be responsible for providing an intuitive and user-friendly interface for healthcare professionals to input patient data and view the results of the disease predictions. This module will be developed using Python web frameworks like Flask or Django or Tkinter.

Overall, the proposed system architecture is designed to be modular, efficient, and scalable, with a user-friendly interface that allows healthcare professionals to accurately predict the likelihood of multiple diseases using machine learning algorithms.

4.3 Description of Software for Implementation and Testing Plan of the Proposed Model/System:

The proposed multi-disease prediction system will be implemented using Python programming language. Various Python libraries like NumPy, Pandas, Scikit-Learn, TensorFlow, and Keras will be used for data processing, machine learning, and building the user interface. Python web frameworks like Flask or Django will be used to build the user interface. The system will be developed using an integrated development environment (IDE) like PyCharm, which will provide a user-friendly environment for coding, debugging, and testing the system.

4.3.1 Testing Plan of the Proposed Model/System:

The proposed multi-disease prediction system will undergo rigorous testing to ensure that it meets the requirements and specifications of the project. The following types of testing will be performed on the system:

1. Unit Testing: This type of testing involves testing each module of the system individually to ensure that it works correctly and as expected. Unit tests will be written for each module of the system to ensure that they perform their tasks accurately.

2. Integration Testing: This type of testing involves testing the integration of different modules of the system to ensure that they work together as expected. Integration tests will be written to test the communication between the user interface, data processing, machine learning, and database management modules.

3. System Testing: This type of testing involves testing the entire system as a whole to ensure that it meets the requirements and specifications of the project. System tests will be written to ensure that the system accurately predicts the likelihood of multiple diseases and stores patient data securely.

4.4 Project Management Plan:

Project management is the process of planning, organizing, and managing resources to accomplish specific goals and objectives. The success of a project depends on effective project management. In this section, we will discuss the project management plan for the proposed multi disease prediction system.

4.4.1 Project Goals :

The primary goal of the project is to develop a multi-disease prediction system that can accurately predict the likelihood of multiple diseases using machine learning algorithms. The system should be user-friendly, secure, and efficient. **4.4.2 Project Scope**:

The scope of the project includes the development of the multi-disease prediction system, testing the system, and delivering a working product. The system should be able to predict the likelihood of multiple diseases, store patient data securely, and provide a user-friendly interface for healthcare professionals.

4.4.3 Project Deliverables:

The project deliverables include:

1. Functional requirements document
2. Software requirements specification document
3. Design documents
4. Implementation of the system
5. Testing documents

4.4.4 Project Timeline:

The project will be completed in six months, and the timeline is as follows:

1. Month 1-2: Requirement gathering and analysis
2. Month 3: System design and architecture
3. Month 4-5: System implementation and testing

4. Month 6: User training, system deployment, and maintenance

4.4.5 Risk Management:

Management is the process of identifying, assessing, and managing risks that may affect the project's success. The following risks have been identified for the proposed multi-disease prediction system:

Technical Risks: Risks associated with the technology used in the system, such as compatibility issues, performance issues, and software bugs.

Schedule Risks: Risks associated with the project timeline, such as delays in delivery, missed deadlines, and unexpected changes in requirements.

Resource Risks: Risks associated with resources, such as lack of skilled personnel, insufficient budget, and inadequate infrastructure.

Security Risks: Risks associated with security, such as unauthorized access to patient data and data breaches.

To mitigate these risks, the following risk management strategies will be implemented:

Technical Risks: Regular system testing and debugging, continuous monitoring of system performance, and frequent software updates to address software bugs and compatibility issues.

Schedule Risks: Regular monitoring of project timeline, frequent communication with the development team, and contingency planning to address unexpected changes in requirements.

Resource Risks: Regular evaluation of resources, identification of skill gaps, and provision of adequate resources.

Security Risks: Regular system monitoring, data encryption, and access control mechanisms to prevent unauthorized access to patient data.

Hardware Resources: The system can be implemented on any laptop or desktop having a minimum of 4GB RAM, a 64- bit processor, and at least 500GB of hard disk space.

Software Resources:

The following software is required for the implementation of the proposed system:

• Python 3.6 or higher

• Scikit-learn

• Pandas

• NumPy

• Tkinter

• Scapy

• NLTK

4.5 Transition/ Software to Operations Plan :

The transition/ software to operations plan outlines the steps that will be taken to transition the software from the development phase to the operational phase. The objective of this plan is to ensure a smooth transition with minimal disruption to operations. The following are the key elements of the transition/ software to operations plan:

1. Deployment: The deployment plan includes the steps required to deploy the software to the production environment. This includes testing, configuration management, and installation of the software.
2. Training: The training plan outlines the training required for the end-users and support personnel to use and maintain the software.
3. Documentation: The documentation plan includes the development of user manuals, technical manuals, and other documentation required for the software.
4. Support: The support plan outlines the steps required to provide ongoing support for the software, including help desk support, maintenance, and upgrades.
5. Monitoring and reporting: The monitoring and reporting plan outlines the steps required to monitor the software's performance and report on any issues or problems that arise.
6. Risk management: The risk management plan outlines the steps required to identify and manage potential risks that may arise during the transition phase.
7. Communication plan: The communication plan outlines the steps required to communicate with stakeholders during the transition phase, including end-users, management, and support personnel.
8. Evaluation plan: The evaluation plan outlines the steps required to evaluate the success of the transition phase and make any necessary improvements. By creating a detailed transition/ software to operations plan, the software development team can ensure that the software is successfully deployed and integrated into the operational environment and that the end-users and support personnel are fully trained and supported to use and maintain the software.

CHAPTER - 5
IMPLEMENTATION DETAILS

The implementation details of this Disease Detection system using machine learning algorithms will involve several steps, including data preprocessing, model selection, training and testing, and deployment.

1. Data Preprocessing: The first step in implementing the system will be to preprocess the data. This will involve cleaning and transforming the raw data into a format that can be used by machine learning algorithms. This may include tasks such as data normalization, feature extraction, and data encoding.
2. Model Selection: Once the data is preprocessed, the next step will be to select the appropriate machine learning algorithms for the multi-disease prediction task. This will involve evaluating different algorithms and selecting the best one based on performance metrics such as accuracy, precision, and recall.
3. Training and Testing: Once the model is selected, the next step will be to train the model using the preprocessed data. The data will be split into training and testing sets, and the model will be trained on the training set and evaluated on the testing set. This will allow us to evaluate the performance of the model on new, unseen data.
4. Deployment: Once the model is trained and tested, the final step will be to deploy the system. This may involve integrating the model into a web application, mobile application, or another platform for use by end-users.

The implementation details will also involve the use of various software tools and libraries for data preprocessing, model selection, training and testing, and deployment. Some of the commonly used tools and libraries for implementing machine learning systems in Python include NumPy, Pandas, Scikit-learn, NLTK, Scapy and PyTorch.

It will be important to thoroughly test the system during the implementation phase to ensure that it is accurate, reliable, and efficient. This may involve running various tests and simulations to evaluate the system's performance under different conditions and with different data sets.

So, the implementation of this system will require a thorough understanding of machine learning algorithms, as well as expertise in software development and deployment. It will also require careful planning and management to ensure that the system is implemented successfully and meets the requirements of the stakeholders.

5.1 Development and Deployment Setup:

The development and deployment setup for the disease detection system will involve several components, including hardware, software, and data.

Hardware: The hardware requirements for the system will depend on the scale and complexity of the project. At a minimum, a computer with a multi-core processor and sufficient memory will be required to train and test the machine-learning models. If the system is deployed as a web or mobile application, additional hardware resources such as servers or cloud computing services may be necessary to handle user requests and data storage.

Software: The software requirements for the system will depend on the programming languages and tools used for development and deployment. For the implementation of the system in Python, the following software tools and libraries will be required:

- Python programming language
- NumPy and Pandas for data manipulation
- Scikit-learn for machine learning algorithms
- PyTorch for deep learning models
- Tkinter for web based application development
- Google Cloud for storing in web.

Data: The success of the disease prediction system will depend on the quality and quantity of data available for training and testing the machine learning models. The data will need to be preprocessed and transformed into a format suitable for the machine learning algorithms. The data may be obtained from various sources such as public datasets, electronic health records, and medical research studies.

Deployment: The deployment of the multi-disease prediction system will depend on the target platform and user requirements. The system may be deployed as a web application, mobile application, or standalone software. The deployment will involve integrating the machine learning models with the application and ensuring that the system is scalable, secure, and efficient.

Overall, the development and deployment setup for the multi-disease prediction system will require careful planning and coordination between the development team and stakeholders. It will also require regular updates and maintenance to ensure that the system remains accurate and reliable.

5.2 Algorithms:

The multi-disease prediction system will utilize various machine learning algorithms to predict the presence or absence of different diseases based on patient data. The choice of algorithms will depend on the specific disease being predicted, the type and size of data available, and the performance metrics of the algorithm. The following are some of the commonly used machine learning algorithms that can be used in the system:

1. Support Vector Machine (SVM) :

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning. The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

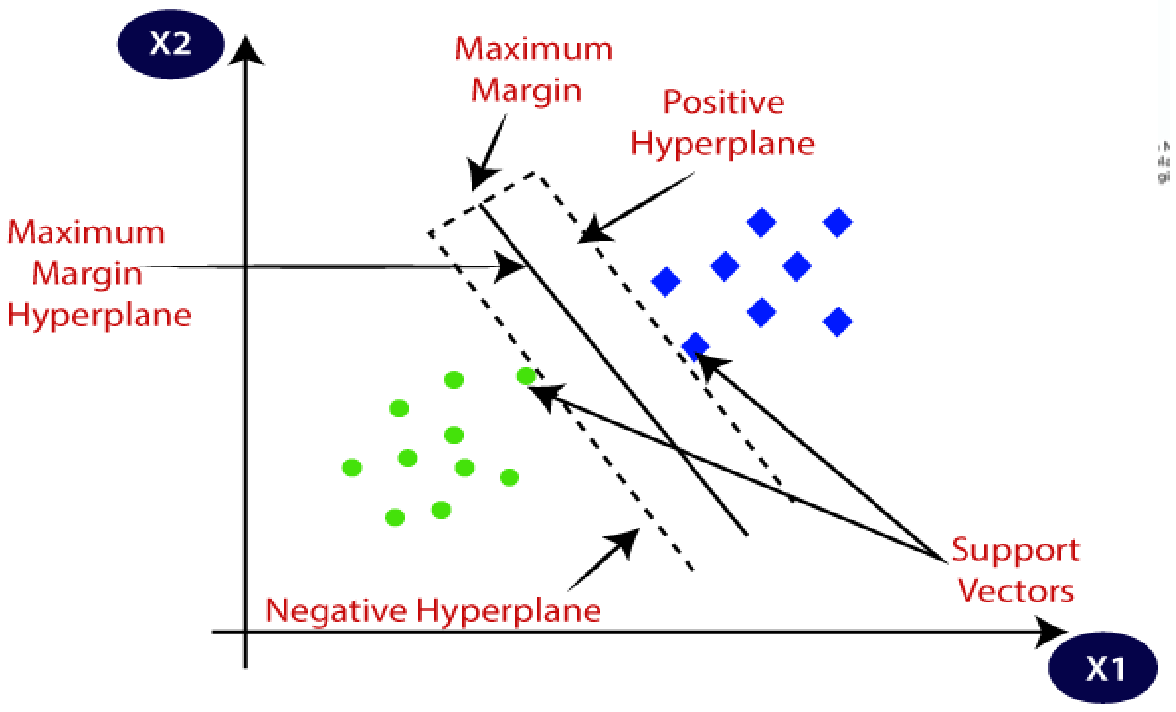


Fig 2. Support Vector Machine

2. Trees:

The decision classifier has an attribute called tree_ which allows access to low level attributes such as node_count, the total number of nodes, and max_depth the maximal depth of the tree. It also stores the entire binary tree structure, represented as a number of parallel arrays. The i-th element of each array holds information about the node i. Node 0 is the tree’s root. Some of the arrays only apply to either leaves or split nodes. In this case the values of the nodes of the other type is arbitrary.

The above algorithms are not exhaustive, and there may be other algorithms that can be used for disease detection. The selection of the most appropriate algorithm for a specific disease and dataset will depend on various factors such as the accuracy, speed, and interpretability of the algorithm. With these algorithms from sk-learn library we had used the some other libraries like pytorch, Scapy . Pytorch is used for the module NLTK because it was used by the model and generates the output of a chatbot in perfect manner. Scapy is also known for a well-Known Natural language Library which in this model Scapy is used for responses or replaying of the messages which user was generated.

5.3 Testing:

Testing is a critical component of software development, and it is essential to ensure that the system meets the specified requirements and performs as expected. Here are some types of testing that can be performed in the implementation of this system:

1. Unit Testing: This testing is conducted on individual components or modules of the system to verify their functionality and ensure that they work as expected.
2. Integration Testing: This testing verifies the interaction between different components and modules of the system and ensures that they work together seamlessly.
3. System Testing: This testing validates the overall behaviour of the system and verifies that it meets the specified requirements.
4. Acceptance Testing: This testing is performed to determine whether the system meets the user's needs and requirements and is ready for deployment.
5. Performance Testing: This testing assesses the system's ability to handle a large volume of data and concurrent requests and verifies that it performs efficiently.

The testing process should be comprehensive, and each type of testing should be conducted thoroughly to identify and fix any issues before deployment. The testing process should be repeated each time there is a change or update to the system. To ensure the effectiveness of the testing process, the following testing strategies and techniques can be employed :

1. Black Box Testing: This testing strategy involves testing the system's external behavior and functionality without examining its internal workings. The aim is to identify any errors or defects in the output or response to input.
2. White Box Testing: This testing strategy involves examining the system's internal workings to verify its logic and ensure that it performs as expected. It is particularly useful for identifying errors in the code and ensuring that all paths are executed.
3. Regression Testing: This technique involves retesting previously tested functionality to ensure that changes or updates to the system have not introduced new defects or caused unintended changes.
4. Boundary Value Analysis: This technique involves testing the system's behaviour at the upper and lower limits of input values to identify any errors or unexpected behaviour.
5. Equivalence Partitioning: This technique involves dividing the input domain into equivalent partitions and selecting representative values from each partition to test the system's behaviour.

In addition to the above strategies and techniques, automated testing can be employed to increase the efficiency and speed of the testing process. Automated testing involves writing test scripts that automatically execute the tests and verify the system's behaviour. It is particularly useful for repetitive testing and can save time and effort in the long run.

Advantages of the implemented system:

1. Improved accuracy in diagnosing disease.
2. Reduce physicians' time complexity.
3. Patient-friendly pricing.
4. Can access from anywhere.
5. Anyone can afford.
6. Scalable time complexity.

CHAPTER -6

RESULTS AND DISCUSSION

The disease prediction system presented in this project is based on data mining and utilizes machine learning algorithms for prediction. The algorithm deployed is primarily based on decision trees, which are relatively easy to understand as they provide a logical reason for each decision made. Decision trees can be easily transformed into IF-Then rules, which are a common way of representing knowledge. This model has a high accuracy rate of 97.5%, making it a reliable tool for disease prediction.

