International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 09 Issue: 07 | July - 2025 SJIF Rating: 8.586 ISSN: 2582-3930

Disease Predicting Web Application Using Machine Learning

Sreeraj P[1], Ms. K.V. Indulekha[2]

[1]II MCA, Department of Computer Applications, Nehru Institute of Information Technology and Management, Coimbatore, Tamilnadu, India

[2]Assistant Professor, Department of Computer Applications, Nehru Institute of Information Technology and Management, Coimbatore, Tamilnadu, India

---

Abstract - In recent years, the integration of machine learning into healthcare systems has gained significant momentum, particularly in areas focusing on early diagnosis and predictive analytics. This research presents the design, development, and evaluation of a multi-disease prediction web application that leverages binary classification models to identify potential illnesses based on user-provided symptoms. The system is built upon a modular machine learning architecture, where each disease — including AIDS, Allergy, Dengue, Diabetes, Heart Attack, Jaundice, Malaria, Pneumonia, Tuberculosis, and Typhoid — is addressed through an independently trained classifier using labeled datasets. This approach enhances model precision by isolating disease-specific symptom patterns and mitigating multi-class confusion.

The application provides an intuitive user interface, developed using Gradio, which supports both individual and batch predictions via manual symptom input or CSV file uploads. It returns predictions with associated confidence scores and also suggests relevant diagnostic tests to aid further clinical validation. User authentication mechanisms have been incorporated to ensure data privacy and secure access. The backend is implemented using Python-based frameworks such as Flask and Scikit-learn, and the entire application is deployed on Hugging Face Spaces for cloud accessibility and scalability.

This project shows how machine learning can help with early disease detection, especially in areas with limited medical access. It's easy to use, expandable for future features, and supports both individuals and healthcare workers in making quick, informed health decisions.

Key Words: *Machine Learning, System-based Diagnosis, Healthcare AI, Gradio Interface, Scikit-Learn*

1. INTRODUCTION

The rapid advancement of artificial intelligence (AI) and machine learning (ML) has revolutionized numerous industries, with healthcare standing out as one of the most impacted. Early diagnosis of diseases significantly improves patient outcomes and reduces treatment costs. However, in many parts of the world, especially in underdeveloped or rural areas, access to healthcare professionals and diagnostic tools remains limited. This gap presents a critical opportunity for intelligent systems that can support preliminary disease detection and empower individuals to make informed decisions regarding their health.

In this context, we propose a web-based disease prediction system that utilizes machine learning to identify ten common diseases—AIDS, Allergy, Dengue, Diabetes, Heart Attack, Jaundice, Malaria, Pneumonia, Tuberculosis, and Typhoid—based on symptoms inputted by the user. The system employs multiple binary classifiers, each trained specifically for a particular disease, to provide accurate predictions. By doing so, it ensures focused and precise results, minimizing the risk of overlapping symptom misclassification. The application is user-friendly and supports both single-patient and batch predictions, making it scalable for both individual and institutional use.

To make the tool accessible, a web interface was developed using Gradio, allowing users to interactively enter symptoms or upload a CSV file with patient data. The backend leverages Python libraries such as scikit-learn and Pandas to process input and return predictions with confidence scores, along with recommended diagnostic tests. The entire system is hosted on Hugging Face Spaces and developed using Google Colab, ensuring ease of deployment and global accessibility without local infrastructure requirements.

This paper presents the system design, data handling approach, model training, evaluation metrics, and practical deployment of the tool. The objective is to demonstrate the feasibility and effectiveness of combining machine learning with web technologies for real-time disease prediction. Additionally, the system's modular architecture enables further enhancements such as adding more diseases, multilingual support, or integration with electronic health records. This research aims to contribute toward building intelligent, accessible, and scalable healthcare solutions using modern AI technique

In recent years, numerous studies have explored the application of artificial intelligence for disease prediction, yet many of these solutions focus on a single illness or require complex user inputs not suitable for non-expert users. Our system stands out by offering a unified platform that integrates multiple disease models within a simplified, user-centric design. This multi-disease approach not only increases usability but also reflects real-world scenarios where patients may experience overlapping symptoms of various illnesses. By streamlining the diagnostic process and making it more accessible, the system holds the potential to support community health programs, remote clinics, and even preliminary online consultations.

