



AI Based Disease Prediction and Health Consultancy

Vijul¹, Aryan², Parveen³, Ms.Shreya⁴

Department of CSE-AI&DS,

Panipat Institute of Engineering & Technology, Samalkha, India^{1,2,3,4}

Mail ID: shreya.cse-aids@piet.co.in

Abstract—This paper presents an AI-based disease prediction and health consultancy system designed to provide first-level, data-driven healthcare support. The application accepts user-selected symptoms and can optionally analyze uploaded medical reports in PDF format. A Random Forest Classifier forms the core disease-prediction engine, while PyMuPDF and regular-expression-based natural language processing extract structured patient information such as age, gender, habits, symptoms, and chronic-risk indicators. The extracted information is used to personalize recommendations including doctor selection, dietary guidance, lifestyle suggestions, risk information, and downloadable summaries. The system is implemented as a web application using Python, Streamlit, Pandas, and scikit-learn. The project report records approximately 96.2% training accuracy, 93.8% test accuracy, 92.4% average precision, and 91.7% average recall for the evaluated model. A sample symptom case returned a 92% confidence score. The system is intended as a preliminary decision-support and awareness tool rather than a replacement for professional diagnosis. The work also identifies privacy, latency, report-format variability, deployment, and evaluation challenges and proposes future extensions such as multilingual support, chatbot assistance, wearable integration, and scalable cloud deployment.

Keywords—Artificial intelligence, disease prediction, Random Forest, medical report parsing, NLP, Streamlit, health consultancy, personalized healthcare.

INTRODUCTION

Artificial intelligence and machine learning are increasingly being applied to healthcare for early disease detection, decision support, and personalized information delivery, a direction consistent with global health-technology guidance [10]. The project documented in the source report, “AI Based Disease Prediction and Health Consultancy,” addresses the need for an accessible first-level health support system that can connect symptom information with probable conditions and subsequent guidance. The system combines symptom classification, document parsing, user profiling, and a web-based interface in one workflow. The problem addressed is not only disease prediction but also the difficulty faced by users in interpreting medical reports and deciding what to do after observing symptoms. The proposed application therefore extends beyond a classifier: it can extract patient attributes from uploaded reports and use them to tailor recommendations. The project is designed particularly for users who may have limited or delayed access to healthcare facilities, while explicitly positioning its output as preliminary support rather than a clinical diagnosis. The source report identifies several practical gaps: shortage of doctors in underserved areas, healthcare-system overload, limited health awareness, unreliable self-diagnosis through generic web searches, and difficulty understanding medical documentation. These gaps motivate a modular system that can provide rapid symptom-based predictions and understandable next-step information.



SYSTEM OBJECTIVES AND CONTRIBUTION

- Predict probable diseases from a user-selected set of symptoms using a supervised machine-learning classifier.
- Accept medical reports in PDF format and extract structured health information using PyMuPDF and rule-based NLP.
- Personalize consultancy using demographic and lifestyle attributes such as age, smoking, alcohol use, and chronic-risk indicators.
- Provide disease-related doctor, diet, lifestyle, and risk guidance through a simple Streamlit interface.
- Generate a downloadable summary so that the user can retain the prediction and recommendations.

RELATED WORK

The literature reviewed in the source report spans classical machine-learning approaches for disease classification, medical NLP, and vision-language systems. Random Forest has been highlighted as a robust option for structured symptom-disease data because an ensemble of decision trees can capture interactions among categorical features. The report cites prior disease-prediction work using Random Forest and related classifiers, as well as NLP-based extraction of medical information [1]– [3].

The review also discusses transformer-based methods such as BLIP and attention-based approaches [4], [6]. Although those technologies are broader than the final symptom-classification objective, the report uses them as part of its discussion of AI-assisted language and document processing. CheXNet is cited as an example of deep learning for medical imaging [5]. These studies motivate a layered design in which structured machine learning is used for prediction while NLP is used to interpret supporting documents.

The source report's comparative literature table records a Random Forest disease-classification study with 91.5% accuracy, a decision-tree/SVM study for diabetes prediction with 88% accuracy, a Regex+NLP medical-document parsing study with 85% field-extraction success, and BLIP/attention-based work for image or report captioning, as summarized in Table I. These results support the selection of a lightweight ensemble classifier and rule-based document extraction for a practical web application.