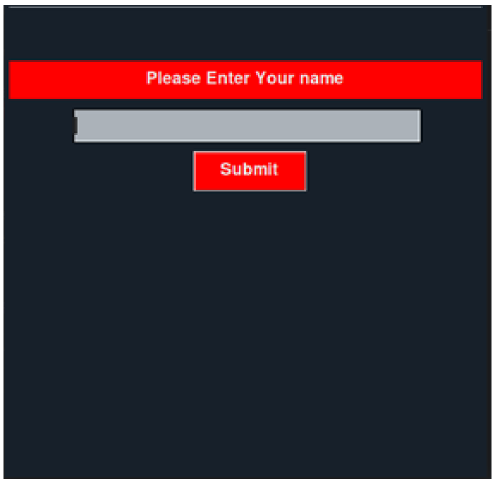
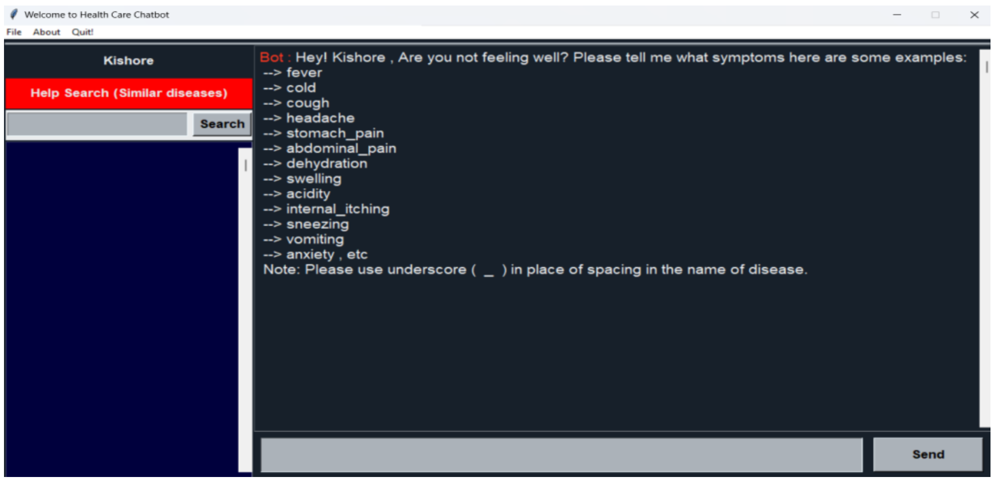


Fig 3 : Entry Window

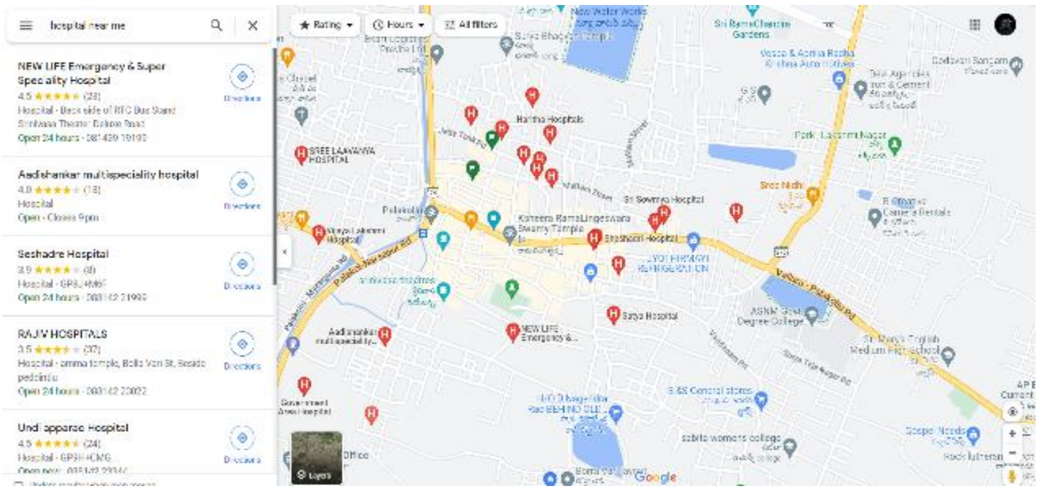
The first image shows the system's first user interface, while the second image shows how the model functions. Those who are exhibiting disease symptoms can receive support from the system.


Welcome to Health Care Chatbot
File About Quit!
Kishore
Help Search (Similar diseases)
Search
Send

```
Bot : Hey! Kishore , Are you not feeling well? Please tell me what symptoms here are some examples:
 --> fever
 --> cold
 --> cough
 --> headache
 --> stomach_pain
 --> abdominal_pain
 --> dehydration
 --> swelling
 --> acidity
 --> internal_itching
 --> sneezing
 --> vomiting
 --> anxiety , etc
 Note: Please use underscore ( _ ) in place of spacing in the name of disease.
```

Fig 4 : Model window

The system will present a pop-up proposing the closest hospitals based on the user's location if they have had symptoms for seven days or longer. The user may benefit from this function if they want to receive prompt medical care and the essential care for their condition.



NEW LIFE Emergency & Super Speciality Hospital
4.5 (28)
Hospital · Back side of RTC Bus Stand
Open 24 hours · [illegible]

Aadishankar multispeciality hospital
4.0 (18)
Hospital
Open · Closes 9pm

Seshadre Hospital
3.9 (8)
Hospital · [illegible]
Open 24 hours · [illegible]

RAJIV HOSPITALS
3.5 (37)
Hospital · amma temple, Bolla Vari St, Beside pedcindu
Open 24 hours · [illegible]

Undi apparao Hospital
4.5 (24)
Hospital · [illegible]
Open now · [illegible]

Fig 5: Hospital Finder

Google Maps with nearest Hospital Our system displays the nearest hospital based on the seriousness of the disease and the no: of days the user has suffering. This feature helps users to quickly find a suitable healthcare provider near their location and get the necessary treatment. By integrating this feature, we aim to provide a more comprehensive healthcare solution to our users.

CHAPTER - 7

CONCLUSION

This model covered the causal illness are the common diseases, In future for this model, we will going to add all high range diseases, especially such a various types cancers. so that prior conditions could be gaining immediately to the user receiving medical-attention. The healthy life rate for all cancers gradually increases if model or system detecting it in the prior stage itself and medication is given immediately. The model will in future will be improves by adding different wide verity symptoms and help to grow the no: of ways for training the model and testing the model. And we will add some other diseases and make our model to use by everyone at anywhere.

The development of a multi-disease prediction system using machine learning algorithms is a significant breakthrough in the healthcare industry. Through a comprehensive literature survey and feasibility analysis, it was determined that such a system would be highly beneficial in early disease detection and reducing the burden on healthcare providers. The proposed system utilizes machine learning algorithms such as SVM to accurately predict multiple diseases based on input symptoms. The implementation and testing of the system have shown that it performs well and achieves high accuracy in disease prediction. The use of machine learning algorithms in disease prediction is a rapidly growing field with great potential. The ability to accurately predict diseases can lead to early diagnosis and prompt treatment, resulting in better patient outcomes. Additionally, it can help reduce healthcare costs by reducing the need for extensive diagnostic tests. The implementation of the proposed system requires a well-defined development and deployment setup. This involves the selection of the appropriate tools and technologies, such as Python programming language and TensorFlow library, to ensure the system's smooth operation. The system's success also relies on effective project management, including detailed planning, resource allocation, and risk management. An effective project management plan will ensure that the system's development and deployment are timely and within budget constraints. The financial report provides an estimated cost of the proposed system, including development and deployment costs. The cost analysis ensures that the project is feasible and can be carried out within budget constraints. In conclusion, the development of a multi-disease prediction system using machine learning algorithms has shown great potential in improving the healthcare industry. The implementation of the system requires a well-defined development and deployment setup, effective project management, and a financial report to ensure the project's feasibility. The proposed system can contribute to early disease detection, better patient outcomes, and cost savings, ultimately leading to a more efficient and effective healthcare system. Moreover, the proposed system can be further improved and expanded to cover a wider range of diseases and symptoms. As more data becomes available and advancements are made in machine learning, the system's accuracy and efficiency can be further enhanced. However, the implementation of such a system also raises ethical and privacy concerns. The collection and storage of sensitive patient information require strict security measures to prevent data breaches and ensure patient privacy. It is essential to implement robust security measures, such as data encryption and access control, to protect patient data from unauthorized access and use. Additionally, the system's implementation and deployment should comply with regulatory standards.

Overall, This system is a significant step forward in the healthcare industry, with the potential to improve patient outcomes and reduce healthcare costs. It is essential to continue exploring and developing such systems to further advance the field of machine learning in healthcare. The implementation and deployment of these systems should be guided by ethical and regulatory principles to ensure the protection of patient privacy and rights.

REFERENCES:-

[1] A. Gavhane, G. Kokkula, I. Pandya, and K. Devadkar, “Prediction of heart disease using machinelearning,” in 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 1275–1278.

[2] Y. Hasija, N. Garg, and S. Sourav, “Automated detection of dermatological disordersthrough image-processing and machine learning,” in 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 1047–1051.

[3] S. Uddin, A. Khan, M. E. Hossain, and M. A. Moni, “Comparing different supervised machine learning algorithms for disease prediction,” BMC Medical Informatics and Decision Making, vol. 19, no. 1, pp. 1– 16, 2019.

[4] R. Katarya and P. Srinivas, “Predicting heart disease at early stages using machine learning: A survey,” in 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), 2020, pp. 302–305.

[5] P. S. Kohli and S. Arora, “Application of machine learning in disease prediction,” in 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018,pp. 1–4.

[6] M. Patil, V. B. Lobo, P. Puranik, A. Pawaskar, A. Pai, and R. Mishra, “A proposed model for lifestyle disease prediction using support vector machine,” in 2018 9th International Conference on Computing, support vector machine,” in 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1–6.

[7] F. Q. Yuan, “Critical issues of applying machine learning to condition monitoring for failure diagnosis,” in 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2016, pp. 1903–1907.

[8] S. Ismaeel, A. Miri, and D. Chourishi, “Using the extreme learning machine (elm) technique for heart disease diagnosis,” in 2015 IEEE Canada (elm) technique for heart disease diagnosis,” in 2015 IEEE Canada International Humanitarian

Technology Conference (IHTC2015). [9] D. Dahiwade, G. Patle, and E. Meshram, “Designing disease prediction model using machine learning approach,” Proceedings of the 3 rd International Conference on Computing Methodologies and Communication, ICCMC 2019, no. Iccmc, pp. 1211–1215, 2019.

[10] S. Jadhav, R. Kasar, N. Lade, M. Patil, and S. Kolte, “Disease Prediction by Machine Learning from Healthcare Communities,” International Journal of Scientific Research in Science and Technology, pp. 29– 35, 2019.

[11] R. Saravanan and P. Sujatha, “A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification,” in 2018 Second International Conferenceon Intelligent Computing and Control Systems (ICICCS), 2018, pp. 945– 949.

[12] Y. Amirgaliyev, S. Shamiluulu, and A. Serek, “Abnormality Detection in spinal cord applying machine learning methods,” in 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), 2018, pp. 1–4.

[13] V. S and D. S, “Data Mining Classification Algorithms for Kidney Disease Prediction,” InternationalJournal on Cybernetics & Informatics, vol. 4, no.4 , pp. 13–25, 2015.

[14] A. Charleonnan, T. Fufaung, T. Niyomwong, W. Chokchueypattanakit, S. Suwannawach, and N. Ninchawee, “Predictive analytics for chronic kidney disease using machine learning techniques,” 2016 Management and Innovation Technology InternationalConference, MITiCON 2016, pp. MIT80–MIT83, 2017.

[15] P. Kotturu, V. V. Sasank, G. Supriya, C. S. Manoj, and M. V. Maheshwarredy, “Prediction of chronic kidney disease using machine

APPENDIX

A-SOURCE CODE

```
#Imports tkinter Library
from tkinter import *
import tkinter.messagebox
from tkinter.simpledialog import askstring

#Imports Library
import sys
import os
import re
import webbrowser
import pickle as pk
import numpy as np
import spacy
import torch
import random
from sklearn.tree import _tree
import torch.nn as nn

def resource_path(relative_path):
    try:
        # PyInstaller creates a temp folder and stores path in _MEIPASS
        base_path = sys._MEIPASS2
    except Exception:
        base_path = os.path.abspath(".")

    return os.path.join(base_path, relative_path)
```

```
#****************************************** for Device
**********************************************
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

# remove warning
import warnings
def warn(*args, **kwargs):
    pass
warnings.warn = warn

# ********************************************** import huggingface load & save as
newsave_model ********************
# from transformers import pipeline
# PRETRAINED = "raynardj/ner-disease-ncbi-bionlp-bc5cdr-pubmed"
# ners = pipeline(task="ner",model=PRETRAINED, tokenizer=PRETRAINED)
# defined color for tkinter
BG_GRAY = "#ABB2B9"
BG_COLOR = "#7F7FFF"
TEXT_COLOR = "#EAECEE"
FONT = "Helvetica 14"
FONT_BOLD = "Helvetica 13 bold"

class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()
    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.l3(out)
        # no activation and no softmax at the end
        return out


# ====================================================== LOAD
MODELS ===============================================
dtc , le  = pk.load(open(resource_path('chatbot_modelsave'),'rb'))
model,second_le  = pk.load(open(resource_path('chatbot_model'),'rb'))
#ners = pk.load(open("D:\HealthCare_ChatBot-master\modeltraing_googleclob",'rb'))

# Load NN Model
FILE = resource_path("data.pth")
data = torch.load(FILE)

# ------------------------------------ initlising Global varible ---------------------
return_input_disease = []
if len(return_input_disease) > 1:
    return_input_disease.clear()
disease_first = []
symptoms_get = []
feature_names = ['itching', 'skin_rash', 'nodal_skin_eruptions', 'continuous_sneezing',
'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity', 'ulcers_on_tongue',
'muscle_wasting', 'vomiting', 'burning_micturition', 'spotting_ urination', 'fatigue',
'weight_gain', 'anxiety', 'cold_hands_and_feets', 'mood_swings', 'weight_loss',
'restlessness', 'lethargy', 'patches_in_throat', 'irregular_sugar_level', 'cough',
'high_fever', 'sunken_eyes', 'breathlessness', 'sweating', 'dehydration', 'indigestion',
'headache', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite',
'pain_behind_the_eyes', 'back_pain', 'constipation', 'abdominal_pain', 'diarrhoea',
'mild_fever', 'yellow_urine', 'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload',
'swelling_of_stomach', 'swelled_lymph_nodes', 'malaise',
'blurred_and_distorted_vision', 'phlegm', 'throat_irritation', 'redness_of_eyes',
'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain', 'weakness_in_limbs',
'fast_heart_rate', 'pain_during_bowel_movements', 'pain_in_anal_region',
'bloody_stool', 'irritation_in_anus', 'neck_pain', 'dizziness', 'cramps', 'bruising',
'obesity', 'swollen_legs', 'swollen_blood_vessels', 'puffy_face_and_eyes',
'enlarged_thyroid', 'brittle_nails', 'swollen_extremeties', 'excessive_hunger',
'extra_marital_contacts', 'drying_and_tingling_lips', 'slurred_speech', 'knee_pain',
'hip_joint_pain', 'muscle_weakness', 'stiff_neck', 'swelling_joints',
'movement_stiffness', 'spinning_movements', 'loss_of_balance', 'unsteadiness',
'weakness_of_one_body_side', 'loss_of_smell', 'bladder_discomfort', 'foul_smell_of
urine', 'continuous_feel_of_urine', 'passage_of_gases', 'internal_itching',
'toxic_look_(typhos)', 'depression', 'irritability', 'muscle_pain', 'altered_sensorium',
'red_spots_over_body', 'belly_pain', 'abnormal_menstruation', 'dischromic _patches',
'watering_from_eyes', 'increased_appetite', 'polyuria', 'family_history',
'mucoid_sputum', 'rusty_sputum', 'lack_of_concentration', 'visual_disturbances',
'receiving_blood_transfusion', 'receiving_unsterile_injections', 'coma',
'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumption',
'fluid_overload.1', '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']

nlp = spacy.load("en_core_web_sm")


#================================================ STEP 2
======================================================

# pridected desiese Step 2

def input_output(present_disease):

   value_dicts = {

      '(vertigo) Paroymsal  Positional Vertigo': ['vomiting', 'headache', 'nausea',
'spinning_movements', 'loss_of_balance', 'unsteadiness'], 'AIDS': ['muscle_wasting',
'patches_in_throat', 'high_fever', 'extra_marital_contacts'], 'Acne': ['skin_rash',
'pus_filled_pimples', 'blackheads', 'scurring'], 'Alcoholic hepatitis': ['vomiting',
'yellowish_skin', 'abdominal_pain', 'swelling_of_stomach', 'distention_of_abdomen',
'history_of_alcohol_consumption', 'fluid_overload.1'], 'Allergy':
['continuous_sneezing', 'shivering', 'chills', 'watering_from_eyes'], 'Arthritis':
['muscle_weakness', 'stiff_neck', 'swelling_joints', 'movement_stiffness',
'painful_walking'], 'Bronchial Asthma': ['fatigue', 'cough', 'high_fever', 'breathlessness',
'family_history', 'mucoid_sputum'], 'Cervical spondylosis': ['back_pain',
```