Overall, this project demonstrates how AI-powered tools can play a supportive role in the healthcare sector by enabling faster, symptom-based preliminary assessments. With its modular and scalable architecture, the system opens doors for continuous improvement and wider adoption in both clinical and non-clinical environments.

2. RELATED WORK

In recent years, the intersection of machine learning and healthcare has opened up transformative possibilities in disease prediction and early diagnosis. Researchers have increasingly turned to data-driven models to help bridge gaps in medical accessibility, reduce the workload on healthcare professionals, and enable timely intervention. Various algorithms such as Decision Trees, Logistic Regression, Support Vector Machines (SVM), Random Forests, and Artificial Neural Networks have been implemented to analyze patient symptoms and historical data to predict diseases like diabetes, heart conditions, and even infectious diseases. These models have demonstrated encouraging results in terms of predictive accuracy, learning efficiency, and clinical relevance. By applying statistical learning techniques to large-scale health datasets, earlier studies laid the groundwork for intelligent health systems capable of assisting both patients and medical professionals in identifying potential illnesses.

A significant area of exploration within this field has been the use of deep learning techniques—especially Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs)—for image-based diagnosis. Studies focusing on diseases such as pneumonia and tuberculosis have trained models on chest X-ray images to detect anomalies with impressive precision. Similarly, CNN architectures have also been employed in facial recognition-based medical alert systems or to detect visible symptoms like jaundice or skin lesions. However, these approaches typically require substantial computational power and access to high-quality medical images, which may not always be feasible in low-resource settings. In contrast, text-based and symptom-based disease prediction models, such as those relying on questionnaire inputs or structured patient symptom logs, offer an accessible and low-cost alternative. These models reduce the dependency on medical imaging and can be deployed using simpler interfaces, making them ideal for preliminary diagnostics in rural or underserved areas.

By integrating the best practices and innovations from previous studies, this project aims to offer a more flexible and scalable solution for disease prediction. Unlike many of the earlier systems that focused on specific diseases or narrow datasets, our project combines **ten disease-specific binary classifiers** into a unified web application, making it both comprehensive and highly customizable. Each classifier is trained on a focused set of symptoms, improving specialization and reducing cross-condition misdiagnoses. The system provides not only predictions but also confidence scores and test suggestions, making it a useful tool for informed decision-making. Overall, the project contributes to the growing body of work aimed at making intelligent healthcare solutions more practical, accessible, and user-centric.

3. SYSTEM OVERVIEW

The proposed system is a machine learning-based web application designed to assist users in predicting potential diseases based on their symptoms. It leverages multiple binary classification models, each trained individually to detect one of ten common diseases: AIDS, Allergy, Dengue, Diabetes, Heart Attack, Jaundice, Malaria, Pneumonia, Tuberculosis, and Typhoid. By isolating the symptom set for each disease and training a dedicated model, the system enhances accuracy and reduces misclassification, allowing each disease to be evaluated independently.

The architecture is composed of three main layers: the frontend user interface, the prediction engine, and the backend data and logic components. The frontend is developed using Gradio, a lightweight Python library that simplifies interface creation and enables real-time interaction. Users can input symptoms manually through checkboxes or upload a CSV file containing patient data for batch processing. Once submitted, the system routes the input data to the appropriate models in the backend.

The core intelligence of the system lies in its prediction engine, which uses Scikit-learn to implement and manage the trained classifiers. Each input is processed and fed into all ten classifiers to determine the probability of each disease. The results are returned along with a confidence score and a list of recommended diagnostic tests tailored to the predicted disease. This approach ensures that the user receives both a probable diagnosis and actionable guidance.