TABLE I: SUMMARY OF SELECTED RELATED WORK

Study / Source	Technique	Application	Reported outcome
Kumar et al. (2022)	Decision Trees + SVM	Diabetes prediction	Accuracy: 88%
Li et al. (2022)	BLIP Transformer	Image/report captioning	Fluent sentence generation
Goyal et al. (2022)	Random Forest	Disease classification	Accuracy: 91.5%
Parate & Choudhary (2021)	Regex + NLP	Medical document parsing	85% field extraction
Sasibhooshan (2023)	Attention layers	Image captioning	Improved object-context mapping

PROBLEM STATEMENT

Users frequently encounter a combination of symptoms without knowing whether immediate medical attention is required. Existing online symptom searches can be generic and may not account for lifestyle or personal history. At the same time, medical reports contain terminology, abbreviations, numerical



values, and clinical context that can be difficult for non-specialists to interpret. The project therefore targets an integrated preliminary-support workflow rather than a stand-alone disease classifier.

The main technical problem is to map a variable set of symptoms to a probable disease class while maintaining a simple interface and supporting additional information from heterogeneous PDF reports. The system must also avoid presenting algorithmic output as a confirmed diagnosis. This is especially important because the source report identifies accuracy and safety, privacy, report-format variability, latency, and lack of standardized real-time evaluation as practical challenges.

PROPOSED SOLUTION

The proposed solution consists of five cooperating modules: (1) an input interface for symptom selection and report upload, (2) a Random Forest disease-prediction engine, (3) a medical-report parser based on PyMuPDF and regular expressions, (4) a personalized consultancy module, and (5) a Streamlit presentation and PDF-generation layer. The modular organization permits each component to be tested independently before integration. The disease engine receives encoded symptom features and returns the most probable condition and an associated confidence value. When a PDF report is provided, the parser extracts relevant patient attributes and symptoms. The consultancy layer then combines the prediction with the extracted profile to produce disease-related recommendations, as illustrated in Fig. 1. Doctor suggestions can be filtered by city, while diet and lifestyle information are selected from project datasets and protocol tables. The final output is presented in plain language and can be exported as a PDF summary.

DESIGN REQUIREMENTS

- Functional: symptom input, disease prediction, PDF upload, report parsing, personalized consultancy, doctor/diet recommendations, and PDF report generation.
- Usability: a simple browser-based interface suitable for technical and non-technical users.
- Performance: fast inference and responsive interaction on available CPU/GPU resources.
- Safety: clear indication that predictions are preliminary and professional medical consultation remains necessary.
- Privacy: temporary processing of uploaded information, with future support for stronger encryption and compliance controls.

SYSTEM ARCHITECTURE AND METHODOLOGY

The complete workflow is organized as a multi-stage pipeline. A user begins by selecting symptoms or uploading a medical report. Selected symptoms are transformed into the feature representation expected by the trained classifier.

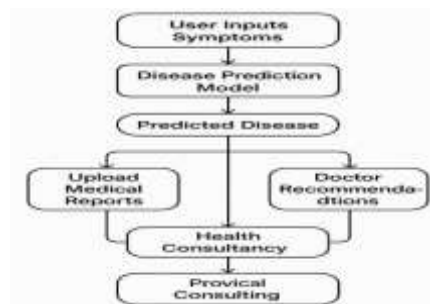


Fig. 1. Overall workflow of the AI based Disease Prediction and Health Consultancy system.



If a report is uploaded, text is extracted with PyMuPDF and important fields are identified with regular expressions and rule-based patterns. The disease prediction and extracted profile are then passed to the consultancy module, which generates contextual recommendations. The result is displayed in Streamlit and may be converted to a downloadable PDF, as shown in Fig. 1.

A. Data Preparation

The project uses a symptom-disease mapping dataset together with supporting doctor and diet datasets. The report states that the disease dataset contains more than 5,000 entries and that users can select approximately 1–10 symptoms. Preprocessing includes organizing the feature matrix and removing the target disease column before model training. Symptom features are represented in a structured form suitable for Random Forest classification.

For report processing, the system extracts text from PDF documents and applies pattern matching to fields such as name, age, gender, smoking, alcohol use, symptoms, illness duration, and chronic conditions. This approach is intentionally lightweight compared with large clinical language models and is intended to make the application practical in a web environment.