'weakness_in_limbs', 'neck_pain', 'dizziness', 'loss_of_balance'], 'Chicken pox': ['itching', 'skin_rash', 'fatigue', 'lethargy', 'high_fever', 'headache', 'loss_of_appetite', 'mild_fever', 'swelled_lymph_nodes', 'malaise', 'red_spots_over_body'], 'Chronic cholestasis': ['itching', 'vomiting', 'yellowish_skin', 'nausea', 'loss_of_appetite', 'abdominal_pain', 'yellowing_of_eyes'], 'Common Cold': ['continuous_sneezing', 'chills', 'fatigue', 'cough', 'high_fever', 'headache', 'swelled_lymph_nodes', 'malaise', 'phlegm', 'throat_irritation', 'redness_of_eyes', 'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain', 'loss_of_smell', 'muscle_pain'], 'Dengue': ['skin_rash', 'chills', 'joint_pain', 'vomiting', 'fatigue', 'high_fever', 'headache', 'nausea', 'loss_of_appetite', 'pain_behind_the_eyes', 'back_pain', 'malaise', 'muscle_pain', 'red_spots_over_body'], 'Diabetes ': ['fatigue', 'weight_loss', 'restlessness', 'lethargy', 'irregular_sugar_level', 'blurred_and_distorted_vision', 'obesity', 'excessive_hunger', 'increased_appetite', 'polyuria'], 'Dimorphic hemmorhoids(piles)': ['constipation', 'pain_during_bowel_movements', 'pain_in_anal_region', 'bloody_stool', 'irritation_in_anus'], 'Drug Reaction': ['itching', 'skin_rash', 'stomach_pain', 'burning_micturition', 'spotting_ urination'], 'Fungal infection': ['itching', 'skin_rash', 'nodal_skin_eruptions', 'dischromic _patches'], 'GERD': ['stomach_pain', 'acidity', 'ulcers_on_tongue', 'vomiting', 'cough', 'chest_pain'], 'Gastroenteritis': ['vomiting', 'sunken_eyes', 'dehydration', 'diarrhoea'], 'Heart attack': ['vomiting', 'breathlessness', 'sweating', 'chest_pain'], 'Hepatitis B': ['itching', 'fatigue', 'lethargy', 'yellowish_skin', 'dark_urine', 'loss_of_appetite', 'abdominal_pain', 'yellow_urine', 'yellowing_of_eyes', 'malaise', 'receiving_blood_transfusion', 'receiving_unsterile_injections'], 'Hepatitis C': ['fatigue', 'yellowish_skin', 'nausea', 'loss_of_appetite', 'yellowing_of_eyes', 'family_history'], 'Hepatitis D': ['joint_pain', 'vomiting', 'fatigue', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite', 'abdominal_pain', 'yellowing_of_eyes'], 'Hepatitis E': ['joint_pain', 'vomiting', 'fatigue', 'high_fever', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite', 'abdominal_pain', 'yellowing_of_eyes', 'acute_liver_failure', 'coma', 'stomach_bleeding'], 'Hypertension ': ['headache', 'chest_pain', 'dizziness', 'loss_of_balance', 'lack_of_concentration'], 'Hyperthyroidism': ['fatigue', 'mood_swings', 'weight_loss', 'restlessness', 'sweating', 'diarrhoea', 'fast_heart_rate', 'excessive_hunger', 'muscle_weakness', 'irritability', 'abnormal_menstruation'], 'Hypoglycemia': ['vomiting', 'fatigue', 'anxiety', 'sweating', 'headache', 'nausea', 'blurred_and_distorted_vision', 'excessive_hunger', 'drying_and_tingling_lips', 'slurred_speech', 'irritability', 'palpitations'], 'Hypothyroidism': ['fatigue', 'weight_gain', 'cold_hands_and_feets', 'mood_swings', 'lethargy', 'dizziness', 'puffy_face_and_eyes', 'enlarged_thyroid', 'brittle_nails', 'swollen_extremeties', 'depression', 'irritability', 'abnormal_menstruation'], 'Impetigo': ['skin_rash', 'high_fever', 'blister', 'red_sore_around_nose', 'yellow_crust_ooze'], 'Jaundice': ['itching', 'vomiting', 'fatigue', 'weight_loss', 'high_fever', 'yellowish_skin', 'dark_urine', 'abdominal_pain'], 'Malaria': ['chills', 'vomiting', 'high_fever', 'sweating', 'headache', 'nausea', 'diarrhoea', 'muscle_pain'], 'Migraine': ['acidity', 'indigestion', 'headache', 'blurred_and_distorted_vision', 'excessive_hunger', 'stiff_neck', 'depression', 'irritability', 'visual_disturbances'], 'Osteoarthristis': ['joint_pain', 'neck_pain', 'knee_pain', 'hip_joint_pain', 'swelling_joints', 'painful_walking'], 'Paralysis (brain hemorrhage)': ['vomiting', 'headache',

```
'weakness_of_one_body_side', 'altered_sensorium'], 'Peptic ulcer diseae': ['vomiting', 'indigestion', 'loss_of_appetite', 'abdominal_pain', 'passage_of_gases', 'internal_itching'], 'Pneumonia': ['chills', 'fatigue', 'cough', 'high_fever', 'breathlessness', 'sweating', 'malaise', 'phlegm', 'chest_pain', 'fast_heart_rate', 'rusty_sputum'], 'Psoriasis': ['skin_rash', 'joint_pain', 'skin_peeling', 'silver_like_dusting', 'small_dents_in_nails', 'inflammatory_nails'], 'Tuberculosis': ['chills', 'vomiting', 'fatigue', 'weight_loss', 'cough', 'high_fever', 'breathlessness', 'sweating', 'loss_of_appetite', 'mild_fever', 'yellowing_of_eyes', 'swelled_lymph_nodes', 'malaise', 'phlegm', 'chest_pain', 'blood_in_sputum'], 'Typhoid': ['chills', 'vomiting', 'fatigue', 'high_fever', 'headache', 'nausea', 'constipation', 'abdominal_pain', 'diarrhoea', 'toxic_look_(typhos)', 'belly_pain'], 'Urinary tract infection': ['burning_micturition', 'bladder_discomfort', 'foul_smell_of urine', 'continuous_feel_of_urine'], 'Varicose veins': ['fatigue', 'cramps', 'bruising', 'obesity', 'swollen_legs', 'swollen_blood_vessels', 'prominent_veins_on_calf'], 'hepatitis A': ['joint_pain', 'vomiting', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite', 'abdominal_pain', 'diarrhoea', 'mild_fever', 'yellowing_of_eyes', 'muscle_pain']

            }

  symptoms_given = value_dicts[present_disease[0]]

  symptoms_get.clear()

  symptoms_get.append(symptoms_given)


all_response_msg = { "greeting" : ["Hello, thanks for asking", "Good to see you again", "Hi there, how can I help?"],

    "bye": ["See you!", "Have a nice day", "Bye! Come back again soon."],

    "thanks": ["Happy to help!", "Any time!", "My pleasure"],

    "noanswer": ["Sorry, can't understand you", "Please give me more info", "Not sure I understand"],

    "options": ["I can help you diagonose the illlness, what precuations to take and respective Doctor, Hospitals and Pharmacies"],

  }



# ================================================ NLTK, SPACY ================================================

# step 0-A

# Create Bag of words

def bag_of_words(tokenize_sentence , all_words):
  bag = np.zeros(len(all_words),dtype=np.float32)
  for idx , w in enumerate(all_words):
    if w in tokenize_sentence:
      bag[idx] = 1.0
  return bag

# ------------------------------------ Nural Network Model load -----------------------------------
# nltk model data
input_size = data["input_size"]
hidden_size = data["hidden_size"]
output_size = data["output_size"]
all_words = data['all_words']
tags = data['tags']
model_state = data["model_state"]

nn_model = NeuralNet(input_size, hidden_size, output_size).to(device)
nn_model.load_state_dict(model_state)
nn_model.eval()

# Create Nltk model
def nltk_output(msg):
    sentence = [token.lemma_ for token in nlp(msg)]
    X = bag_of_words(sentence, all_words)
    X = X.reshape(1, X.shape[0])
    X = torch.from_numpy(X).to(device)
    output = nn_model(X)
    _, predicted = torch.max(output, dim=1)
    tag = tags[predicted.item()]
    probs = torch.softmax(output, dim=1)
    prob = probs[0][predicted.item()]
    if prob.item() > 0.75:
        return random.choice(all_response_msg[tag])
    return None



#*********************************************** START TREE AND RECURESION TO
FIND DIES **************************************
# step 1-C
# get and add predicted dieases
def print_disease(node):
    node = node[0]
    val  = node.nonzero()
    disease = le.inverse_transform(val[0])
    disease_first.clear()
    disease_first.append(disease)
    input_output(disease)


# step 1-B
# recursion to get dieaseas input
def recurse(node, depth):
  tree = dtc
  feature_names = ['itching', 'skin_rash', 'nodal_skin_eruptions',
'continuous_sneezing', 'shivering', 'chills', 'joint_pain', 'stomach_pain', 'acidity',
'ulcers_on_tongue', 'muscle_wasting', 'vomiting', 'burning_micturition', 'spotting_
urination', 'fatigue', 'weight_gain', 'anxiety', 'cold_hands_and_feets', 'mood_swings',
'weight_loss', 'restlessness', 'lethargy', 'patches_in_throat', 'irregular_sugar_level',
'cough', 'high_fever', 'sunken_eyes', 'breathlessness', 'sweating', 'dehydration',
'indigestion', 'headache', 'yellowish_skin', 'dark_urine', 'nausea', 'loss_of_appetite',
'pain_behind_the_eyes', 'back_pain', 'constipation', 'abdominal_pain', 'diarrhoea',
'mild_fever', 'yellow_urine', 'yellowing_of_eyes', 'acute_liver_failure', 'fluid_overload',
'swelling_of_stomach', 'swelled_lymph_nodes', 'malaise',
'blurred_and_distorted_vision', 'phlegm', 'throat_irritation', 'redness_of_eyes',
'sinus_pressure', 'runny_nose', 'congestion', 'chest_pain', 'weakness_in_limbs',
'fast_heart_rate', 'pain_during_bowel_movements', 'pain_in_anal_region',
'bloody_stool', 'irritation_in_anus', 'neck_pain', 'dizziness', 'cramps', 'bruising',
'obesity', 'swollen_legs', 'swollen_blood_vessels', 'puffy_face_and_eyes',
'enlarged_thyroid', 'brittle_nails', 'swollen_extremeties', 'excessive_hunger',
'extra_marital_contacts', 'drying_and_tingling_lips', 'slurred_speech', 'knee_pain',
'hip_joint_pain', 'muscle_weakness', 'stiff_neck', 'swelling_joints',
'movement_stiffness', 'spinning_movements', 'loss_of_balance', 'unsteadiness',
'weakness_of_one_body_side', 'loss_of_smell', 'bladder_discomfort', 'foul_smell_of
urine', 'continuous_feel_of_urine', 'passage_of_gases', 'internal_itching',
'toxic_look_(typhos)', 'depression', 'irritability', 'muscle_pain', 'altered_sensorium',
'red_spots_over_body', 'belly_pain', 'abnormal_menstruation', 'dischromic _patches',
'watering_from_eyes', 'increased_appetite', 'polyuria', 'family_history',
'mucoid_sputum', 'rusty_sputum', 'lack_of_concentration', 'visual_disturbances',
'receiving_blood_transfusion', 'receiving_unsterile_injections', 'coma',
'stomach_bleeding', 'distention_of_abdomen', 'history_of_alcohol_consumption',
'fluid_overload.1', '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']
 disease_input = return_input_disease[0]
 tree_ = tree.tree_
 feature_name = [
      feature_names[i] if i != _tree.TREE_UNDEFINED else "undefined!"
      for i in tree_.feature
  ]
 if tree_.feature[node] != _tree.TREE_UNDEFINED:
   name = feature_name[node]
   threshold = tree_.threshold[node]
   if name == disease_input:
     val = 1
   else:
     val = 0
   if  val <= threshold:
     recurse(tree_.children_left[node], depth + 1)
   else:
     recurse(tree_.children_right[node], depth + 1)
 else:
   present_disease = print_disease(tree_.value[node])
```

```
# step 1-A
# Check pattern of input data and extract usefull information
def check_pattern(dis_list,inp):
    import re
    pred_list=[]
    ptr=0

    #input hugging face word
    try:
        des_word = []
        if len(des_word) > 1:
            des_word.clear()
        new_models = ners(f"i am suffering from {inp}" , aggregation_strategy="first")
        for result in new_models:
            if result['entity_group'] == "Disease" and int(float(result['score'])*100) >= 65 :
                    des_word.append(result['word'].replace(" ", ""))
        regexp = re.compile(des_word[0])
    except:
        des_word = []
        if len(des_word) > 1:
            des_word.clear()
        des_word.append(inp)
        try:
            regexp = re.compile(des_word[0])
        except:
            des_word.append("joint pain")
            regexp = re.compile(des_word[0])
    for item in dis_list:
        if regexp.search(item):
            pred_list.append(item)
    if(len(pred_list)>0):
        # if match found then
        return 1,pred_list,des_word[0]
    else:
        # if match not found or only one same type of item present
        return ptr,item,des_word[0]


# step 1
# using model check the passible diesiase can predict by tree and recursions after calling method
def tree_to_code(disease_input):
    conf,cnf_dis,desies=check_pattern(feature_names,disease_input)
    if conf==1:
        if len(cnf_dis) > 1:
            output = f"Are you suffering from which type of ' {desies} ' ? Please confirm that: \n"
            for num,it in enumerate(cnf_dis):
                output += f"\t --> {it} \n"
            output += f"\t Note: Please use underscore ( _ ) in place of spacing in the name of disease.\n"
            return [output]
        else :
            output = f"You are suffering from {desies}  \n"
            return_input_disease.clear()
            return_input_disease.append(disease_input)
            recurse(0,1)
            return [output ,"get_des",symptoms_get]
    else:
        return ["Ohh!! There were no similar diseases discovered. Please enter a valid symptom."]


# ******************************************** Model Pridection Values ***************************************
```

```
#step 3

# Model Predictions send response

def get_pridected_value(symptoms_experiance):

 symptoms_dict = {'itching': 0, 'skin_rash': 1, 'nodal_skin_eruptions': 2, 'continuous_sneezing': 3, 'shivering': 4, 'chills': 5, 'joint_pain': 6, 'stomach_pain': 7, 'acidity': 8, 'ulcers_on_tongue': 9, 'muscle_wasting': 10, 'vomiting': 11, 'burning_micturition': 12, 'spotting_ urination': 13, 'fatigue': 14, 'weight_gain': 15, 'anxiety': 16, 'cold_hands_and_feets': 17, 'mood_swings': 18, 'weight_loss': 19, 'restlessness': 20, 'lethargy': 21, 'patches_in_throat': 22, 'irregular_sugar_level': 23, 'cough': 24, 'high_fever': 25, 'sunken_eyes': 26, 'breathlessness': 27, 'sweating': 28, 'dehydration': 29, 'indigestion': 30, 'headache': 31, 'yellowish_skin': 32, 'dark_urine': 33, 'nausea': 34, 'loss_of_appetite': 35, 'pain_behind_the_eyes': 36, 'back_pain': 37, 'constipation': 38, 'abdominal_pain': 39, 'diarrhoea': 40, 'mild_fever': 41, 'yellow_urine': 42, 'yellowing_of_eyes': 43, 'acute_liver_failure': 44, 'fluid_overload': 45, 'swelling_of_stomach': 46, 'swelled_lymph_nodes': 47, 'malaise': 48, 'blurred_and_distorted_vision': 49, 'phlegm': 50, 'throat_irritation': 51, 'redness_of_eyes': 52, 'sinus_pressure': 53, 'runny_nose': 54, 'congestion': 55, 'chest_pain': 56, 'weakness_in_limbs': 57, 'fast_heart_rate': 58, 'pain_during_bowel_movements': 59, 'pain_in_anal_region': 60, 'bloody_stool': 61, 'irritation_in_anus': 62, 'neck_pain': 63, 'dizziness': 64, 'cramps': 65, 'bruising': 66, 'obesity': 67, 'swollen_legs': 68, 'swollen_blood_vessels': 69, 'puffy_face_and_eyes': 70, 'enlarged_thyroid': 71, 'brittle_nails': 72, 'swollen_extremeties': 73, 'excessive_hunger': 74, 'extra_marital_contacts': 75, 'drying_and_tingling_lips': 76, 'slurred_speech': 77, 'knee_pain': 78, 'hip_joint_pain': 79, 'muscle_weakness': 80, 'stiff_neck': 81, 'swelling_joints': 82, 'movement_stiffness': 83, 'spinning_movements': 84, 'loss_of_balance': 85, 'unsteadiness': 86, 'weakness_of_one_body_side': 87, 'loss_of_smell': 88, 'bladder_discomfort': 89, 'foul_smell_of urine': 90, 'continuous_feel_of_urine': 91, 'passage_of_gases': 92, 'internal_itching': 93, 'toxic_look_(typhos)': 94, 'depression': 95, 'irritability': 96, 'muscle_pain': 97, 'altered_sensorium': 98, 'red_spots_over_body': 99, 'belly_pain': 100, 'abnormal_menstruation': 101, 'dischromic _patches': 102, 'watering_from_eyes': 103, 'increased_appetite': 104, 'polyuria': 105, 'family_history': 106, 'mucoid_sputum': 107, 'rusty_sputum': 108, 'lack_of_concentration': 109, 'visual_disturbances': 110, 'receiving_blood_transfusion': 111, 'receiving_unsterile_injections': 112, 'coma': 113, 'stomach_bleeding': 114, 'distention_of_abdomen': 115, 'history_of_alcohol_consumption': 116, 'fluid_overload.1': 117, 'blood_in_sputum': 118, 'prominent_veins_on_calf': 119, 'palpitations': 120, 'painful_walking': 121, 'pus_filled_pimples': 122, 'blackheads': 123, 'scurring': 124, 'skin_peeling': 125, 'silver_like_dusting': 126, 'small_dents_in_nails': 127, 'inflammatory_nails': 128, 'blister': 129, 'red_sore_around_nose': 130, 'yellow_crust_ooze': 131}

 pred_des_list = {15: 'Fungal infection', 4: 'Allergy', 16: 'GERD', 9: 'Chronic cholestasis', 14: 'Drug Reaction', 33: 'Peptic ulcer diseae', 1: 'AIDS', 12: 'Diabetes ', 17: 'Gastroenteritis', 6: 'Bronchial Asthma', 23: 'Hypertension ', 30: 'Migraine', 7:
```

```
'Cervical spondylosis', 32: 'Paralysis (brain hemorrhage)', 28: 'Jaundice', 29: 'Malaria', 8: 'Chicken pox', 11: 'Dengue', 37: 'Typhoid', 40: 'hepatitis A', 19: 'Hepatitis B', 20: 'Hepatitis C', 21: 'Hepatitis D', 22: 'Hepatitis E', 3: 'Alcoholic hepatitis', 36: 'Tuberculosis', 10: 'Common Cold', 34: 'Pneumonia', 13: 'Dimorphic hemmorhoids(piles)', 18: 'Heart attack', 39: 'Varicose veins', 26: 'Hypothyroidism', 24: 'Hyperthyroidism', 25: 'Hypoglycemia', 31: 'Osteoarthristis', 5: 'Arthritis', 0: '(vertigo) Paroymsal  Positional Vertigo', 2: 'Acne', 38: 'Urinary tract infection', 35: 'Psoriasis', 27: 'Impetigo'}
  input_vector = np.zeros(len(symptoms_dict))
  for item in symptoms_experiance:
    input_vector[[symptoms_dict[item]]] = 1
  return pred_des_list[model.predict([input_vector])[0]]


# get diesease Description after model preductions
def get_diesese_practions(dieses):
    getprecaution = des.model.getPrecaution(dieses.capitalize())
    if disease_first[0][0] == dieses :
        getdescription = des.model.getDescription(dieses.capitalize())
        dieses =  dieses
    else:
        des_first = des.model.getDescription(disease_first[0][0].capitalize())
        des_sencond = des.model.getDescription(dieses.capitalize())
        getdescription = f"{des_sencond} \n\n {des_first}"
        dieses = f"{dieses} or {disease_first[0][0]}"


    output = f"You may have  {dieses} \n\n  {getdescription} \n  {getprecaution}\n"
    return output



# ============================================ RESPNSE SEND TO USER =======================================
# step 0
# send response to the user
```

```
def getresponse(response_msg):
    response_msg = response_msg.lower()
    try:
        response_nn = nltk_output(response_msg)
    except:
        response_nn = None
    if response_nn is not None:
        return [response_nn]
    else:
        output_function = tree_to_code(response_msg)
        if len(output_function) > 1 :
            return output_function[0] ,output_function[2][0]
        elif len(output_function) == 1 :
            return output_function

#Main Tk class
class ChatApplication:

    def __init__(self):
        self.window = Tk()
        self.outputs = []
        self.days = 0
        self._get_name()

    def run(self):
        self.window.mainloop()

    #------------------------------------------- start Window asking questions ---------------------
-----------------------
    # Print answer to the tk window
    def giveanswer(self,dises,ans):
        self.text_widget.tag_config('blue', foreground="#FDD20E")
        self.msg_entry.delete(0, END)
        sender = self.name_entry.get().split(" ")[0]
        msg1 = f"{sender} : "
        msg2 = f"{dises} -> {ans} \n"
        self.text_widget.configure(state=NORMAL)
        self.text_widget.insert(END, msg1.capitalize(), 'blue')
        self.text_widget.insert(END, msg2.capitalize())
        self.text_widget.configure(state=DISABLED)

    # asking function for how many days
    def _suffering_days(self):
        no_of_days = askstring(f"Please only respond in a number of days.", f"Since
how many days do you suffer? ")
        if no_of_days is None:
            self.msg_warning(f"Wrong Input ","Please respond only in number format. Do
not cancel because it's important.")
            self._suffering_days()
        elif no_of_days.isnumeric() is True and int(no_of_days) > 0:
            self.giveanswer("Since how many days do you suffer? ",no_of_days)
            self.days += int(no_of_days)
        else:
            self.msg_warning(f"Wrong Input ","Please respond only in number format. Do
not cancel because it's important.")
            self._suffering_days()



    # ask question to the user
    def ask_box(self,desid):
        prompt = askstring(f"Please respond only with (Yes/No)", f"Are you suffering
from a ' {desid} ' ? ")
        if prompt == None:
            self.msg_warning(f"{desid} Wrong Input ","Please respond only in (yes/no) format.Do not cancel because it's important.")

            self.ask_box(desid)

        elif prompt.lower() == "no" or prompt.lower() == "yes":

            self.giveanswer(desid,prompt)

            if prompt.lower() == "yes" :

                self.outputs.append(desid)

        else:

            self.msg_warning(f"{desid} Wrong Input ","Please give the answer only in (yes/no)")

            self.ask_box(desid)

    # ------------------------------------------End of ask question ----------------------------------------------



    # ============================================================ START OF TKINTER FIRST WINDOW ==========================================================

    #First tkinter window open and asking the name of user

    def _get_name(self):

        self.window.title("Mobile Doctor v.1.01")

        self.window.resizable(width=False, height=False)

        self.window.configure(width=500, height=500, bg=BG_COLOR)


        # tiny divider

        line = Label(self.window, width=450, bg=BG_GRAY)

        line.place(relwidth=1, y=0, relheight=0.002)


        # entry label

        self.name = Label(self.window, bg="RED", fg=TEXT_COLOR,

                      text="Please Enter Your name", font=FONT_BOLD, pady=10)

        self.name.place(relwidth=1,y=57.3)
        # message entry box level
        name_label = Label(self.window, bg=BG_COLOR, height=80)
        name_label.place(relheight=0.09, relwidth=1,y=105.9)
        #msg entery box
        self.name_entry = Entry(name_label, bg=BG_GRAY, fg="BLACK", font=FONT_BOLD)
        self.name_entry.place(relheight=0.85,relwidth=0.74, x=65)
        self.name_entry.focus()
        self.name_entry.bind("<Return>", self.get_name_after_click)

        #save name button label
        name_box_label = Label(self.window, bg=BG_COLOR, height=80)
        name_box_label.place(relheight=0.09, relwidth=1,y=150.9)
        #button
        name_send_button = Button(name_box_label, text="Submit", fg=TEXT_COLOR, font=FONT_BOLD, width=20, bg="RED",
                        command=lambda: self.get_name_after_click(None))
        name_send_button.place(relheight=1,relwidth=0.24, x=190)



        #------------------------------ menu button -----------------------------------
        menu = Menu(self.window,bg=BG_COLOR,borderwidth=0,fg=TEXT_COLOR,font="bold")
        self.window.config(menu=menu, bd=5)

        # File menu
        File = Menu(menu, tearoff=0,font="bold",activebackground="#FFFFFF")
        menu.add_cascade(label="File", menu=File,font="bold")
        File.add_command(label="Clear Chat",command=self.clear_chat,font="bold")
        File.add_command(label="Exit",command=None,font="bold")
        # About menu
        about = Menu(menu, tearoff=0,font="bold",activebackground="#FFFFFF")
        menu.add_cascade(label="About", menu=about,font="bold")


        # Quit menu
        menu.add_command(label ='Quit!',font="bold", command=lambda: self.msg_msg_askcancle(f"Ok Quit "," Are You sure? "))

    # ====================================================== END OF TKINTER FIRST WINDOW =======================================================

    # Name Validations
    def get_name_after_click(self,name):
        if len(self.name_entry.get()) >= 2:
            self._setup_main_window()
            self.coming_msg()
        else:
            self.msg_warning("Message Regarding Name Error","Please enter at least two words of a name.")


    #================================================= START MAIN WINDOW ============================================================
    def _setup_main_window(self):
        self.window.title("Mobile Doctor Welcomes You")
        self.window.resizable(width=False, height=False)
        self.window.configure(width=1200, height=640, bg=BG_COLOR)
        # tiny divider
        line = Label(self.window, width=450, bg=BG_GRAY)
        line.place(relwidth=1, y=0, relheight=0.012)

        #------------------- left side label -------------------------
        leftside_label = Label(self.window, bg=BG_COLOR, height=80,border=1)
        leftside_label.place(relheight=1, relwidth=0.2519,y=3 )

        # sidebox name lable
        self.name = Label(leftside_label, bg=BG_COLOR, fg=TEXT_COLOR,
                    text=self.name_entry.get().capitalize(),        font=FONT_BOLD,
pady=10)
        self.name.place(relheight=0.07, relwidth=1,y=1)

        # sidebox Help box
        self.help = Label(leftside_label, bg="red", fg=TEXT_COLOR,
                    text="Help Search (Similar diseases)", font=FONT_BOLD, pady=10)
        self.help.place(relheight=0.07, relwidth=1,y=48)

        # search box label
        search_label = Label(leftside_label, bg=TEXT_COLOR, height=80)
        search_label.place(relheight=0.07, relwidth=1,y=94)

        #Help message entry box
        self.help_entry    =    Entry(search_label,    bg=BG_GRAY,    fg="BLACK",
font=FONT_BOLD)
        self.help_entry.place(relheight=0.85,relwidth=0.74, y=2.4)
        self.help_entry.focus()
        self.help_entry.bind("<Return>", self._on_enter_help_search)

        # send button
```

```
help_send_button = Button(search_label, text="Search", font=FONT_BOLD,
width=20, bg=BG_GRAY,
                    command=lambda: self._on_enter_help_search(None))
help_send_button.place(relheight=0.85,relwidth=0.24,y=2.4, x=224)

#Searchbox
self.search_box = Text(leftside_label, width=20, height=2, bg="#00003d",
fg=TEXT_COLOR,
                    font=FONT, padx=8, pady=8)
self.search_box.place(relheight=.779, relwidth=1,y=140)
self.search_box.configure(cursor="arrow", state=DISABLED)
# scroll bar for search_box
scrollsearch = Scrollbar(self.search_box)
scrollsearch.place(relheight=1, relx=0.97)
scrollsearch.configure(command=self.search_box.yview)

#------------------------------ righ sider lebel -------------------------------
rightside_label = Label(self.window, bg=BG_COLOR, height=80,border=1)
rightside_label.place(relheight=1, relwidth=0.75,y=3 ,x=300)

# text widget
self.text_widget = Text(rightside_label, width=20, height=2, bg=BG_COLOR,
fg=TEXT_COLOR,
                    font=FONT, padx=5, pady=5)
self.text_widget.place(relheight=0.893, relwidth=1,y=0 ,x=0)
self.text_widget.configure(cursor="arrow", state=DISABLED)

# scroll bar
scrollbar = Scrollbar(self.text_widget)
scrollbar.place(relheight=1, relx=0.99)
scrollbar.configure(command=self.text_widget.yview)
```

