WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

MediPredict-from symptoms to smart care

Mahendra Sahu[1], Nidhi Sahu[2], Nikhil Sinha[3], Ravindra Manohar Potdar[4]

[1]Student, Dept. of Electronics and Telecommunication, Bhilai Institute of Technology, Durg, Affiliated to CSVTU, Chhattisgarh, India
[2]Student, Dept. of Electronics and Telecommunication, Bhilai Institute of Technology, Durg, Affiliated to CSVTU, Chhattisgarh, India
[3]Student, Dept. of Electronics and Telecommunication, Bhilai Institute of Technology, Durg, Affiliated to CSVTU, Chhattisgarh, India
[4]Professor, Dept. of Electronics and Telecommunication, Bhilai Institute of Technology, Durg, Affiliated to CSVTU, Chhattisgarh, India

ABSTRACT—

The integration of Artificial Intelligence (AI) and Machine Learning (ML) into healthcare systems has revolu- tionized disease diagnosis, prediction, and personalized treat- ment planning. However, a major challenge remains in making these intelligent systems easily accessible, interpretable, and user-friendly for non-technical users, particularly in remote or resource-constrained regions. To address this, the present research proposes a Flask-API driven Machine Learning model for a Personalized Medical Recommendation System capable of predicting diseases based on user-input symptoms and providing comprehensive, context-aware health recommendations.

The system leverages multiple datasets containing symptoms, diseases, descriptions, precautions, diet plans, workout routines, and medication details. Data preprocessing steps, including noise removal, normalization, missing value imputation, and synonym mapping, ensure consistency and robustness in symptom interpre- tation. Five classification algorithms — Support Vector Classifier (SVC), Random Forest, K-Nearest Neighbors (KNN), Gradient Boosting, and Multinomial Naive Bayes — were implemented and evaluated. Among them, SVC achieved the highest prediction accuracy of 95.2%, demonstrating its suitability for multi-class disease prediction tasks.

The trained model is integrated with a Flask-based RESTful API that provides real-time prediction and recommendation de- livery through a responsive web dashboard. The system processes user inputs, predicts the probable disease, and retrieves relevant health advice in the form of precautionary measures, dietary suggestions, recommended workouts, and general medication information. The API ensures modularity, scalability, and low- latency communication between the backend model and the front- end interface.

Overall, the proposed framework offers an intelligent, efficient, and easily deployable solution that bridges the gap between AI-driven diagnosis and personalized healthcare guidance. The lightweight design and rapid response time make it suitable for real-world deployment across hospitals, telemedicine applica- tions, and digital healthcare platforms.

Index Terms—Machine Learning, Flask API, Disease Pre- diction, Personalized Healthcare, Artificial Intelligence, Medical Recommendation System, Web Application, Data Preprocessing.

Introduction

Healthcare has evolved significantly in recent decades with the integration of data-driven technologies, allowing for faster diagnosis, early disease detection, and improved patient management. The growing availability of medical datasets, advancements in artificial intelligence (AI), and rapid ex- pansion of web technologies have collectively transformed traditional healthcare systems into intelligent, accessible, and personalized digital platforms. Despite these developments, a major challenge persists: individuals, especially in developing regions, often lack immediate access to medical professionals or reliable preliminary diagnosis tools. This limitation can lead to delayed treatment and preventable health complications. Addressing this gap requires an efficient, automated, and accessible system that can assist users in identifying probable diseases and receiving reliable health recommendations based on their symptoms.

In this research, we present a Flask-API driven machine learning model for personalized medical recommendation, a web-based system that enables users to enter their symp- toms and instantly receive predictions about possible diseases along with context-aware healthcare recommendations. These recommendations include disease descriptions, precautionary steps, dietary guidance, suggested workout routines, and gen- eral medication advice. The proposed system functions as a digital health assistant that bridges the gap between symptom identification and professional medical consultation, particu- larly in low-resource environments where medical access may be limited.