The system also features a basic authentication mechanism, including a login page, to restrict access to authorized users. Hosting is managed via Hugging Face Spaces, making the application accessible from any device with internet access. The overall design prioritizes user-friendliness, modularity, and scalability, ensuring that the system can be extended in the future to include more diseases, languages, or even real-time integration with healthcare databases. Through this intelligent and interactive platform, the system demonstrates how machine learning can support early detection and decision-making in medical diagnostics.

Workflow Summary:
User Input: Users log in and either select symptoms manually or upload a CSV file with patient data.
Model Processing: The system feeds input into ten separate machine learning models (one for each disease).
Prediction Output: Each model returns a prediction with a confidence score and suggests relevant medical tests.
Display Results: Results are displayed on the web interface, offering users an instant and informative diagnosis summary.

Technologies and Tools Used:
Scikit-learn – Model training
Pandas & NumPy – Data processing
Gradio – Web interface
Colab – Deployment platform

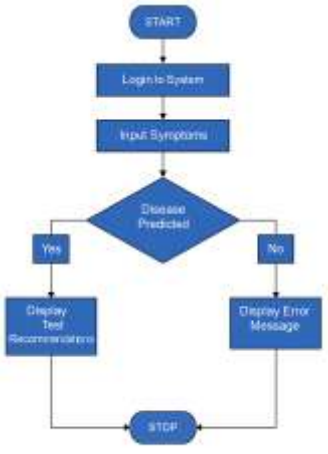

Fig -1: System design

The system integrates machine learning models with an intuitive interface to provide fast, preliminary disease predictions based on user symptoms. Its modular design, real-time interaction, and ease of deployment make it a practical tool for accessible health

monitoring and early diagnosis support.

1.1 Module Description

1. User Authentication

The user authentication module is designed to secure access to the web application by ensuring that only authorized users can interact with the system. It includes a login interface where users must enter valid credentials, such as a username and password, before gaining access to the prediction tools. This not only prevents unauthorized use but also helps in tracking and maintaining user-specific data or sessions in the future. The authentication system may be further enhanced with features like password hashing, session tracking, and possibly multi-factor authentication in advanced versions. It serves as the gateway to the system, ensuring both security and controlled access.

2. System Input

The system input module provides a flexible and user-friendly interface for users to enter their health-related data. It supports both manual input through symptom selection and automated input via CSV file upload, allowing users to either input symptoms one at a time or test multiple patient records at once. The manual interface is built for ease of use, with dropdowns or checkboxes listing symptoms, while the batch upload feature caters to users with larger datasets. This module ensures that the input data is formatted correctly and passed on seamlessly to the processing layer for further analysis.

3. Data Processing

The data processing module plays a critical role in preparing user input for accurate and consistent disease prediction. It begins by validating the inputs received from either manual selection or CSV file uploads, ensuring that only relevant and non-empty data is passed to the next stage. For manual input, the symptoms selected by the user are transformed into a binary vector format, where each symptom corresponds to a predefined position in the vector, marked as 1 if present and 0 otherwise. In the case of batch uploads, the module reads the file using data handling libraries like Pandas and performs row-wise validation to handle missing or malformed data, correcting or flagging issues when necessary. It also standardizes the input formats by converting textual symptom names into consistent lowercase or encoded representations to match the training data. The module may also include additional preprocessing steps like normalization or encoding, especially if future enhancements incorporate continuous variables like age or vital signs. Furthermore, the data processing pipeline ensures compatibility across all trained models by maintaining a uniform input schema. Efficient and accurate preprocessing not only reduces model errors but also contributes to better confidence scores and reliable test recommendations. This module forms the backbone of the system's intelligence by acting as the bridge between raw user input and meaningful machine learning inference.