B. Disease Prediction Model

Random Forest is used as the primary classifier. The implementation described in the report uses scikit-learn's RandomForestClassifier with 100 estimators [7]. The ensemble combines multiple decision trees and aggregates their outputs, providing robustness for multi-class symptom classification. The source report also considers Decision Tree and Naive Bayes models as baseline or exploratory alternatives.

For an input vector x containing encoded symptoms, the classifier produces a predicted disease class $\hat{y}$ and a probability distribution over classes. The displayed confidence value is derived from the model's class-probability output. The project uses this value as an indicator of prediction certainty rather than as a clinical probability, and Table II summarizes the modular data flow of the system.

TABLE II: SYSTEM MODULES AND DATA FLOW

Module	Input	Processing	Output
Input module	Symptoms / PDF	Validation and collection	Structured request
Prediction engine	Encoded symptoms	Random Forest	Disease + confidence
Report parser	PDF text	PyMuPDF + regex	Patient attributes
Consultancy	Disease + profile	Rules / lookup datasets	Advice and referrals
Presentation	Module outputs	Streamlit + FPDF	Interactive / PDF report

C. Personalized Consultancy

The consultancy component uses the predicted condition together with patient-specific information to select relevant recommendations. The report describes doctor recommendations based on disease and city, disease-linked diet guidance, lifestyle modifications, risk-factor scoring, and common medication-management information accompanied by a professional-consultation disclaimer. This design makes the system more useful than a classifier that only returns a disease label.

D. Development and Deployment

The application is implemented in Python with Streamlit as the web interface [8], Pandas for data handling, scikit-learn for model training and inference [7], PyMuPDF for PDF extraction [9], regular



expressions for field identification, and FPDF for downloadable summaries. The project was developed and tested modularly, with the reported deployment using Google Colab and an ngrok tunnel for public access. The report notes that permanent deployment would be better supported by a managed cloud platform.

IMPLEMENTATION

The implementation follows an iterative sequence of requirement analysis, system design, module development, testing, deployment, and maintenance. The interface is intentionally lightweight so that a user can select symptoms, upload a report, inspect the prediction, and obtain consultancy without requiring specialist technical knowledge, as shown in Fig. 2.

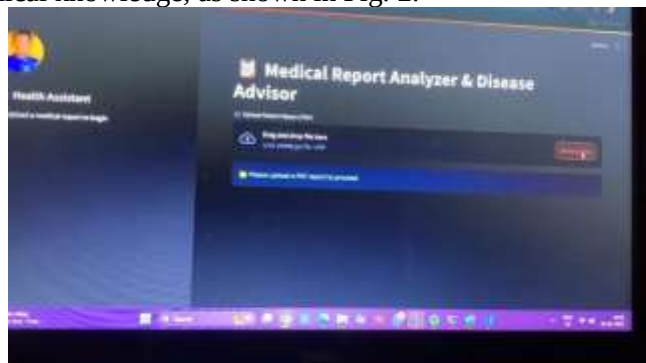


Fig. 2. Streamlit interface used for the health prediction and consultancy workflow.

A. Symptom Input and Classification

The user selects symptoms from the available list. These symptoms are converted into the structured feature vector expected by the Random Forest model. The prediction stage returns the most probable disease, a confidence score, and a risk indication. The application then connects the result to the consultancy dataset. The report gives an example, shown in Fig. 3, in which fever, chills, headache, muscle pain, and fatigue produced a predicted dengue condition with a 92% confidence score and a high-risk indication.

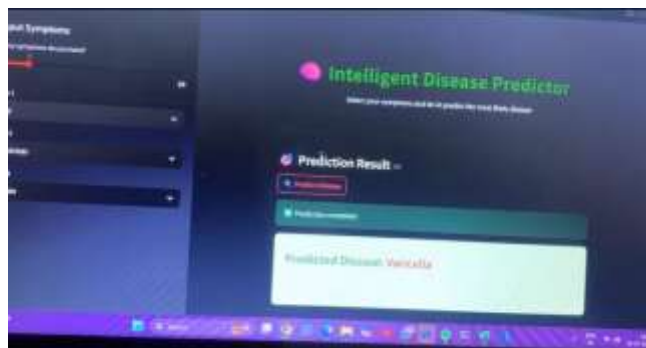


Fig. 3. Example disease-prediction output generated by the application.