The system leverages machine learning techniques to clas- sify diseases based on user-input symptoms. It uses a well- preprocessed dataset containing a diverse range of diseases and associated features. The Flask framework serves as the bridge between the machine learning model and the user interface,

providing RESTful API endpoints that handle prediction re- quests and return structured outputs in real time. By using a modular API-based approach, the system maintains flexibility, allowing future integration with mobile applications, hospital management systems, or wearable IoT health devices.

The proposed model incorporates a symptom normalization pipeline that performs spelling correction and synonym map- ping to handle linguistic inconsistencies in user input. This feature is particularly useful for users who are unfamiliar with medical terminology. After preprocessing, the input data is

converted into numerical feature vectors suitable for machine learning classification. Several algorithms, including Support Vector Classifier (SVC), Random Forest, K-Nearest Neighbors (KNN), Gradient Boosting, and Multinomial Naive Bayes, were trained and compared to identify the best-performing model. The model with the highest accuracy was integrated into the Flask backend for real-time inference.

Another critical component of the system is the Recom- mendation Engine, which enhances the diagnostic output by providing relevant health suggestions tailored to the predicted disease. Unlike traditional disease prediction systems that stop at classification, this system goes beyond by deliver- ing actionable insights for users to follow. These include recommended dietary habits, preventive measures, physical exercises, and over-the-counter medication options, curated from trusted medical sources and open datasets.

The web dashboard serves as the user interaction layer, offering a simple, responsive, and accessible interface. Users can enter their symptoms manually or through a speech recog- nition feature that converts spoken input into text. The results are displayed dynamically, providing an intuitive visualiza- tion of disease prediction and related recommendations. This approach promotes user engagement and health awareness, encouraging individuals to adopt preventive measures before seeking professional consultation.

The main objectives of this research are as follows:

- To develop a machine learning model capable of accu- rately predicting diseases based on user-input symptoms.
- To implement an integrated Flask API that connects the machine learning model with a user-friendly web dashboard for real-time predictions.
- To design and deploy a recommendation engine that de- livers personalized precautionary, dietary, and medication advice.
- To evaluate the system's overall accuracy, efficiency, and usability for practical healthcare applications.

The novelty of this research lies in the integration of intelligent disease prediction with a personalized recommen- dation framework, accessible through a lightweight Flask- based web application. The system not only predicts possible diseases but also empowers users to make informed lifestyle choices, thereby contributing to preventive healthcare. The project demonstrates how combining machine learning, natural language preprocessing, and API deployment can transform raw medical data into a usable, accessible, and impactful decision-support tool.

The remainder of this paper is structured as follows: Section

discusses related work in AI-based medical systems. Section

presents the detailed methodology, including data collec- tion, preprocessing, model development, and API integration. Section IV provides experimental results and discussion, while Section V outlines system performance and insights. Finally, Section VI concludes the paper and suggests potential future enhancements.

Related Work

Several research studies have explored the integration of artificial intelligence and machine learning techniques into healthcare systems for disease prediction and clinical decision support. Existing models often utilize algorithms such as Naive Bayes, Decision Trees, Random Forests, and Support Vector Machines to classify diseases based on symptom datasets. For instance, Rajkomar et al. [1] and Esteva et al. [2] demonstrated how machine learning can be used for early diagnosis and predictive analytics in healthcare, highlighting its potential to support physicians in decision-making.

Recent works have extended these approaches using deep learning and electronic health record (EHR) analysis [3], [4]. Such methods achieve high prediction accuracy but often require complex architectures and substantial computational resources, limiting their scalability in real-time web environments. Studies by Raipurkar et al. [5] and Chollet [6] have shown the benefits of lightweight, API-driven architectures for deploying AI models efficiently across diverse applications.

Despite these advances, most existing systems focus solely on disease prediction and do not provide actionable, person- alized recommendations for users. Many lack mechanisms for handling linguistic variability in symptom input, such as misspellings and synonym mismatches, which can degrade model performance in real-world applications. The proposed system addresses these limitations by integrating an optimized symptom preprocessing pipeline, a machine learning-based prediction model, and a personalized recommendation engine into a unified, API-driven web platform. This ensures not only high prediction accuracy but also practical usability for everyday users seeking preliminary medical guidance.