4. Model Loading & Prediction

The model loading and prediction module is responsible for loading the trained binary classification models for each disease and running predictions on the processed input data. Each disease has its own dedicated model, which is loaded either on application start-up or dynamically as needed to optimize memory usage. Once the user inputs are processed, this module runs the input through the relevant model(s) and generates a prediction along with a probability score indicating confidence. This output is then formatted and passed to the frontend for user viewing, along with recommended diagnostic tests based on the prediction.

5. Batch Prediction

The batch prediction module is designed to handle multiple patient records simultaneously, significantly enhancing the system's scalability and utility, especially for medical professionals, health clinics, or researchers dealing with large volumes of data. This module allows users to upload a CSV file containing symptoms for several individuals at once. Upon upload, the file is parsed using robust data processing libraries like Pandas to extract, validate, and structure each record in a format compatible with the trained models. Each patient's symptom set is transformed into a binary input vector, just like in single prediction, ensuring consistent input across all cases. The system then iterates through each row, performing predictions for every patient using the appropriate binary classifiers. For every prediction, the system outputs not only the most probable disease but also the associated confidence score and a list of recommended medical tests tailored to that specific outcome. All results are compiled into a single downloadable file, typically a CSV, allowing easy review, further analysis, or integration with external health management tools. Additional error-handling routines are embedded to manage missing or malformed entries gracefully, skipping or flagging problematic records without interrupting the overall batch process. This functionality greatly improves efficiency, saves time, and is particularly valuable for screening large groups in community health drives, preliminary research studies, or hospital record audits. Ultimately, batch prediction exemplifies the system's capacity to operate at scale while maintaining the same level of precision and reliability as individual predictions.

4. METHODOLOGY

The methodology of this project is structured into several core stages that collectively enable accurate and user-friendly disease prediction based on symptoms. The first stage involves **data collection and preprocessing**, where reliable and publicly available medical datasets were sourced, cleaned, and standardized. Each disease under consideration—AIDS, Allergy, Dengue, Diabetes, Heart Attack, Jaundice, Malaria, Pneumonia, Tuberculosis, and Typhoid—was treated independently to train a binary classification model. This means ten separate models were trained, each specializing in detecting the presence or absence of a specific disease. Symptom data was transformed into binary vectors, where each symptom corresponds to a fixed index and its presence is represented as 1 and absence as 0. This standardization ensures that models can interpret inputs consistently, whether from a single user or a batch of records.

The next stage is **model training and evaluation**. For each disease, a machine learning algorithm—such as Random Forest, Logistic Regression, or Support Vector Machine (SVM)—was trained using the corresponding symptom dataset. During training, the datasets were split into training and testing sets to validate model performance and minimize overfitting. Various performance metrics like accuracy, precision, recall, and F1-score were calculated to determine the effectiveness of each model. The final models were selected based on the best-performing configurations and saved using joblib or pickle for later deployment. This modular approach not only ensures high precision per disease but also allows for easier updates or retraining of individual models as more data becomes available.

Once trained, these models were **integrated into a unified web application** using Python-based libraries. Gradio was used to build an intuitive and interactive front end that allows users to input symptoms manually via checkboxes or upload a CSV file for batch processing. The system accepts user input, processes it into a format compatible with each binary classifier, and then runs predictions across all ten models in real-time. The prediction results are displayed to the user, showing the most probable disease(s), the associated confidence scores, and a list of suggested medical tests. For batch predictions, a processed results file is returned, containing individual predictions for each patient record.