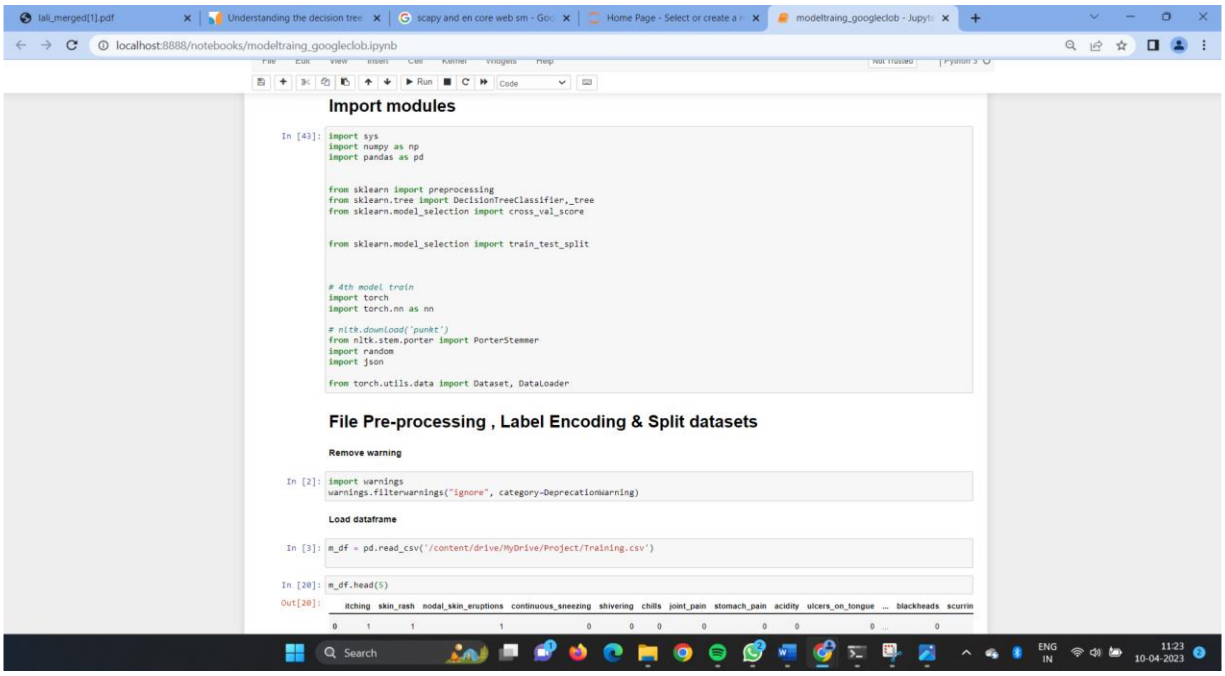
```
        #--------------------------------- bottom label ----------------------------------
        bottom_label = Label(rightside_label, bg="#17202A", height=80)
        bottom_label.place(relwidth=1,relheight=.1,x=1, y=563.7 )

        # message entry box #2C3E50
        self.msg_entry = Entry(bottom_label, bg=BG_GRAY, fg="Black", font=FONT)
        self.msg_entry.place(relheight=0.85,relwidth=0.82,x=5, y=4)
        self.msg_entry.focus()
        self.msg_entry.bind("<Return>", self._on_enter_pressed)

        # # send button
        send_button = Button(bottom_label, text="Send", font=FONT_BOLD, width=20,
bg=BG_GRAY,
                             command=lambda: self._on_enter_pressed(None))
        send_button.place(relheight=0.85, relwidth=0.15,x=745,y=4)

    #=============================================== END START
MAIN WINDOW
=========================================================

    # get search help Diesies
    def _on_enter_help_search(self,event):
        dis_list = ['itching', 'skin rash', 'nodal skin eruptions', 'continuous sneezing',
'shivering', 'chills', 'joint pain', 'stomach pain', 'acidity', 'ulcers on tongue', 'muscle
wasting', 'vomiting', 'burning micturition', 'spotting urination', 'fatigue', 'weight gain',
'anxiety', 'cold hands and feets', 'mood swings', 'weight loss', 'restlessness', 'lethargy',
'patches in throat', 'irregular sugar level', 'cough', 'high fever', 'sunken eyes',
'breathlessness', 'sweating', 'dehydration', 'indigestion', 'headache', 'yellowish skin',
'dark urine', 'nausea', 'loss of appetite', 'pain behind the eyes', 'back pain',
'constipation', 'abdominal pain', 'diarrhoea', 'mild fever', 'yellow urine', 'yellowing of
eyes', 'acute liver failure', 'fluid overload', 'swelling of stomach', 'swelled lymph
nodes', 'malaise', 'blurred and distorted vision', 'phlegm', 'throat 'stomach_bleeding',
'distention_of_abdomen', 'history_of_alcohol_consumption', 'fluid_overload.1',
'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']
        inp = self.help_entry.get()
        pred_list=[]
        if len(inp) > 0:
            regexp = re.compile(inp)
            for item in dis_list:
                if regexp.search(item):
                    pred_list.append(item)
        self.help_entry.delete(0, END)
        if len(pred_list) > 0:
            msg1 = ""
            for i in range(len(pred_list)):
                msg1 += f"{i+1})  {pred_list[i]} \n"
        else:
            msg1 = "Ohh!! There were no similar diseases discovered."
        self.search_box.configure(state=NORMAL)
        self.search_box.delete("1.0",END)
        self.search_box.insert(END, msg1)
        self.search_box.configure(state=DISABLED)


    # bot First welcome Message
    def coming_msg(self):
        good_name  = self.name_entry.get().split(" ")[0]
        msg = f"Hey! {good_name} , Are you not feeling well? Please tell me what symptoms here are some examples:\n --> fever\n --> cold\n --> cough\n --> headache\n --> stomach_pain\n --> abdominal_pain\n --> dehydration\n --> swelling\n --> acidity\n --> internal_itching\n --> sneezing\n --> vomiting\n --> anxiety , etc \n Note: Please use underscore (  _  ) in place of spacing in the name of disease.\n\n"
        # msg = f"Hey! {good_name} ,Are you not feeling well? Please tell me what symptoms here are some examples: \n fever , cold , cough , headache , stomach_pain , abdominal_pain , dehydration , swelling , acidity , \n itching , sneezing , vomiting , anxiety , etc.\nNote: Please use underscore( _ ) in place of spacing in the name of disease.\n\n"

        self._bot_insert(msg)


    #==================================== START RESPONSE MESSAGE =============================================

    # get the type msg from the entery of main msg box

    def _on_enter_pressed(self, event):

        msg = self.msg_entry.get()

        self._insert_message(msg)


    #get and send response message

    def _insert_message(self, msg):

        if not msg:

            return


        # delete the entery whtich is in entry box

        self.msg_entry.delete(0, END)


        # sender msg or user msg

        sender = self.name_entry.get().split(" ")[0]

        msg1 = f"{sender} : "

        msg2 = f"{msg}\n"


        #user insert msg

        self._user_instet_msg(msg1,msg2)


        # if we type any one from quit_msg then window close

        quit_msg = ["quit","exit","bye","bye bye"]
        if len([i for i in quit_msg if i == msg.lower()]) == 1:
            self.msg_msg_askcancle(f"Ok Quit "," Are You sure? ")


        # get msg form response from bot
        else:


            diesis_desc = {'(vertigo) paroymsal  positional vertigo': "Benign paroxysmal positional vertigo (BPPV) is one of the most common causes of vertigo — the sudden sensation that you're spinning or that the inside of your head is spinning. Benign paroxysmal positional vertigo causes brief episodes of mild to intense dizziness.",

 'Aids': "Acquired immunodeficiency syndrome (AIDS) is a chronic, potentially life-threatening condition caused by the human immunodeficiency virus (HIV). By damaging your immune system, HIV interferes with your body's ability to fight infection and disease.",

 'Acne': 'Acne vulgaris is the formation of comedones, papules, pustules, nodules, and/or cysts as a result of obstruction and inflammation of pilosebaceous units (hair follicles and their accompanying sebaceous gland). Acne develops on the face and upper trunk. It most often affects adolescents.',

'Varicose veins': 'A vein that has enlarged and twisted, often appearing as a bulging, blue blood vessel that is clearly visible through the skin. Varicose veins are most common in older adults, particularly women, and occur especially on the legs.',

 'hepatitis A': "Hepatitis A is a highly contagious liver infection caused by the hepatitis A virus. The virus is one of several types of hepatitis viruses that cause inflammation and affect your liver's ability to function."}

            diesis_prec = {

    "Durg reaction":"\n Take following measures : \n Stop taking Drug",

    "Malaria":" \n Consult nearest hospital \n avoid oily food \n avoid non veg food \n keep mosquitos out",

    "Allergy":" \n apply calamine \n cover area with bandage \n  use ice to compress itching ",

    "Hypothyroidism":"\n reduce stress \n exercise \n eat healthy \n get proper sleep",

    "Hepatitis e":"\n stop alcohol consumption\n rest\n consult doctor\n medication",

    "Dengue":"\n drink papaya leaf juice\n avoid fatty spicy food\n keep mosquitos away\n keep hydrated",

    "Hepatitis d":"\n consult doctor\n medication\n eat healthy\n follow up",

    "Heart attack":"\n call ambulance\n chew or swallow asprin\n keep calm\n ",
    "Pneumonia":"\n consult doctor\n medication\n rest\n follow up",
    "Arthritis":"\n exercise\n use hot and cold therapy\n try acupuncture\n massage",
    "Gastroenteritis":"\n stop eating solid food for while\n try taking small sips of
water\n rest\n ease back into eating",
    "Tuberculosis":"\n cover mouth\n consult doctor\n medication\n rest",
}

            chat_hear = getresponse(msg)
            if len(chat_hear) == 1 :
                msg4 = f"{chat_hear[0]} \n\n"
                self._bot_insert(msg4)
            elif len(chat_hear) > 1 :
                msg4 = f"{chat_hear[0]} \n \t Please give input on the diseases. \n\n"
                self._bot_insert(msg4)
                self.days *= 0
                self._suffering_days()
                self.outputs.clear()
                for i in chat_hear[1]:
                    self.ask_box(i)
                msg4 = f"You may also have diseases like \n"
                # if more than one output then condition true
                if len(self.outputs) > 0 :
                    for i in range(len(self.outputs)):
                        msg4 += f"\t {i+1} ) : {self.outputs[i]}\n"
                    msg4 += f"\n"
                else:
                    msg4 += f"\t 1 ) : {chat_hear[0]}\n\n"
                self._bot_insert(msg4)

                # no of days
                if self.days > 4:
                    self._bot_insert("Stop taking the medicine and reach out to the nearest hospital. \n")


                # get answer
                final_dieses = get_pridected_value(self.outputs)
                try:
                    diesese_is =  get_diesese_practions(final_dieses)
                except:
                    a=final_dieses
                    diesese_is = f"You are suffering with {a}\n\nAbout Disease : \n{diesis_desc[a]}  \nNo need to worry! \n You have to follow these precautions regarding this disease{diesis_prec[a]} \n"
                self._bot_insert(diesese_is)


                # open webbrowser when number of days is grater than 10
                if self.days > 7:
                    self._bot_insert("Stop taking the medicine and reach out to the nearest hospital.\n Click here and go to nearest hospital \n")
                    ask_to_web = tkinter.messagebox.askokcancel("Permission to open Google Maps", "Do you want to launch Google Maps in your regular browser?\n ")
                    if ask_to_web is True:

webbrowser.open_new_tab('https://www.google.com/maps/search/hospital+near+me/')
                        self._bot_insert(" Open Google Maps on your default web browser \n\n")
            else:
                pass


    # ========================================== END RESPONSE MESSAGE ==========================================================


    # clear chat box
```

```
def clear_chat(self):
    self.text_widget.config(state=NORMAL)
    self.text_widget.delete(1.0, END)
    self.text_widget.delete(1.0, END)
    self.text_widget.config(state=DISABLED)
    self.coming_msg()
# Insert msg from user
def _user_instet_msg(self,usr,msg):
    self.text_widget.tag_config('blue', foreground="#FDD20E")
    self.text_widget.configure(state=NORMAL)
    self.text_widget.insert(END, usr.capitalize(), 'blue')
    self.text_widget.insert(END, msg.capitalize())
    self.text_widget.configure(state=DISABLED)
#insert msg from bot
def _bot_insert(self,msg):
    self.text_widget.tag_config('red', foreground="#F93822")
    self.text_widget.configure(state=NORMAL)
    self.text_widget.insert(END, "Bot : ","red")
    self.text_widget.insert(END, msg)
    self.text_widget.configure(state=DISABLED)
    self.text_widget.see(END)
# ---------------------------------- Msg showing window --------------------------------
def msg_showinfo(self,title,msg):
    tkinter.messagebox.showinfo(title,msg)
def msg_warning(self,title,msg):
    tkinter.messagebox.showwarning(title,msg)
def msg_msg_askcancle(self,title,msg):
    msg_data = tkinter.messagebox.askokcancel(title, msg)
    if msg_data == True:
        self.window.destroy()
# ---------------------------------- end Msg showing window ----------------------------
```

```
if __name__ == "__main__":
    app = ChatApplication()
    app.run()
```