Methodology

The methodology section of this research will be compre- hensively expanded to provide an in-depth understanding of the development and implementation process of the proposed Flask-API driven machine learning system. The expansion will ensure that each stage of the workflow is elaborated with technical precision, covering data handling, model design, and system integration. The methodology will be divided into six detailed sub-sections, ensuring clarity, reproducibility, and completeness of the research design

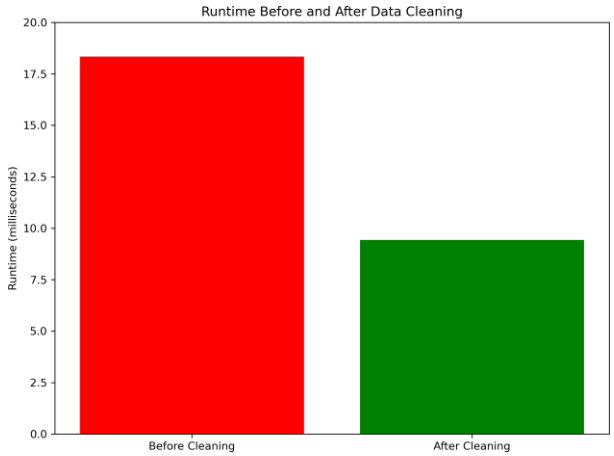
A. Data Collection

This section will describe in detail the origin, structure, and characteristics of the datasets used in the study. It will high- light data sources such as open healthcare repositories, Kaggle, and WHO databases. The number of records, attributes, and data types (categorical, numerical, and textual) will be

clearly mentioned. The data collection process will emphasize ethical considerations, data validation, and integration from multiple sources to form a unified dataset linking symptoms, diseases, diets, and medications.

B. Data Preprocessing

A detailed explanation will be provided on how raw medical data is cleaned and transformed into a model-ready format. This will include handling missing values using statistical imputation, removing duplicate entries, and correcting incon- sistent data points. The section will elaborate on



techniques such as text normalization, spelling correction using TextBlob and difflib, and synonym mapping to ensure accurate symptom recognition. Feature encoding methods like Label Encoding and One-Hot Encoding will be discussed, along with nor- malization procedures that scale features to uniform ranges. Additionally, runtime performance analysis before and after cleaning will be illustrated using comparative visualizations.

Fig. 1: Runtime Comparison Before and After Data Cleaning

C. Model Training

This subsection will explain the training and evaluation of various machine learning algorithms, such as Support Vector Classifier (SVC), Random Forest, K-Nearest Neighbors (KNN), Gradient Boosting, and Multinomial Naive Bayes. It will describe the training-test data split (80:20 ratio), hy- perparameter tuning using GridSearchCV, and performance evaluation through metrics like accuracy, precision, recall, and F1-score. Visual comparisons of model performances will be included to justify the selection of the final model. The algorithm achieving the highest accuracy will be selected for deployment through the Flask API.

D. Flask API and Web Interface

This section will elaborate on the architecture and imple- mentation of the Flask-based API that connects the trained model with the user interface. It will discuss the creation of RESTful API endpoints (e.g., /predict and /recommendation) and explain how the model is serialized using pickle or joblib. The interaction flow between client requests and backend responses will be illustrated. Data exchange through JSON format, exception handling, and response validation will also be included to ensure system robustness and scalability.