B. Medical Report Parsing

The optional report-analysis path accepts PDF documents. PyMuPDF extracts text from the document and regular-expression rules identify relevant fields. The extracted information is passed to the consultancy layer to improve personalization. The source report demonstrates a sample profile containing



age, smoking status, alcohol status, chest pain, and shortness of breath, followed by a possible cardiovascular-risk assessment and city-specific cardiology recommendation, as shown in Fig. 4.

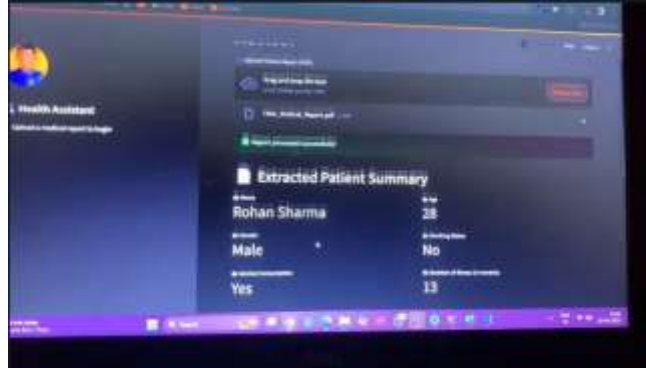


Fig. 4. Example of extracted patient information and health-summary output.

C. User-Centric Output

The interface is designed to return understandable results instead of raw model structures. The final summary can contain the predicted disease, risk level, doctor information, dietary advice, lifestyle suggestions, and other project-defined recommendations. The report emphasizes that medication-related information is advisory and not a prescription. The same workflow can be used with or without an uploaded medical report, which keeps the system accessible for users who only have symptom information.

D. Hardware and Software Environment

The documented minimum environment includes an Intel Core i5-class processor, at least 8 GB RAM, approximately 256 GB SSD storage, and stable Internet connectivity. A GPU is optional but recommended for accelerated inference. The software stack includes Python, Streamlit [8], scikit-learn [7], Pandas/NumPy, PyMuPDF [9], FPDF, regular expressions, and a modern web browser. The source report also identifies Jupyter Notebook, PyCharm, and Visual Studio Code as suitable development environments.

RESULTS AND DISCUSSION

The source report evaluates the disease-prediction component using accuracy, precision, recall, interpretability, and qualitative clinical relevance. The recorded model metrics, summarized in Table III, are approximately 96.2% training accuracy, 93.8% test accuracy, 92.4% average precision, and 91.7% average recall. The report notes that the model performed well across diseases, with minor confusion among conditions that have similar symptom patterns.

TABLE III: REPORTED MODEL PERFORMANCE

Metric	Reported score
Training accuracy	≈96.2%
Test accuracy	≈93.8%
Average precision	92.4%
Average recall	91.7%

The reported test accuracy indicates that the model generalizes reasonably well on the evaluation data used in the project. Precision and recall above 90% further suggest balanced performance across the



evaluated classes. However, these values should not be interpreted as evidence of clinical effectiveness because the source report does not document a prospective clinical validation study, an external hospital dataset, or a standardized clinical trial. The results are therefore best understood as project-level evaluation metrics.

A. Functional Results

The application successfully demonstrates the complete workflow from symptom input or report upload to a final consultancy output. In the sample disease-prediction case, five symptoms resulted in a dengue prediction with a 92% confidence score. In the sample PDF-parsing case, the application extracted the patient's age and lifestyle information and recognized chest pain and shortness of breath before generating a possible cardiovascular-risk output and cardiologist recommendation.

The report also states that PDF automation and interactive deployment were achieved through Streamlit and ngrok. This demonstrates that the proposed architecture can combine machine-learning inference, document parsing, and recommendation logic in a single web interface.

B. Comparison With Initial Goals

Table IV summarizes the achievement of the project against its initial goals.

TABLE IV: ACHIEVEMENT AGAINST PROJECT GOALS

Project goal	Status in source report
AI-based disease prediction	Implemented using Random Forest
Medical-report analysis	Implemented using PyMuPDF + regex
Personalized consultancy	Implemented using profile and lookup data
Doctor/diet recommendations	Implemented
Downloadable PDF summary	Implemented
Scalable permanent deployment	Future enhancement

C. Discussion

The principal strength of the project is integration. Instead of treating disease classification, document interpretation, and health guidance as independent tools, the application connects them in a single user workflow. This is particularly useful for first-level awareness because the user receives a probable condition together with contextual information about what to consider next. The modular design also permits future replacement of the classifier, parser, or recommendation layer without redesigning the entire interface.