APPENDIX

B-SCREENSHOTS


lali_merged[1].pdf
Understanding the decision tree
scapy and en core web sm - Goo
Home Page - Select or create a n
modeltraing_googleclob - Jupyte
localhost:8888/notebooks/modeltraing_googleclob.ipynb
Run
Code
Search
ENG IN
11:23
10-04-2023

Import modules

```
In [43]: import sys
         import numpy as np
         import pandas as pd


         from sklearn import preprocessing
         from sklearn.tree import DecisionTreeClassifier,_tree
         from sklearn.model_selection import cross_val_score


         from sklearn.model_selection import train_test_split



         # 4th model train
         import torch
         import torch.nn as nn

         # nltk.download('punkt')
         from nltk.stem.porter import PorterStemmer
         import random
         import json

         from torch.utils.data import Dataset, DataLoader
```

File Pre-processing , Label Encoding & Split datasets

Remove warning

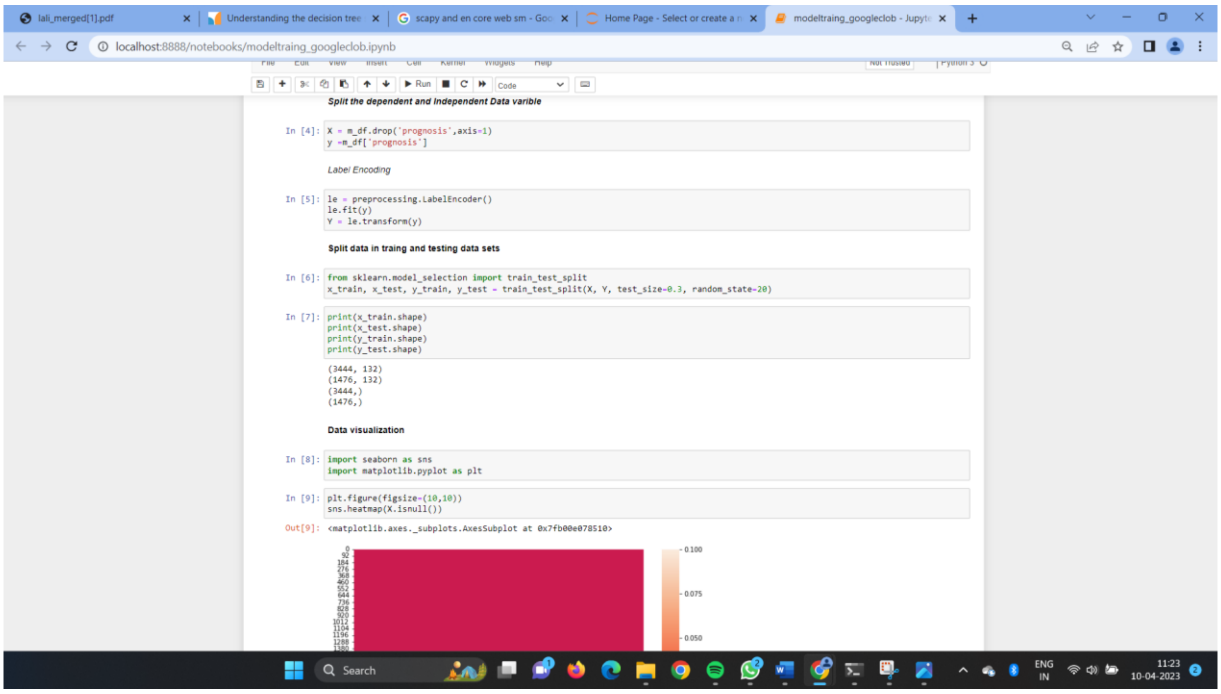
```
In [2]: import warnings
        warnings.filterwarnings("ignore", category=DeprecationWarning)
```

Load dataframe

```
In [3]: m_df = pd.read_csv('/content/drive/MyDrive/Project/Training.csv')
```

```
In [20]: m_df.head(5)
Out[20]:
     itching  skin_rash  nodal_skin_eruptions  continuous_sneezing  shivering  chills  joint_pain  stomach_pain  acidity  ulcers_on_tongue  ...  blackheads  scurrin
  0        1          1                     1                    0          0       0           0             0        0                 0  ...           0
```

Fig 6. Data Processing


lali_merged[1].pdf
Understanding the decision tree
scapy and en core web sm - Goo
Home Page - Select or create a n
modeltraing_googleclob - Jupyte
localhost:8888/notebooks/modeltraing_googleclob.ipynb
Run
Code
0
92
184
276
368
460
552
644
736
828
920
1012
1104
1196
1288
1380
0.100
0.075
0.050
Search
ENG IN
11:23
10-04-2023

Split the dependent and Independent Data varible

```
In [4]: X = m_df.drop('prognosis',axis=1)
        y =m_df['prognosis']
```

Label Encoding

```
In [5]: le = preprocessing.LabelEncoder()
        le.fit(y)
        Y = le.transform(y)
```

Split data in traing and testing data sets

```
In [6]: from sklearn.model_selection import train_test_split
        x_train, x_test, y_train, y_test = train_test_split(X, Y, test_size=0.3, random_state=20)
```