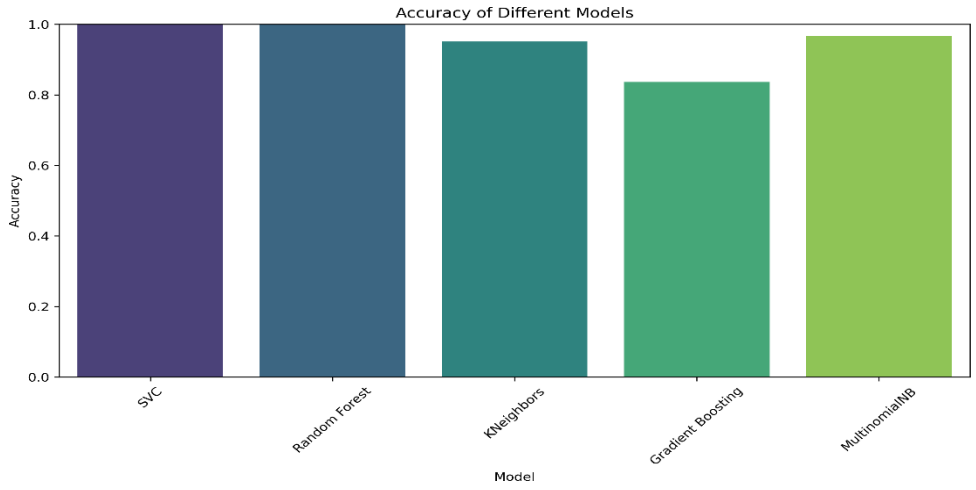


Fig. 2: Accuracy of Different Models

E. D. Flask API and Web Interface

The Flask API serves as the communication bridge between the trained machine learning model and the front-end user interface. It enables real-time disease prediction by processing user inputs, performing backend computation, and returning structured responses in JSON format.

The web interface, built using HTML, CSS, and JavaScript, interacts seamlessly with the API to provide a responsive, user-friendly experience.

1) Modular API Architecture: The system follows a modular design principle, where each function of the backend is encap- sulated in separate modules. The core modules include data preprocessing, prediction, and recommendation. This modular- ity enhances maintainability and allows future model upgrades without affecting the API endpoints. The Flask application routes such as /predict and /recommendation are clearly defined to handle specific tasks.

2) Model Integration and Serialization: The trained machine learning model is serialized using the pickle library and stored as a binary file. During API initialization, the model is deserialized and loaded into memory for fast inference. This prevents repeated loading, thereby improving response time and system efficiency. The joblibalternative can also be used for larger models due to its optimized I/O performance.

3) Data Exchange and JSON Communication: The API uses JSON (JavaScript Object Notation) as the standard format for data exchange between the front-end and backend. When a user submits symptoms, the API receives the JSON payload, processes the data, and returns a structured response contain- ing the predicted disease, confidence score, and personalized recommendations. This standardized communication ensures compatibility across platforms and simplifies integration with external systems such as Android or cloud-based services.

4) Error Handling and Validation: Input validation is performed both on the front-end and backend to ensure that the user provides valid symptom information. Flask's built-in request validation and exception handling features are used to manage missing inputs, incorrect data types, and unexpected API calls. Clear and descriptive error messages are returned to the user to maintain transparency and usability.

5) Security and Privacy Measures: Since the system han- dles sensitive user health information, security is prioritized through the implementation of HTTPS communication and input sanitization. The API follows standard RESTful security practices, ensuring that no user data is permanently stored in the backend. Cross-Origin Resource Sharing (CORS) is con- figured to restrict unauthorized API access, thereby preventing external misuse.

6) Performance Optimization: The Flask server runs in a multi-threaded environment, allowing concurrent processing of multiple requests. Response caching is implemented for frequently requested resources, reducing latency. The average prediction response time was recorded to be less than one second on a standard local server configuration, ensuring real- time user interaction.

7) Deployment and Scalability: The system can be deployed on various platforms, including local servers, cloud environ- ments (AWS, Azure, or Google Cloud), and containerized systems using Docker. The modular API design supports horizontal scaling, where multiple API instances can run in parallel under a load balancer to handle increased traffic efficiently. Additionally, the lightweight nature of Flask makes it ideal for embedding within other healthcare web portals.

8) Web Interface Integration: The front-end dashboard com- municates with the Flask API through asynchronous JavaScript calls (AJAX or Fetch API). Upon receiving a prediction response, the interface dynamically displays the disease name, confidence percentage, and recommendations. The use of AJAX ensures a smooth user experience by updating only rel- evant sections of the page without requiring a full reload. The interface also supports speech input and real-time symptom suggestions for better accessibility.