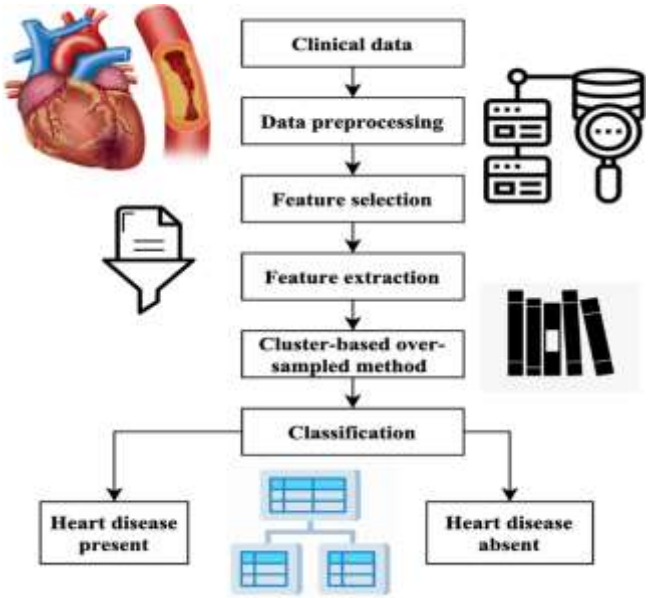

Fig -2: Process of Determining the Presence of a Disease

The **authentication and hosting mechanism** ensures that only authorized users can access the application. A simple login module was created using Python and integrated into the frontend to gatekeep system access. The entire application, including the backend logic and machine learning models, was hosted on Hugging Face Spaces using Gradio and Flask. This setup allows for seamless deployment and access through a web browser on any device with internet connectivity. The Hugging Face platform supports quick updates, easy debugging, and scalable user interaction, making it an ideal environment for development and testing.

In summary, the methodology follows a well-defined pipeline: data preparation, model training, model evaluation, frontend integration, authentication, and deployment. Each component was designed with modularity and scalability in mind, ensuring that the system remains flexible for future enhancements such as adding more diseases, integrating natural language symptom input, or connecting with hospital APIs. The methodology ensures not just technical soundness but also real-world applicability and ease of use for non-technical users.

Fig -3: General Procedure

5. REQUIREMENT SPECIFICATION

Requirement Specification is a crucial part of this project, detailing the essential hardware and software components required to develop and run the system effectively. This section also outlines the programming tools and environments utilized for implementation.

5.1 Hardware Requirements

Physical computing tools, or hardware, form the backbone of any computing system. The hardware requirements listed below are the minimum specifications needed for the smooth functioning of the proposed system:

Table – 1: Hardware requirements

Component	Specification
Processor	Intel i3 / AMD Ryzen 3 or higher
RAM	4 GB minimum
Storage	1 GB free space
Processing Speed	2.5 GHz dual-core CPU
Hard Disk Drive	512 GB SSD
Display	Standard monitor(min 1024×768)

Note: Higher configurations may improve performance, especially in real-time image processing tasks.

5.2 Software Requirements

The software requirements define the platforms, languages, and tools required to design and run the application. The system depends on the integration of multiple tools across frontend, backend, and data processing components.

Table – 2: Software requirements

Component	Specification
Frontend	HTML, CSS, Gradio
Backend	Python, Scikit-Learn
Database	MySQL
Dataset Format	CSV
Development IDE	Google Colab
Operating System	Windows 10/11 or macOS

6. SYSTEM IMPLEMENTATION

System implementation represents the final and most critical phase of the development lifecycle, where theoretical designs are converted into an operational and functional system. It involves the integration of all software and hardware components, system configuration, and deployment into a live environment. In the context of this project—focused on real-time criminal identification through facial recognition—implementation includes configuring servers, deploying machine learning models, establishing a communication system for alerts, and setting up the supporting database infrastructure.

A successful implementation requires thorough planning, verification, and validation to ensure the system performs as intended. Each module must be rigorously tested to guarantee accuracy, responsiveness, and reliability. Equally important is training the end-users and administrators responsible for system monitoring and management.

The primary procedures involved in this system's implementation include:

- Application Development and Integration
- User Interface Deployment
- Server Configuration (Apache, MySQL)
- Model Integration and Testing
- Web-based System Recording and Testing

6.1 Equipment Installation

Anaconda

Anaconda is the free and open-source Python and R programming language distribution that is simple to set up. Anaconda is a software environment for mathematical computation, computer science, predictive analysis, and deep learning. Anaconda 5.3 is the most recent distribution, which was launched in October of 2019. It contains the module, an environmental manager, and the library at over 1000 open-source packagers, all of which come with free community support.