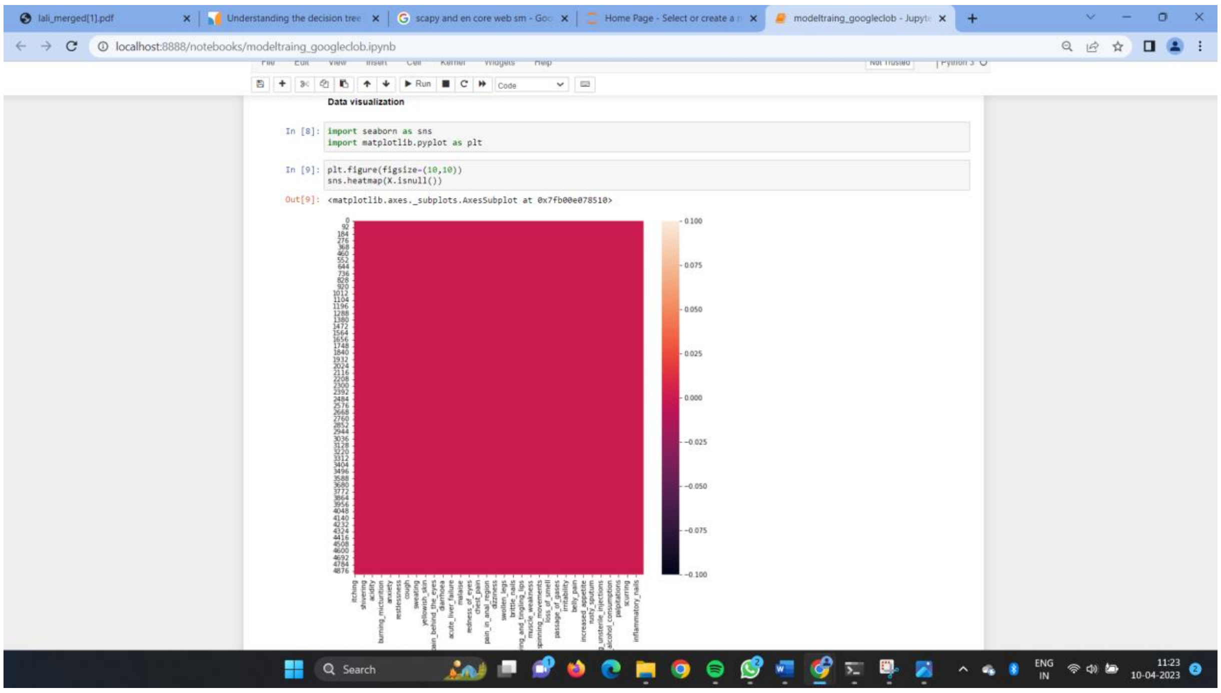
```
In [7]: print(x_train.shape)
        print(x_test.shape)
        print(y_train.shape)
        print(y_test.shape)

        (3444, 132)
        (1476, 132)
        (3444,)
        (1476,)
```

Data visualization

```
In [8]: import seaborn as sns
        import matplotlib.pyplot as plt
```

```
In [9]: plt.figure(figsize=(10,10))
        sns.heatmap(X.isnull())
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb00e078510>
```

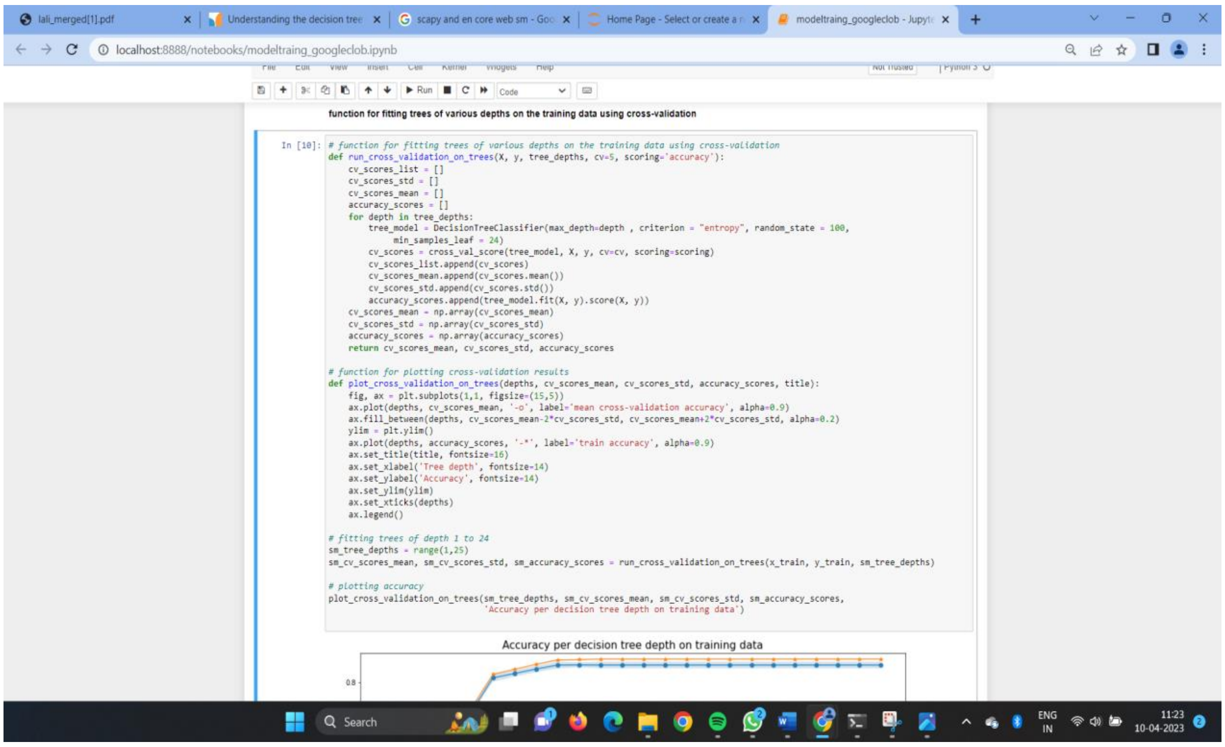
Fig 7. Data preprocessing



Data visualization

```
In [8]: import seaborn as sns
        import matplotlib.pyplot as plt

In [9]: plt.figure(figsize=(10,10))
        sns.heatmap(X.isnull())

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x7fb00e078510>
```

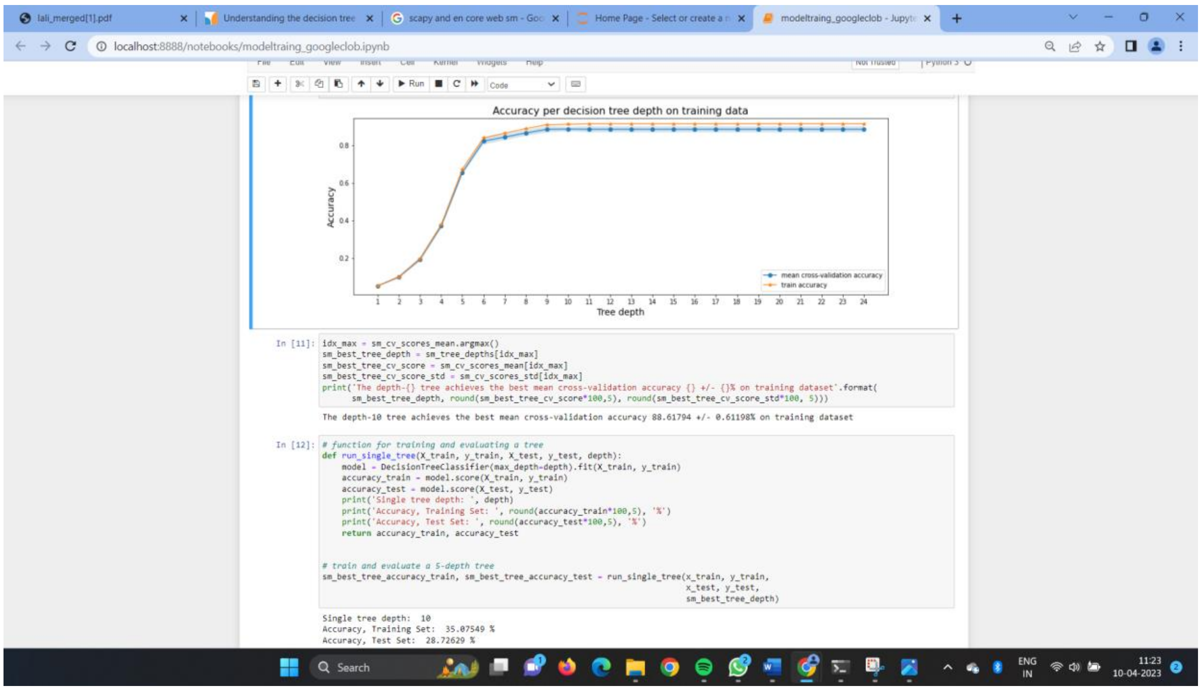
Fig 8 : Heatmap



function for fitting trees of various depths on the training data using cross-validation

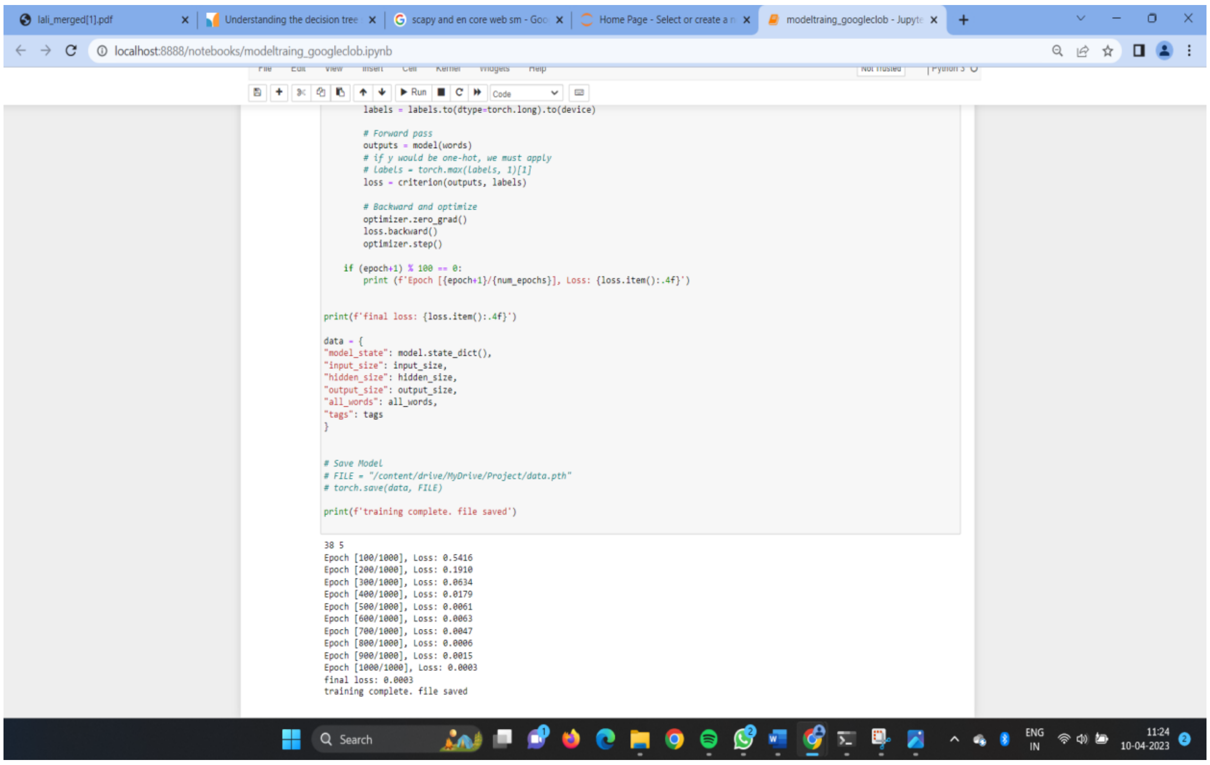
```
In [10]: # function for fitting trees of various depths on the training data using cross-validation
         def run_cross_validation_on_trees(X, y, tree_depths, cv=5, scoring='accuracy'):
             cv_scores_list = []
             cv_scores_std = []
             cv_scores_mean = []
             accuracy_scores = []
             for depth in tree_depths:
                 tree_model = DecisionTreeClassifier(max_depth=depth , criterion = "entropy", random_state = 100,
                      min_samples_leaf = 24)
                 cv_scores = cross_val_score(tree_model, X, y, cv=cv, scoring=scoring)
                 cv_scores_list.append(cv_scores)
                 cv_scores_mean.append(cv_scores.mean())
                 cv_scores_std.append(cv_scores.std())
                 accuracy_scores.append(tree_model.fit(X, y).score(X, y))
             cv_scores_mean = np.array(cv_scores_mean)
             cv_scores_std = np.array(cv_scores_std)
             accuracy_scores = np.array(accuracy_scores)
             return cv_scores_mean, cv_scores_std, accuracy_scores

         # function for plotting cross-validation results
         def plot_cross_validation_on_trees(depths, cv_scores_mean, cv_scores_std, accuracy_scores, title):
             fig, ax = plt.subplots(1,1, figsize=(15,5))
             ax.plot(depths, cv_scores_mean, '-o', label='mean cross-validation accuracy', alpha=0.9)
             ax.fill_between(depths, cv_scores_mean-2*cv_scores_std, cv_scores_mean+2*cv_scores_std, alpha=0.2)
             ylim = plt.ylim()
             ax.plot(depths, accuracy_scores, '-*', label='train accuracy', alpha=0.9)
             ax.set_title(title, fontsize=16)
             ax.set_xlabel('Tree depth', fontsize=14)
             ax.set_ylabel('Accuracy', fontsize=14)
             ax.set_ylim(ylim)
             ax.set_xticks(depths)
             ax.legend()

         # fitting trees of depth 1 to 24
         sm_tree_depths = range(1,25)
         sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores = run_cross_validation_on_trees(x_train, y_train, sm_tree_depths)

         # plotting accuracy
         plot_cross_validation_on_trees(sm_tree_depths, sm_cv_scores_mean, sm_cv_scores_std, sm_accuracy_scores,
                                        'Accuracy per decision tree depth on training data')
```

Fig :9. Model building


Accuracy per decision tree depth on training data
Accuracy
Tree depth
mean cross-validation accuracy
train accuracy

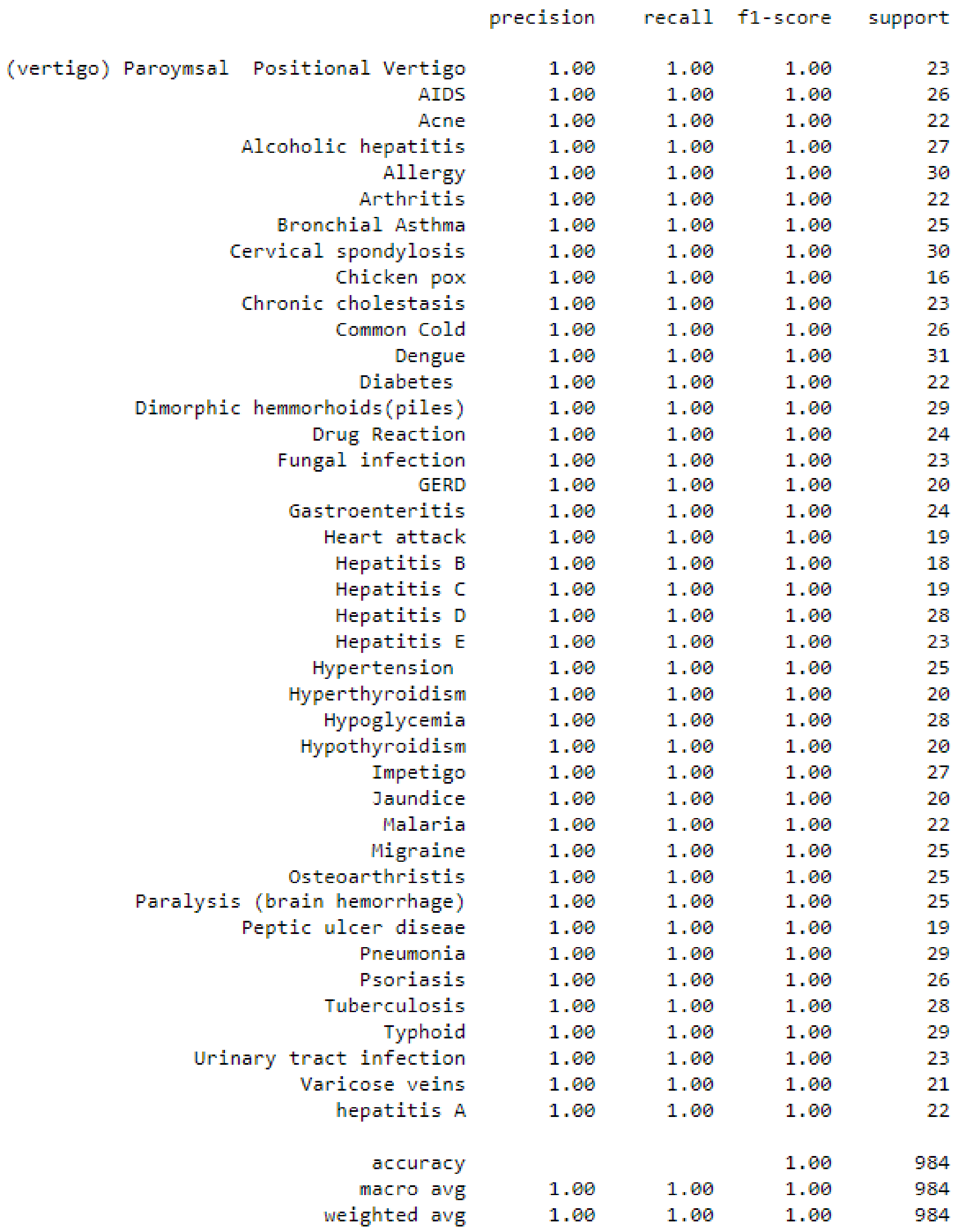
```
In [11]: idx_max = sm_cv_scores_mean.argmax()
         sm_best_tree_depth = sm_tree_depths[idx_max]
         sm_best_tree_cv_score = sm_cv_scores_mean[idx_max]
         sm_best_tree_cv_score_std = sm_cv_scores_std[idx_max]
         print('The depth-{} tree achieves the best mean cross-validation accuracy {} +/- {}% on training dataset'.format(
               sm_best_tree_depth, round(sm_best_tree_cv_score*100,5), round(sm_best_tree_cv_score_std*100, 5)))

         The depth-10 tree achieves the best mean cross-validation accuracy 88.61794 +/- 0.61198% on training dataset

In [12]: # function for training and evaluating a tree
         def run_single_tree(X_train, y_train, X_test, y_test, depth):
             model = DecisionTreeClassifier(max_depth=depth).fit(X_train, y_train)
             accuracy_train = model.score(X_train, y_train)
             accuracy_test = model.score(X_test, y_test)
             print('Single tree depth: ', depth)
             print('Accuracy, Training Set: ', round(accuracy_train*100,5), '%')
             print('Accuracy, Test Set: ', round(accuracy_test*100,5), '%')
             return accuracy_train, accuracy_test


         # train and evaluate a 5-depth tree
         sm_best_tree_accuracy_train, sm_best_tree_accuracy_test = run_single_tree(x_train, y_train,
                                                                                   x_test, y_test,
                                                                                   sm_best_tree_depth)

         Single tree depth:  10
         Accuracy, Training Set:  35.07549 %
         Accuracy, Test Set:  28.72629 %
```