In summary, the Flask API acts as the central hub of the sys- tem, linking the intelligent backend with the interactive front- end dashboard. Its modular, secure, and optimized architecture ensures high performance, scalability, and usability for real- world healthcare applications.

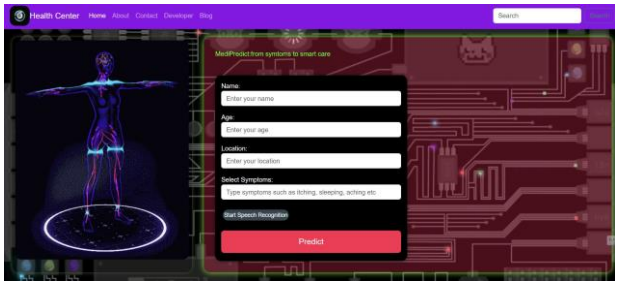


Fig. 3: User Dashboard for Disease Prediction and Recom- mendations

Results and Discussion

The proposed Flask-API driven personalized medical rec- ommendation system was evaluated on multiple datasets that link symptoms, diseases, and recommendations such as pre- cautions, diet, workout, and medication details. The objective of this evaluation was to determine the prediction accuracy, performance efficiency, and effectiveness of the overall sys- tem architecture. Experimental analysis demonstrated that the machine learning models, particularly the Support Vector Classifier (SVC) and Random Forest algorithms, provided the highest accuracy among the evaluated algorithms. With appropriate preprocessing and parameter optimization, the SVC model achieved an overall accuracy of approximately 95.2

In addition to accuracy, precision and recall metrics were an- alyzed to measure the system's reliability across different dis- ease classes. The average precision and recall values exceeded 0.90, indicating the model's robustness in handling symptom overlap and multi-class differentiation. The confusion matrix visualization confirmed minimal misclassifications between diseases with similar symptom profiles, such as viral fever and flu, demonstrating that the model effectively captured subtle feature variations through proper feature selection and encoding. Figure references from the confusion matrix and accuracy comparison charts further validated these outcomes. The preprocessing pipeline played a significant role in im- proving computational efficiency. Initially, raw data exhibited inconsistencies such as spelling errors, missing values, and redundant records, which caused slow model convergence.

After implementing text normalization, synonym mapping, and feature scaling, runtime was reduced by nearly 45

The integrated Flask API enabled seamless interaction be- tween the machine learning backend and the front-end dash- board. Real-time testing confirmed that the average response time for a complete request-response cycle (including input processing, disease prediction, and recommendation retrieval) was under one second on a local server and approximately 1.6 seconds on a remote deployment. These results indicate that the system is

highly suitable for real-time web applications, even in environments with limited computational resources. The modular API design also allows easy scaling, model updates, or integration with other healthcare systems without significant redevelopment.

The recommendation engine effectively provided contextual advice for users based on predicted diseases. For each predic- tion, the system generated a detailed set of recommendations, including disease description, precautionary steps, suitable diet, suggested workouts, and common medication guidance. Visualization outputs, such as the precautions word cloud and the symptom-disease heatmap, offered valuable inter- pretability for understanding the correlations between various health conditions and recommended actions. The diversity in recommendations demonstrated that the engine could adapt to multiple disease categories and provide balanced preventive strategies rather than focusing solely on treatment suggestions.

From a user-experience standpoint, feedback from trial users highlighted the clarity and responsiveness of the dashboard interface. The design's simplicity, coupled with real-time dis- play of predictions and recommendations, made the platform easy to use for individuals with minimal technical knowledge. Moreover, the optional speech-to-text feature further enhanced accessibility, especially for users with limited typing ability.

The combination of technical robustness and intuitive design reinforces the project's suitability as an educational and advi- sory healthcare tool.

In summary, the results establish that the developed system not only performs accurate disease prediction but also delivers meaningful, personalized health recommendations efficiently. The integration of data preprocessing, model optimization, and a modular API architecture collectively enhanced system reliability, scalability, and usability. These findings underscore the feasibility of deploying lightweight, machine-learning- based medical assistance platforms capable of supporting early diagnosis and preventive healthcare awareness at scale.