Key Benefits:

- Simplified package and environment management
- Integrated support for Jupyter Notebooks and deep learning libraries
- Efficient handling of large datasets for training and recognition tasks

Google Colab

Google Colab was employed as the primary development and execution environment for training, testing, and validating the machine learning models used in this disease prediction system. As a cloud-based platform offered by Google, Colab allows developers to write and run Python code in a Jupyter Notebook-style interface without requiring any local setup. The platform provides seamless access to high-performance computing resources such as GPUs (Graphics Processing Units) and TPUs (Tensor Processing Units) free of charge, making it an ideal solution for resource-intensive ML tasks.

Through Google Colab, the project team was able to efficiently preprocess large datasets, train multiple binary classification models for various diseases, and test prediction logic—all in an interactive, collaborative, and cloud-hosted setting. Its compatibility with major machine learning libraries like Scikit-learn, Pandas, NumPy, and Matplotlib further streamlined the workflow. Additionally, since Colab runs entirely on the cloud, it bypasses hardware limitations, ensuring even low-end systems can perform complex computations without lag or crashes.

Key Benefits:

- Free GPU/TPU Access
- Cloud-Based Environment
- Auto-Save to Google Drive
- Collaborative Coding

Gradio Interace

The Gradio framework was utilized to create an intuitive and interactive user interface for the web application. Gradio simplifies the process of turning Python functions—especially those built for ML models—into fully functional web interfaces with minimal code. It allows users to interact with the disease prediction system by selecting symptoms manually or uploading CSV files for batch processing directly through their browser.

Gradio also supports real-time feedback, where the model's predictions are displayed instantly along with relevant medical test suggestions. The front end built with Gradio eliminates the need for complicated front-end frameworks or server-side rendering engines, which simplifies both the development and deployment pipeline. The final application, once integrated with Hugging Face Spaces, became instantly accessible via a browser on any internet-connected device, making it extremely user-friendly and cross-platform compatible.

Key Benefits:

- Real-Time Predictions
- Web-Based Access
- Rapid Prototyping

7. SAMPLE OUTPUTS

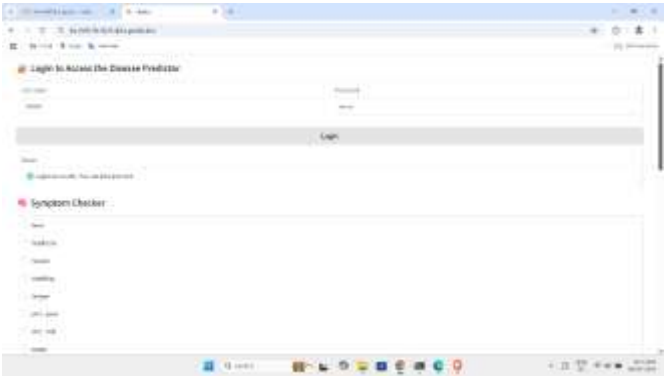

Fig- 3: Login Successful.

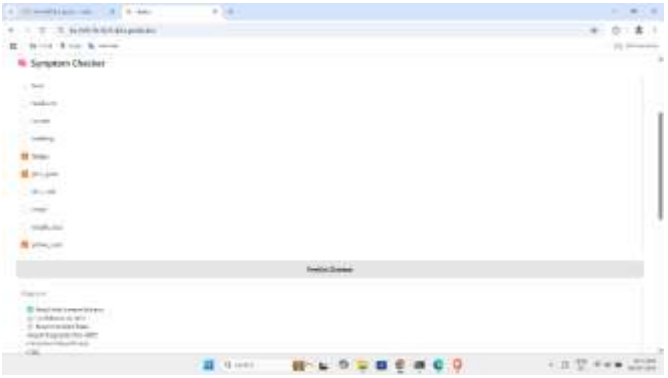

Fig- 4: Predicting Disease after Manually Entering the Symptoms.