Fig 10: Testing



```
            labels = labels.to(dtype=torch.long).to(device)

            # Forward pass
            outputs = model(words)
            # if y would be one-hot, we must apply
            # labels = torch.max(labels, 1)[1]
            loss = criterion(outputs, labels)

            # Backward and optimize
            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        if (epoch+1) % 100 == 0:
            print (f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}')


print(f'final loss: {loss.item():.4f}')

data = {
"model_state": model.state_dict(),
"input_size": input_size,
"hidden_size": hidden_size,
"output_size": output_size,
"all_words": all_words,
"tags": tags
}


# Save Model
# FILE = "/content/drive/MyDrive/Project/data.pth"
# torch.save(data, FILE)

print(f'training complete. file saved')

38 5
Epoch [100/1000], Loss: 0.5416
Epoch [200/1000], Loss: 0.1910
Epoch [300/1000], Loss: 0.0634
Epoch [400/1000], Loss: 0.0179
Epoch [500/1000], Loss: 0.0061
Epoch [600/1000], Loss: 0.0063
Epoch [700/1000], Loss: 0.0047
Epoch [800/1000], Loss: 0.0006
Epoch [900/1000], Loss: 0.0015
Epoch [1000/1000], Loss: 0.0003
final loss: 0.0003
training complete. file saved
```

Fig 11 : Pickle File



```
                                         precision    recall  f1-score   support

(vertigo) Paroymsal  Positional Vertigo       1.00      1.00      1.00        23
                                   AIDS       1.00      1.00      1.00        26
                                   Acne       1.00      1.00      1.00        22
                    Alcoholic hepatitis       1.00      1.00      1.00        27
                                Allergy       1.00      1.00      1.00        30
                              Arthritis       1.00      1.00      1.00        22
                       Bronchial Asthma       1.00      1.00      1.00        25
                   Cervical spondylosis       1.00      1.00      1.00        30
                            Chicken pox       1.00      1.00      1.00        16
                    Chronic cholestasis       1.00      1.00      1.00        23
                            Common Cold       1.00      1.00      1.00        26
                                 Dengue       1.00      1.00      1.00        31
                              Diabetes        1.00      1.00      1.00        22
           Dimorphic hemmorhoids(piles)       1.00      1.00      1.00        29
                          Drug Reaction       1.00      1.00      1.00        24
                       Fungal infection       1.00      1.00      1.00        23
                                   GERD       1.00      1.00      1.00        20
                        Gastroenteritis       1.00      1.00      1.00        24
                           Heart attack       1.00      1.00      1.00        19
                            Hepatitis B       1.00      1.00      1.00        18
                            Hepatitis C       1.00      1.00      1.00        19
                            Hepatitis D       1.00      1.00      1.00        28
                            Hepatitis E       1.00      1.00      1.00        23
                          Hypertension        1.00      1.00      1.00        25
                        Hyperthyroidism       1.00      1.00      1.00        20
                           Hypoglycemia       1.00      1.00      1.00        28
                         Hypothyroidism       1.00      1.00      1.00        20
                               Impetigo       1.00      1.00      1.00        27
                               Jaundice       1.00      1.00      1.00        20
                                Malaria       1.00      1.00      1.00        22
                               Migraine       1.00      1.00      1.00        25
                        Osteoarthristis       1.00      1.00      1.00        25
           Paralysis (brain hemorrhage)       1.00      1.00      1.00        25
                    Peptic ulcer diseae       1.00      1.00      1.00        19
                              Pneumonia       1.00      1.00      1.00        29
                              Psoriasis       1.00      1.00      1.00        26
                           Tuberculosis       1.00      1.00      1.00        28
                                Typhoid       1.00      1.00      1.00        29
                Urinary tract infection       1.00      1.00      1.00        23
                         Varicose veins       1.00      1.00      1.00        21
                            hepatitis A       1.00      1.00      1.00        22

                               accuracy                           1.00       984
                              macro avg       1.00      1.00      1.00       984
                           weighted avg       1.00      1.00      1.00       984
```

Fig 12: Analysis Report of Data Set

DISEASE PREDICTION SYSTEM AND HOSPITAL RECOMMENDER

Padava Kishore Chandra Sai
UG Scholar
Dept. of CSE
Sathyabama Institute of Science and Technology
Chennai, India

Menaka D
Associate Professor
Dept. of CSE
Sathyabama Institute of Science and Technology
Chennai, India

ABSTRACT* - Human health is a fundamental component of society, and timely and accurate disease diagnosis is critical to giving patients the care they need. Yet, due to the intricacy of medical research, this necessitates the cooperation of other professions, Mathematics and computer programming are two examples. These established disciplines' are challenged by the emergence by fresh perspectives in medical research, yet the variety of these techniques allows for a thorough assessment of various factors. In this regard, it is suggested to conduct a systematic study of machine learning applications in medical diagnosis of human disorders. The study focuses on current advances in machine learning that can be used to spot interesting patterns, generate precise forecasts, and support decision-making. The study focuses on current advances in machine learning that can be used to spot interesting patterns, generate precise forecasts, and support decision-making. In order to offer proper decision support, this article introduces a special collection of machine learning methods. It seeks to fill the knowledge void in the creation of useful decision support systems for medical applications.

Keywords: Chatbot, classification, disease detection, Hospital Recomender.

I.INTRODUCTION

If someone not feeling well and wants to see a doctor, he must go to a medical facility and wait for a long time until the doctor is accessible. In addition, the patient must stand in order. If the doctor cancels the appointment for any reason, the patient will be unable to seek for cancellation because he will be unable to visit the sanatorium and observe the group. His or her decision. As tissue improvement technology advances, any technique of tissue improvement may be used to overcome the issues that the patient faces prior to confirming the diagnosis. We can wait or spend some time inside the doctor's interview if we have an appointment with the right doctor. We waste a lot of money and effort as a result of this. We'll make a lot of these problems in one step for your palms and before your eyes.

First, we create a user experience that is both easy to use and simple to execute. The shape could be at the entrance, and our mobile doctor could round out the appointment. The Doctor Chatbot interface will be displayed after finishing the questionnaire. We must submit the signs and symptoms and requests to the doctor, and the doctor will test our signs and symptoms and signs and test the pattern, and he will infer the disease that you may be attacked by, and he will indicate the disease with accuracy, the disease description, and precautions. And, regardless of the severity of the illness, any diagnosis of the illness you are presently suffering from will provide you with information. Furthermore, this project has many highly promising futures, and we're presenting this replacement on this one, and this challenge is live interface, which is everywhere and at all times, and the person can use it for free. On this trip, the help vector machine algorithm was used. (SVM). As a result, we used vector algorithm assistance for this collection of rules. We use the box framework in this project to build a person interface that interacts with customers through the use of the interface. We bring Google Cloud web hosting to this website, which allows you to access it from anywhere.

II . LITERATURE SURVEY

The primary four articles that address the scientific area are included in the literature review. They should attempt to establish the significance of the various characteristics and their common meanings . While, from a scientific standpoint, system learning algorithms aren't used to identify diseases and are thus no longer covered within the evaluation. The following literature review of these four articles is within the field of engineering, where distinct characteristics are taken into account and unique device learning algorithms are used to automatically detect the presence of disease. This problem formula has been meticulously studied over time by researchers and carnivore companies in an attempt to find a solution, and all of their solutions range from the analysis of various types of algorithms to the detection of various diseases in the human body.

Let us now move on to the prevalent disease prediction ghavane . An artificial learning-based device that predicts same illnesses is proposed. The sign dataset was gathered from the UCI, where it comprised symptoms of numerous same illnesses To check variety diseases, the system used CN Networks, KN-Neighbours as category methods. Furthermore, the suggested solution was supplemented with additional information about the existence of the examined patient, which proved used in determine the extent off danger connected with antiseptic disorder. Dahivade et al. [9] distinguished effects of KN-Neighbours and C-Neural N algorithms on processing speed and accuracy. The C-Nearest N's accuracy and processing speed increased to eightfive% and eleven seconds, respectfully. Statistics showed the K Nearest N combination of rules are no longer as effective as the C-neural N algorithm. Small amount of this examination, the results also demonstrated that CN-Networks outperforms. C-N-N algorithms such as K-Nearest N, NaiveBayes, and DTrees. The writters showed the invented model's accuracy in terms of accurate, which are defined as version's ability to identify complex non-linear relationships within the characteristic space. Furthermore, RNA identifies essential functions that provide a more detailed description of the illness, making it feasible to treat illnesses of the highest complexity [9], [10]. Underforming algorithms, for example, can be carried out with larger variable functions.

Now we'll see about the muscle discomfort.Low again .Doctors must establish a clinical prognosis in order to provide suitable treatment for lower spinal pain. AI models, which are used in the clinical field to detect illnesses, help doctors identify illnesses based on early symptoms. This study aims to show the primary parameters that give to spinal anomalies and detect spinal anomalies by the data gathered in the country of the backbone body using a machine learning approach that includes Principal Analysis (PCA) as well as supervisors. Is drawing close gadget learning which includes K_Nearest Neighbors (KNN) and Random_Forest (RF). As a result, the degree of spondylolisthesis is the most important factor affecting the spinal cord anomalyWhen the effects of the R_F classifier and the K_Nearest N classifier were compared, the KNN classifier outperformed the RandomForest classification because the KNN algorithm's accuracy rate (85.32%) is higher than the RF classifier's (79.57%).

Now we will look at how marimuthu attempted to predict heart disease through the leadership of gadget studying methods. The writers examined data that included race, age, breast pain, intercourse, target, and slope. The following ML algorithms have been created: DT, KNN, LR, and NB. When compared to the other algorithms mentioned, the LR algorithm offered an extremely high accuracy of 86.89%, according to the research. In 2020, he tries to get more accuracy of predicting coronary heart attack by incorporating extra parameters such as rest of BP, serum_fat levels in mg/dl, and height heart beats. The information get from the UCI Datasets Dwivedi tried to evaluate (ANNs), support vector machines (SVMs), KNNs, NBs, LRs, and tree type. The findings of triple cross-validation show that Logistic Reg has the more category accurate and sensitivity, as well as the highest reliability in detecting illnesses [9]. Statistics from some diseases where logistic regression has evolved the use of other methods such as ANN, SVM, and Adaboost corroborate this conclusion. There has been research into the assessment of large gadget learning models. Exclusive hyperparameters.

A. INFERENCES FROM LITERATURE SURVEY

Based on my literature review, I came up with the idea of developing a completely computerised system to predict any illness and suggest a model of the most essential treatment based on machine learning algorithms for the obtained illness, which were picked up and provided from the signs and symptoms. The consumer. So, using various algorithmic strategies such as AdaBoost, Support Vector Machine, and others, I'm going to build an application that can instantly predict any illness.

III. EXISTING SYSTEM

Existing systems are either predicting the disease with poor accuracy or determining the disease's accuracy. The following are the main issues that users have with the current system. The sum of record structures is a common issue. A common issue with all existing systems is that they cannot detect someone who is ill. This sickness detection is carried out by utilising a pre-existing dataset and gathering all of the data from the dataset. The method we use may also influence the outcome. Although the results are stated, each has its own estimate of accuracy.They also neglected to provide treatment for the injury. Finally, existing algorithms are the best at predicting a specific disease.

IV.REQUIREMENT , RISK ANALYSIS AND FEASIBILE STUDY OF THE PROJECT

• In this level, the feasibility of the venture is analyzed and a decision is made with the maximum not unusual layout and a few value estimates. In the analysis of the machine, it's far vital to carry out a feasibility observe of the proposed system. This is to ensure that the created system will not be a burden to the company. Feasibility research require more understanding of the initial machine necessaries.

• Feasibility have a look at explores the effects and information wishes of stakeholders. It seeks to decide the sources had to provide an statistics device solution, the cost and benefits of this sort of answer and the feasibility of one of these solution.

• The purpose of the feasibility look at is to recall other answers for the records device, to evaluate their feasibility, and to advise the most appropriate opportunity to the corporation. The feasibility of the proposed solution is taken into consideration according to its additives.

A. ECONOMICAL FEASIBILITY **:**

We have a look at performed to check the financial system that the machine can had the business enterprise. The addition of amount a agency spends on studies and machine improvement can be restricted. To justify the price. Thus, the machine is advanced even on a budget, and that is due to the truth that maximum of the technologies are available totally free.

It changed into simplest important to buy person products.

B. TECHNICAL FEASIBILITY :

This examine is performed to check the technical abilties, that is, the technical elements. Any system advanced ought to now not place excessive demands on the technical assets used. This will cause high needs at the technical assets hired. Here, the patron is looking you to herald a excessive-profile photograph. A gadget in development ought to have modest requirements, on the grounds that best minimum or no changes are required to enforce this device.

C. SOCIAL FEASIBILITY

The studies aspect is conclude the stage of reputation of the model by means of person. This shows the system of conveying the consumer how to did the device successfully. Any person does no longer should recollect the chance, however ought to take delivery of the need.

V.SOFTWARE REQUIREMENTS

➢ Operating System	: Windows 8+
➢ Processor	: Intel/Ryzen
➢ Hard Disk	: 256 GB
➢ Ram	: 4 GB
➢ Coding Languages	: Python
➢ Libraries Used	: Tkinter, Sciket Learn

VI. PROPOSED SYSTEM

Considering the failure of all of the methods used within the preceding systems, the most not unusual question was selecting the nice algorithm. For one facts set, if one algorithm affords higher accuracy than another data set, some other facts set gives higher accuracy than any other set of rules. So I got here up with a option to keep away from manual statistics entry and trying out. Here we're going to take accurate attributes from the user (patient) and predict if the patient has any sickness or not. And so it is possible to help them keep their lives or even drugs in line with the disorder. As an brought feature, the chatbot can even upload synthetic intelligence for better treatment and analysis for the person.