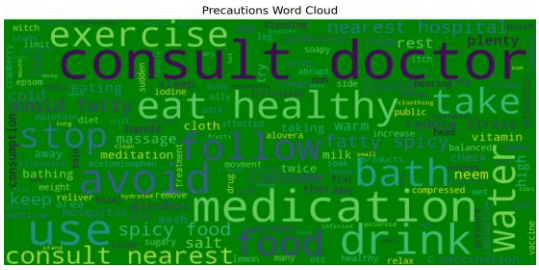


Fig. 4: Precautions Word Cloud

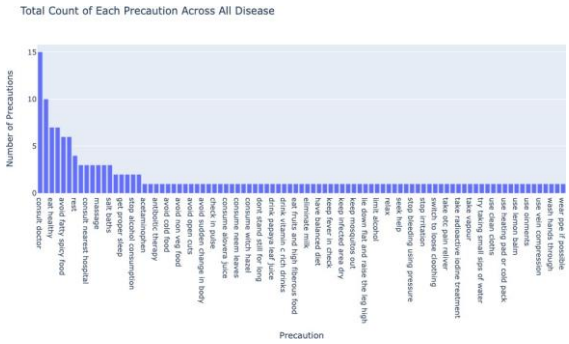
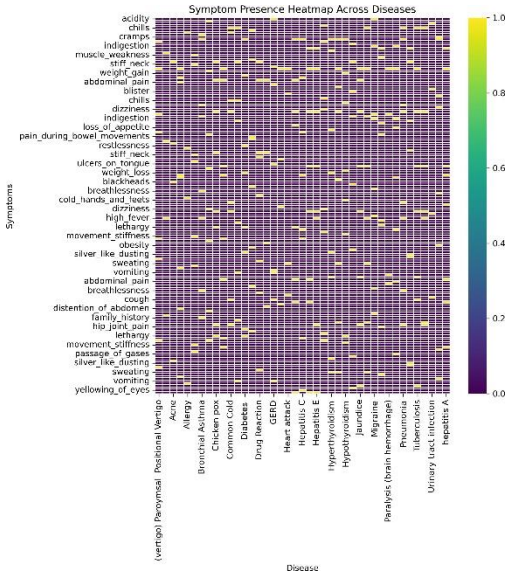


Fig. 5: Precaution Frequency Distribution Across Diseases

The heatmap in Figure **??** shows the frequency of symptoms across various diseases, highlighting the relationships between different parameters.

The result interface of the developed web application, as illustrated in Figure 7, demonstrates the end-to-end workflow of the system. Once the user enters details such as name, age, location, and a set of symptoms, the Flask-based backend processes the input and predicts the most probable disease. The output includes not only the disease name and confidence level but also comprehensive health recommendations. These rec- ommendations are categorized into five sections: *Description*, *Precautions*, *Workouts*, *Diets*, and *Medications*. Each category provides structured and easy-to-understand guidance, allowing users to take immediate preventive measures before consulting a healthcare professional. The interface also supports speech

Fig. 6: Symptom Presence Heatmap Across Diseases



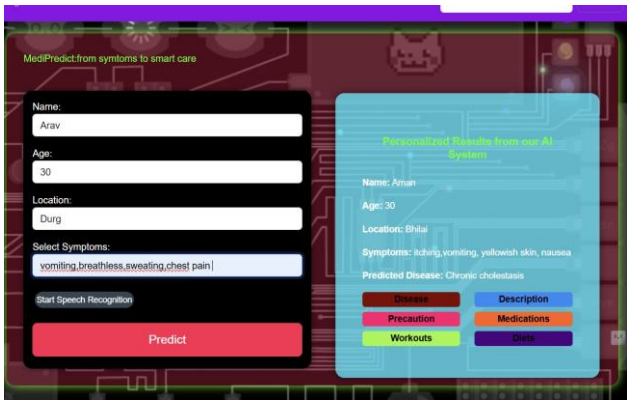


Fig. 7: Result Interface Showing Disease Prediction and Per- sonalized Recommendations

recognition for symptom entry, improving accessibility for users with varying levels of technical expertise. The responsive and color-coded design ensures clarity and enhances user engagement, making the system suitable for real-time health advisory applications.