At the same time, the evaluation methodology has limitations. The report relies substantially on manual and project-level testing, and it notes the absence of standardized real-time evaluation metrics. Further validation should therefore use externally collected datasets, clinically reviewed labels, calibration analysis, confusion matrices, subgroup analysis, and prospective usability testing.

CHALLENGES AND LIMITATIONS

- Clinical safety: an ML prediction cannot replace professional diagnosis, especially for severe or ambiguous symptoms.
- Report variability: PDF layouts and terminology differ substantially across hospitals and laboratories.



- Privacy: health reports contain sensitive information and require secure transmission, processing, and deletion.
- Latency and scalability: Colab/ngrok is useful for demonstration but is not an ideal production architecture.
- Evaluation: real-user inputs are heterogeneous and require more rigorous clinical and usability metrics.
- Language and accessibility: medical English can be difficult for many users; multilingual and accessible outputs are needed.

FUTURE SCOPE

The source report identifies several directions for extending the system. A conversational chatbot could provide a more natural interaction layer, while multilingual support could reduce language barriers. Wearable-device integration could allow continuous monitoring of selected health indicators. The architecture could also be connected with broader e-health platforms and professional medical portals.

For production deployment, the application should move from temporary Colab/ngrok hosting to a persistent cloud environment. The report suggests scalable platforms such as AWS, Azure, Google Cloud, Render, Hugging Face Spaces, or Streamlit Cloud [8]. Model optimization through ONNX or quantization can reduce latency. Future versions should also introduce stronger encryption, secure storage policies, access controls, audit logging, and appropriate regulatory/privacy compliance, in line with recognized health-AI guidance [10].

From a machine-learning perspective, future work should include external validation, calibrated confidence estimates, explainability of symptom contributions, class-imbalance analysis, and fairness testing across demographic groups. A larger clinically curated dataset and expert review of recommendations would be important before any high-stakes deployment.

CONCLUSION

This paper presented an IEEE-style conversion of the project “AI Based Disease Prediction and Health Consultancy.” The system integrates Random Forest-based disease prediction [1], PDF medical-report parsing, personalized consultancy, and a Streamlit web interface. According to the project report, the evaluated classifier achieved approximately 93.8% test accuracy, 92.4% average precision, and 91.7% average recall, while a sample user interaction produced a 92% confidence score. The application demonstrates how AI can support first-level health awareness by connecting symptoms and patient context with structured recommendations.

The project should nevertheless be treated as a preliminary decision-support and awareness system rather than an autonomous diagnostic service. Clinical validation, privacy protection, robust evaluation, scalable deployment, and expert-reviewed recommendation logic are necessary for real-world healthcare use. With these improvements, the modular architecture provides a practical foundation for future multilingual, conversational, wearable-integrated, and cloud-based health-support applications.

REFERENCES

- [1] A. Goyal and R. Sharma, “Disease Prediction Using Random Forest Classifier,” *International Journal of Advanced Research in Computer Science and Software Engineering*, 2022.
- [2] S. Kumar and R. Gupta, “AI-based Early Detection of Chronic Diseases Using Machine Learning Techniques,” *Journal of Medical Systems*, vol. 45, no. 3, p. 89, 2021.
- [3] M. Choudhary, “Text Analytics and NLP for Medical Diagnosis Automation,” *Journal of Biomedical Informatics*, vol. 113, p. 103642, 2021.



-
- [4] J. Li et al., “BLIP: Bootstrapping Language-Image Pre-training for Unified Vision-Language Understanding,” arXiv preprint arXiv:2201.12086, 2022.
- [5] P. Rajpurkar et al., “CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning,” arXiv preprint arXiv:1711.05225, 2017.
- [6] A. Vaswani et al., “Attention Is All You Need,” in Advances in Neural Information Processing Systems (NIPS), 2017.
- [7] Scikit-learn Developers, “Scikit-learn: Machine Learning in Python,” 2023.
- [8] Streamlit, “Streamlit Documentation,” documentation resource, 2023.
- [9] Fitz (PyMuPDF) Documentation, “Python Bindings for MuPDF,” 2023.
- [10] World Health Organization, “Artificial Intelligence in Health: Issues and Opportunities,” WHO, 2020.