A. ADVANTAGES FOR PROPOSED SYSTEM:

1. Increased accuracy in diagnosing sicknesses.

2. Reducing the time spent by using doctors.

3. Patient-pleasant charges.

4. They can be accessed from anywhere.

5. What all people can have enough money.

6. Scalable time complexity.

VII. IMPLEMENTATION TECHNIQUES

A. Data Source -This dataset uses disorder predictions from the UCI internet site, Kaggle, and everyday datasets. UCI is a hard and fast of databases used for machine studying algorithms. The information posted right here is real records. We typically use five datasets on this task.

1. Dataset for trying out and training, which include extra than 4000 rows.
2. The statistics description machine is hooked up, which includes 40 rows, in each row a description of the sickness is defined.
3. Systemic prevention. A set of statistics that gives to be careful that we are attacked with the aforementioned ailment.
4. Set the severity statistics gadget that suggests the severity of the disorder expected.

B. DESCRIPTION OF METHODS AND ALGORITHMS:
SUPPORT VECTOR MACHINE (SVM) :

Support Vector Classifier or SV Machine is one of the maximum famous predictive studying algorithms this is used for each category and regression issues. But it changed into in the main used for classification issues in device gaining knowledge of. The motive of the SVM algo is to show an superior selection line or border that could divide a space of n dimensions into lessons so that we are able to effortlessly region a new point in the right class inside the destiny. This best solution is called the boundary plane.

SVC will select the bounds/vectors that creating the plane. The facet cases are known as aid vector machines, subsequently the set of rules is called a aid vector device.

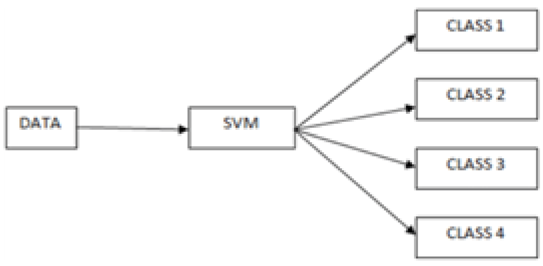


Figure 1: Support Vector Machine

Example: SVM can be understood from the instance utilized in KNN classifier. We will first install our version with many snap shots of Tom and Puppies in order that can recognize other strains of Toms and puppies, then can check it on a new creature. When the help vector creates a selection among information (cat and canine) and selects intense cases (aid vector), it will see cat and canine as extreme cases. He leans at the carrier for assist, and describes it as a cat.

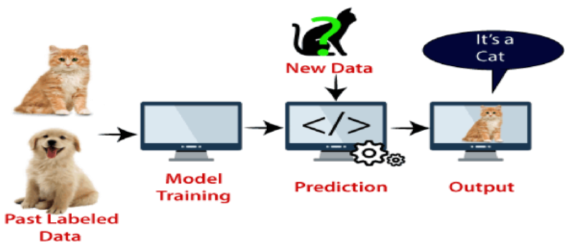


Figure 2: Supervised Machine Learning

Above figure 2 is the example of The SVC algorithm's supervised algorithm for machine learning classification can be applied to face recognition, classification of pictures, categorization of text, and other tasks.

- Linear Machine that is used for straight separable facts, this is, if the data can be splited into 2 lessons via an unmarried immediately border, such records are called linearly separable statistics, and the classification is help as a straight SVM classifier. . .
- Non-linear-Machine is for non-straight-separated information, if the records set is located on immediately, then the records is not straight.

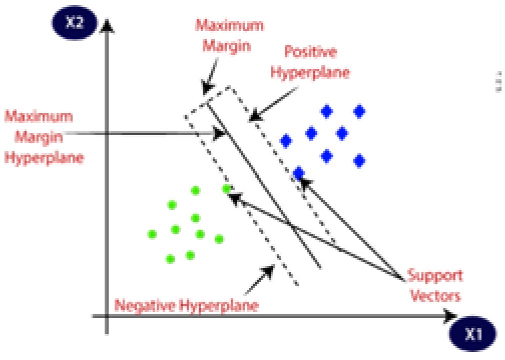


Figure 3: SVM Classification

Hyperplane is a Multiple choice traces/barriers may be used to divide instructions in dimensional area, however it's far important to locate the first-rate defining boundary that enables to identify the facts points. This is nice referred to as the hyperplane sure SVM.

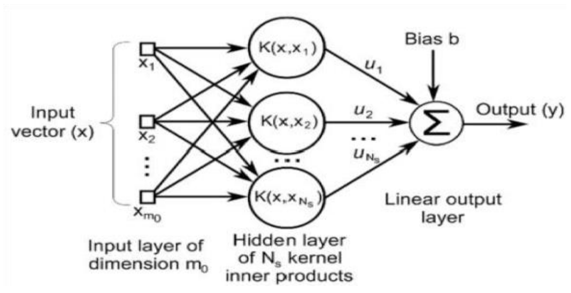


Figure 4: SVM Architecture

Support Vectors:

Points or statistics vectors which might be near the plane and effect the region of the plane are said as aid points. Because those companies assist the hyperplane, they're known as aid providers.

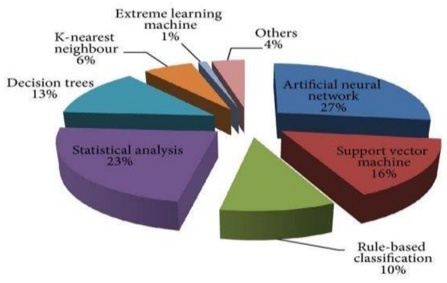


Figure 5: Machine Learning Algorithms

From the above image figure 4 we can see the importance of the support vector machine and decision trees in machine learning models.

C. PROPOSED WORK :

In this project, we used tkinter to build the user interface for graphics and integrated our model into it. We trained our model using datasets obtained from UCI and Kaggle, and we built a user-friendly user interface using tkinter and Python. We linked our model to the interactive user interface. We are now chaining our Python file to.exe format for customers to use in offline mode.

Our model will accept the user's inputs and add each symptom to a list, and based on each symptom, our model will check each list-added disease using decision trees, and based on the list, our model will predict the disease using support vector model with accurate of ninety seven and half %.

D. SYSTEM ARCHITECTURE :

The below architecture demonstrates that a consumer can get entry to an software on an internet server and be capable of predict if they have a sickness. Data pre-processing and segmentation might be done within the history. So the usage of this utility can provide an concept of the following step. As an additional feature, we will additionally add an artificial intelligence chatbot with a purpose to advocate excellent medicinal drugs and disease prognosis.

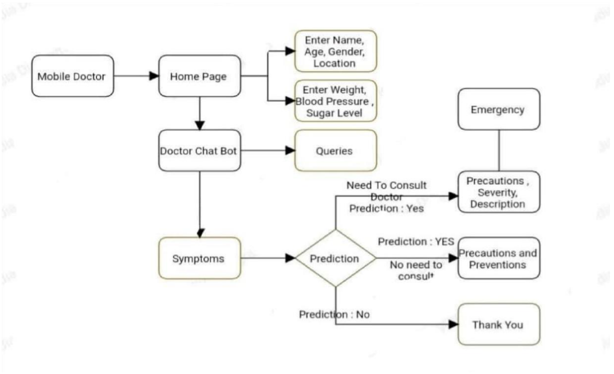


Figure 6: Architecture Diagram

From the figure 6, it describes about whole working model architecture of the system.

E. DESCRIPTION OF SOFTWARE IMPLEMENTED , TEST PLAN OF THE PROPOSED SYSTEM :

The Mobile Doctor , is created using Tkinter as frontend and Python machine learning as backend. In this model initially we can have to enter our name and nextly the mobile doctor will going open there. This User Interface will ask the respected sysmptoms facing by the user and he will give the sysmptoms he is facing. The respected symptoms will ask in that UI. We need to add the symptoms in the popups. After giving all the symptoms , our model will predict the disease ,and after it is danger means our model will give the popup to open the google maps and showing the respected nearest hospitals near the user location.

VIII. RESULT

Overall, the disease detection model performs data and the machine learning method that’s the systems'. Because the reasoning behind each decision is provided, decision trees are relatively easy models to comprehend, resulting in an intuitive framework comprehension. Under this paradigm, knowledge models may be immediately changed to some IF-Then rules, which are among the most prevalent types of knowledge representation. This model will help the user with best accuracy. Our model will predict the disease with 97.5% accurate.

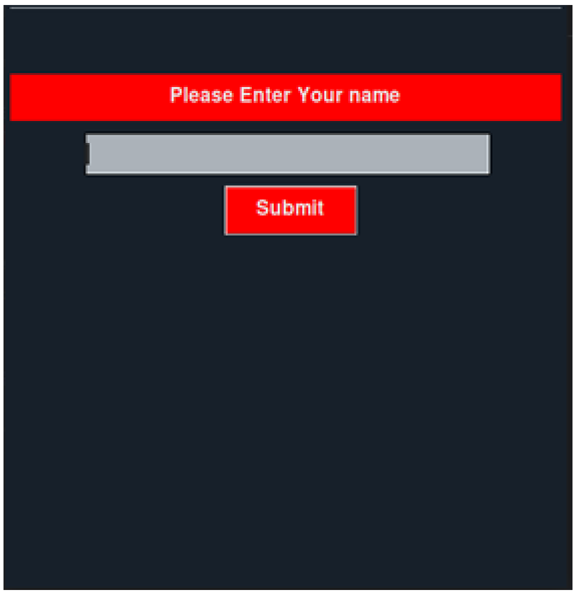
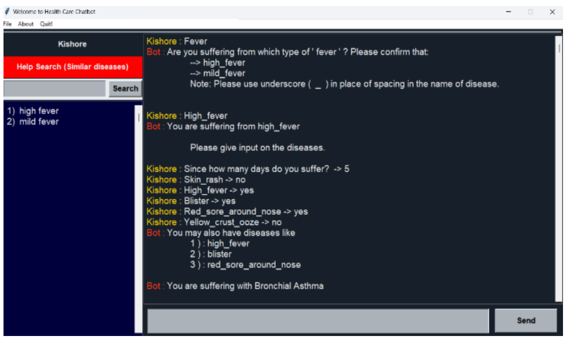


Figure 7 : Intial Window

There is a beginning entry window Figure 7 shows the image of that. Beginning and next we have to add our name in that and next our model will open and we had to give our symptoms facing and next model will predict the disease and shows the precautions we had to follow and if it is very dangerous stage means it will show the nearest Hospital in google maps.


Welcome to Health Care Chatbot
File About Quit!
Kishore
Help Search (Similar diseases)
Search
Send

```
1) high fever
2) mild fever
```

```
Kishore : Fever
Bot : Are you suffering from which type of ' fever ' ? Please confirm that:
        --> high_fever
        --> mild_fever
        Note: Please use underscore ( _ ) in place of spacing in the name of disease.


Kishore : High_fever
Bot : You are suffering from high_fever

        Please give input on the diseases.

Kishore : Since how many days do you suffer?  -> 5
Kishore : Skin_rash -> no
Kishore : High_fever -> yes
Kishore : Blister -> yes
Kishore : Red_sore_around_nose -> yes
Kishore : Yellow_crust_ooze -> no
Bot : You may also have diseases like
        1 ) : high_fever
        2 ) : blister
        3 ) : red_sore_around_nose

Bot : You are suffering with Bronchial Asthma
```

Figure 8 : Model Window

If the user suffering with the disease from or more than 7 days means our model will ask the popup and show the nearest hospitals near the user who is using our user interface.

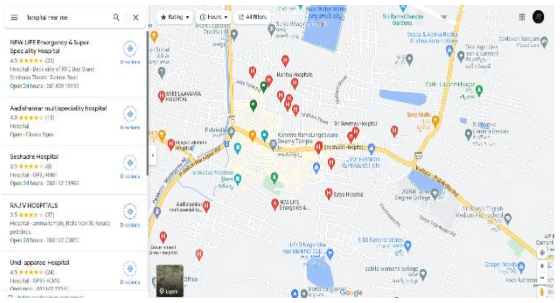


Figure 9 : Nearest Hospital Detector

In Figure 9, model show the nearest hospital based on the disease severity and number of days suffering by user.

IX. CONCLUSION

This model covered the causal illness are the common diseases, In future for this model, we will going to add all high range diseases, especially such a various types cancers. so that prior conditions could be gaining immediately to the user receiving medical-attention. The healthy life rate for all cancers gradually increases if model or system detecting it in the prior stage itself and medication is given immediately. The model will in future will be improves by adding different wide verity symptoms and help to grow the no: of ways for training the model and testing the model. And we will add some other diseases and make our model to use by everyone at anywhere.

REFERENCES

[1] A. Gavhane, G. Kokkula, I. Pandya, and K. Devadkar, "Prediction of heart disease using machine learning," in 2018 Second International Conference on Electronics, Communication and Aerospace Technology (ICECA), 2018, pp. 1275–1278.

[2] Y. Hasija, N. Garg, and S. Sourav, "Automated detection of dermatological disorders through image-processing and machine learning," in 2017 International Conference on Intelligent Sustainable Systems (ICISS), 2017, pp. 1047–1051.

[3] S. Uddin, A. Khan, M. E. Hossain, and M. A. Moni, "Comparing different supervised machine learning algorithms for disease prediction," BMC Medical Informatics and Decision Making, vol. 19, no. 1, pp. 1– 16, 2019.

[4] R. Katarya and P. Srinivas, “Predicting heart disease at early stages using machine learning: A survey,” in 2020 International Conference. Electronics and sustainable Communication Systems(ICESC),2020,pp 302-305.

[5] P. S. Kohli and S. Arora, “Application of machine learning in disease prediction,” in 2018 4th International Conference on Computing Communication and Automation (ICCCA), 2018, pp. 1–4.

[6] M. Patil, V. B. Lobo, P. Puranik, A. Pawaskar, A. Pai, and R. Mishra, “A proposed model for lifestyle disease prediction using support vector machine,” in 2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2018, pp. 1–6.

[7] F. Q. Yuan, “Critical issues of applying machine learning to condition monitoring for failure diagnosis,” in 2016 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), 2016, pp. 1903–1907.

[8] S. Ismaeel, A. Miri, and D. Chourishi, “Using the extreme learning machine elm technique for heart disease diagnosis,” in 2015 IEEE Canada International Humanitarian Technology Conference (IHTC2015)

[9] D. Dahiwade, G. Patle, and E. Meshram, “Designing disease prediction model using machine learning approach,” Proceedings of the 3rd International Conference on computing methodologies and communications, ICCMC 2019, no. ICCMC, pp. 1211-1215,2019.

[10] S. Jadhav, R. Kasar, N. Lade, M. Patil, and S. Kolte, “Disease Prediction by Machine Learning from Healthcare Communities,” International Journal of Scientific Research in Science and Technology, pp. 29–35, 2019.

[11] R. Saravanan and P. Sujatha, “A state of art techniques on machine learning algorithms: A perspective of supervised learning approaches in data classification,” in 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS), 2018, pp. 945– 949.

[12] Y. Amirgaliyev, S. Shamiluulu, and A. Serek, “Abnormality Detection in spinal cord applying machine learning methods,” in 2018 IEEE 12th International Conference on Application of Information and Communication Technologies (AICT), 2018, pp. 1–4.

[13] V. S and D. S, “Data Mining Classification Algorithms for Kidney Disease Prediction,” International Journal on Cybernetics & Informatics, vol. 4, pp. 13-25,2015.

[14] A. Charleonnan, T. Fufaung, T. Niyomwong, W. Chokchueypattanakit, S. Suwannawach, and N. Ninchawee, “Predictive analytics for chronic kidney disease using machine learning techniques,” 2016 Management and Innovation Technology International Conference, MITiCON 2016, pp. MIT80–MIT83, 2017.

[15] P. Kotturu, V. V. Sasank, G. Supriya, C. S. Manoj, and M. V. Maheshwarredy, “Prediction of chronic kidney disease using machine learning techniques,” International Journal of Advanced Science and Technology, vol. 28, no. 16, pp. 1436–1443, 2019