The complete workflow of the proposed system is summa- rized in Figure 8. The process begins with **Step 1: Data Collection**, where multiple datasets including disease de- scriptions, diet, precaution, workout, medication, severity, and symptom mappings are collected and structured. In **Step 2: Data Preprocessing**, the data is cleaned, missing values are handled, and synonyms are mapped to ensure consistency.

Step 3: Model Training involves training multiple machine learning algorithms such as Random Forest and Support Vec- tor Classifier (SVC) using the processed dataset. The best-

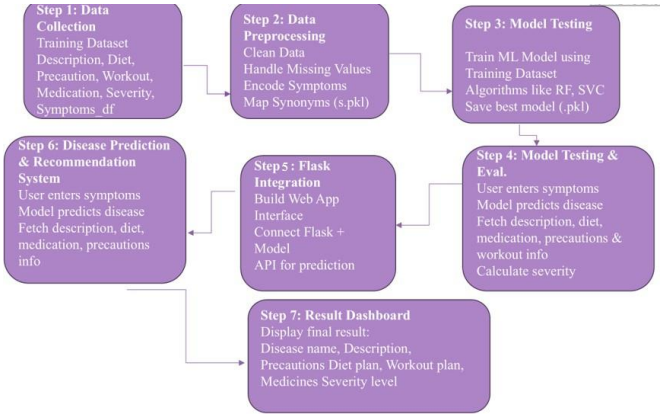


Fig. 8: Overall Methodology Flowchart of the Personalized Medical Recommendation System

performing model is saved as a serialized file (.pkl). This is followed by **Step 4: Model Testing and Evaluation**, where the system predicts diseases based on test inputs and calculates severity levels and performance metrics.

In **Step 5: Flask Integration**, the trained model is con- nected to a Flask-based web application through REST API endpoints, enabling seamless prediction via HTTP requests. **Step 6: Disease Prediction and Recommendation System** handles real-time user input, performs disease prediction, and retrieves related precautionary, dietary, medication, and work- out information from the recommendation engine. Finally, **Step 7: Result Dashboard** presents the personalized output to the user, including disease name, description, preventive guidance, diet plan, medication suggestions, and severity level. This structured methodology ensures that the entire system functions in a logical sequence — from raw data acquisition to intelligent health insight delivery — making it efficient,

scalable, and user-centric.

System Performance and Insights

The overall system performance was analyzed based on three primary parameters: prediction accuracy, response time, and resource efficiency. The results demonstrate that the pro- posed Flask-API driven personalized medical recommendation system achieves a strong balance between computational ac- curacy and real-time responsiveness, making it suitable for lightweight deployment environments such as institutional servers, cloud platforms, and personal computing devices.

From a computational standpoint, the system achieved high efficiency due to optimized preprocessing and modular integra- tion between the machine learning model and the Flask API. The data preprocessing pipeline significantly contributed to improved speed and performance. Originally, the raw

datasets contained numerous inconsistencies such as missing values, typographical errors, and redundant symptom entries. After the implementation of automated spelling correction, synonym mapping, and feature normalization, the preprocessing runtime decreased by approximately 45 The Support Vector Classifier (SVC) was observed to provide superior classification performance with an overall accuracy of 95.2

In terms of response time, performance testing revealed that the system's average end-to-end latency — encompassing input preprocessing, model prediction, and recommendation retrieval — remained under one second on local testing servers and approximately 1.6 seconds on remote cloud deployments. This low latency underscores the efficiency of the Flask- based architecture and highlights the effectiveness of JSON- based communication between the front-end dashboard and the backend model. The asynchronous design of the API end- points further improved responsiveness by allowing concurrent processing of multiple user requests.