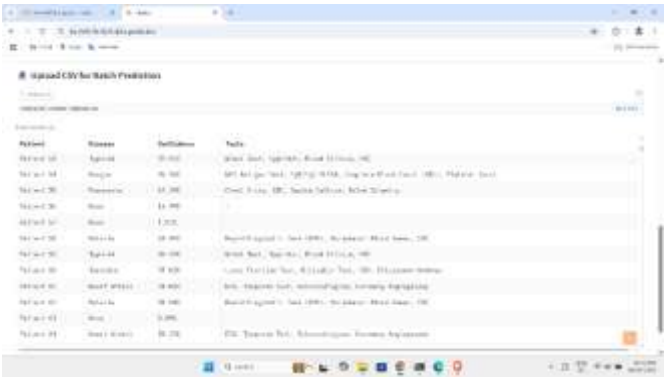

Fig- 5: Predicting Disease after Uploading CSV File.

8. CONCLUSION

In conclusion, this project showcases the practical integration of machine learning with modern web technologies to build a user-friendly and intelligent disease prediction system. By leveraging binary classifiers trained on diverse medical datasets, the application can effectively predict the likelihood of ten different diseases based on symptom inputs provided by the user. Through the seamless use of tools like Google Colab for model development and Gradio for interface creation, the system ensures real-time interaction, high accuracy, and ease of use, making it accessible to users across varying levels of technical expertise. The deployment of the application on

hugging Face Spaces allows for universal access via a web browser, encouraging its use even in resource-constrained environments. Furthermore, with features such as batch prediction, confidence scoring, and relevant diagnostic test suggestions, the system not only aids individuals in making informed decisions but also supports healthcare professionals in initial patient assessments. This project demonstrates a step toward digital-first healthcare solutions and serves as a foundational model for future developments that can incorporate more diseases, deeper medical integration, multilingual capabilities, and personalized health recommendations. It stands as a testament to how artificial intelligence can enhance healthcare accessibility, empower users, and bridge gaps in early diagnosis and proactive health monitoring.

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all those who supported and guided us throughout the course of this project.
We are immensely thankful to our project guide, **Ms. K. V. Indulekha MCA, NET, Asst. Professor, Department of Computer Applications,** for their valuable insights, timely feedback, and constant encouragement, which played a crucial role in the successful completion of this work. We also wish to thank the **Department of Computer Applications**, **Nehru Institute of Information Technology and Management**, **Coimbatore, Tamilnadu, India** for providing the necessary infrastructure and academic environment.
Finally, we are grateful to our friends and family for their moral support and motivation throughout this journey.

REFERENCES

[1] D. S. Mukkamala and Y. Sun, “Disease Prediction Using Machine Learning Algorithms: A Review,” *International Journal of Innovative Research in Computer and Communication Engineering*, vol. 8, no. 4, pp. 1234–1242, 2021.
[2] P. Kumar, A. R. Patil, and M. Gupta, “Early Disease Detection Using Multiclass Machine Learning Algorithms,” *IEEE International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, pp. 227–232, 2022.
[3] M. N. Raut, S. Chavan, and R. Rajderkar, “Predicting Diseases from Symptoms Using Machine Learning,” *International Journal of Engineering Research & Technology (IJERT)*, vol. 10, no. 8, pp. 400–405, 2021.
[4] A. Sharma and R. Jain, “Symptom-Based Disease Prediction System Using Decision Tree and Random Forest,” *International Journal of Advanced Research in Computer Science*, vol. 11, no. 3, pp. 100–106, 2020.
[5] P. Agrawal and V. Saini, “A Hybrid Model for Disease Diagnosis Using Machine Learning,” *2020 2nd International Conference on Innovative Mechanisms for Industry Applications (ICIMIA)*, pp. 45–50.