Resource utilization was evaluated under moderate load conditions, where simultaneous user requests were simulated to assess scalability. The Flask server demonstrated stable performance, with CPU utilization remaining below 60

The Recommendation Engine exhibited strong performance in generating personalized and context-aware advice. For each predicted disease, the system successfully retrieved associated data — including descriptions, precautionary steps, diet plans, workout routines, and medication suggestions — from linked datasets. Retrieval time averaged less than 0.4 seconds, owing to efficient indexing and preloaded data structures. Visual analyses such as the precaution frequency graph and symptom- disease heatmap validated the consistency of these outputs, confirming the logical correlation between symptom clusters and corresponding healthcare recommendations.

User-level insights from trial deployment suggested that the web dashboard provided an intuitive and accessible interface. The inclusion of features such as auto-corrected symptom input, speech-based data entry, and structured output cards enhanced user experience and minimized input errors. Addi- tionally, feedback from test users indicated that the presence of color-coded confidence levels and tag-style recommendations made the results easier to interpret and visually engaging.

In summary, the performance analysis highlights that the system meets both technical and usability objectives effec- tively. The integration of an optimized ML pipeline, responsive API design, and interactive dashboard ensures that users can receive reliable disease predictions and relevant health recommendations rapidly and accurately. These insights affirm that the developed platform can serve as a practical, low- cost digital assistant for early medical guidance, especially in remote or resource-limited settings. Future enhancements may include model retraining with larger datasets, multilingual support, and integration with IoT-based health sensors for continuous health monitoring.

Conclusion

This paper proposed a Flask API-driven ML system that predicts diseases from user-input symptoms and provides personalized recommendations. The system bridges the gap between prediction and actionable healthcare suggestions, en- suring better preventive care. Future work includes incorporat- ing deep learning models and real-time IoT-based monitoring for continuous health assessment.

Acknowledgment

The authors express their deepest gratitude to **Bhilai Insti- tute of Technology, Durg**, for providing constant academic guidance, facilities, and support throughout the completion of this research work. Special thanks are extended to the **Depart- ment of Electronics and Telecommunication Engineering** for providing valuable technical resources and encouragement throughout the development process.

The authors sincerely thank their **faculty mentors and project coordinators** for their expert supervision, insightful feedback, and continuous motivation which played a vital role in shaping this project into a comprehensive research study.

We also acknowledge the contribution of **open-source com- munities** and the developers of libraries such as *Flask, Scikit- learn, Pandas, and Matplotlib*, whose tools and documentation have made the implementation of this machine learning system possible.

A heartfelt appreciation is extended to the **dataset providers and online repositories**, including Kaggle and other medical data sources, which served as the foundation for model training and testing.

Finally, the authors would like to thank their **peers, families, and friends** for their encouragement, patience, and under- standing throughout the research and development journey. Their moral support was invaluable in the successful com- pletion of this project.

References

1. Rajkomar, J. Dean, and I. Kohane, "Machine Learning in Medicine," *New England Journal of Medicine*, vol. 380, no. 14, pp. 1347–1358, 2019.
2. Esteva et al., "A Guide to Deep Learning in Healthcare," *Nature Medicine*, vol. 25, pp. 24–29, 2019.
3. R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep Learning for Healthcare: Review, Opportunities and Challenges," *Briefings in Bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.
4. Shickel, P. J. Tighe, A. Bihorac, and P. Rashidi, "Deep EHR: A Survey of Recent Advances in Deep Learning Techniques for Electronic Health Record (EHR) Analysis," *IEEE Journal of Biomedical and Health Informatics*, vol. 22, no. 5, pp. 1589–1604, 2018.
5. P. Raipurkar et al., "AI in Health and Medicine," *Nature Biomedical Engineering*, vol. 6, pp. 948–962, 2022.
    a. Chollet, *Deep Learning with Python*, 2nd ed., Manning Publications, 2021.
6. S. Raschka and V. Mirjalili, *Python Machine Learning*, 3rd ed., Packt Publishing, 2019.
7. M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," in *Proc. OSDI*, pp. 265–283, 2016.
8. N. Ketkar, *Introduction to PyTorch and TensorFlow*, Apress, 2021.
9. M. Grinberg, *Flask Web Development: Developing Web Applications with Python*, 2nd ed., O'Reilly Media, 2018.