



Heart Attack Risk Prediction Using Machine Learning Algorithms

D. Srinivasulu, Assistant Professor & HOD, KLM college of engineering for women, Kadapa, India.

P. Bhaskar, Assistant Professor, KLM college of engineering for women, Kadapa, India.

S. Harini Yadav, Assistant Professor, KLM college of engineering for women, Kadapa, India.

Abstract: Heart attacks remain a leading cause of death worldwide, and early detection is crucial for saving lives. Traditional diagnostic approaches often rely on doctors' subjective evaluations, physical tests, and symptom recognition—many of which can be delayed or missed. This paper proposes a machine learning-based solution that can predict the risk of a heart attack using patient health data. To approach the structured data inputs such as age, blood pressure, cholesterol, smoking habits, exercise levels, and family history to assess the likelihood of heart-related events.

The system is built on a predictive model trained using the XGBoost algorithm, known for its high accuracy in classification tasks. To handle class imbalance in the dataset, SMOTE (Synthetic Minority Over-sampling Technique) is used to generate synthetic data points, ensuring fair learning. Additionally, Optuna, an advanced hyperparameter tuning framework, is integrated to automatically optimize the model's performance. Together, these tools enable a highly accurate and robust prediction to support clinical decision-making.

Keywords: XGBoost, SMOTE, data, Optuna

1. Introduction:

Cardiovascular diseases, especially heart attacks, pose a significant health threat and account for a large percentage of mortality worldwide. In most cases, early symptoms go unnoticed or are underestimated, leading to delayed medical intervention. Traditional heart attack prediction methods depend on manual evaluations by healthcare professionals, physical examinations, and diagnostic tests such as ECG, blood pressure analysis, and cholesterol level assessments. While effective, these processes are often time-consuming, inconsistent due to human judgment, and inaccessible for continuous monitoring. With the growing availability of health data and technological advancements, there is a strong need to adopt intelligent, automated methods for predicting heart attack risk.

Machine learning (ML), a powerful subset of artificial intelligence (AI), has demonstrated exceptional capability in analyzing complex medical data and recognizing hidden patterns that might be overlooked in conventional approaches. By learning from historical patient data, ML algorithms can predict health outcomes, identify at-risk individuals, and provide reliable insights for timely preventive actions. The goal of this project is to design and develop a Heart Attack Risk Prediction System using machine learning algorithms, particularly the XGBoost (Extreme Gradient Boosting) classifier. The model is trained on a dataset containing critical features such as age, heart rate, diabetes status, smoking and alcohol habits, exercise levels, diet type, and family medical history.

To improve prediction quality, the dataset undergoes preprocessing and enhancement techniques. The SMOTE (Synthetic Minority Over-sampling Technique) is used to address class imbalance by generating synthetic samples for the underrepresented class, which typically represents high-risk patients. Additionally, Optuna, a state-of-the-art hyperparameter optimization framework, is employed to fine-tune the model parameters for better performance. These enhancements ensure that the system not only provides accurate results but also generalizes well across diverse patient profiles. Data standardization, feature selection, and rigorous validation processes further support the reliability of the model.

The final system is implemented as a web-based application using Flask, making it easily accessible and user-friendly. Patients or healthcare practitioners can input relevant health parameters into the system, which then instantly returns a prediction—either "Low Risk" or "High Risk"—based on the trained model. This system serves as a valuable tool for early diagnosis, lifestyle adjustment, and preventive treatment planning. It demonstrates how machine learning can bridge the gap between data and diagnosis, significantly improving the speed, scalability, and accuracy of heart attack prediction in modern healthcare settings.

Literature Survey

This literature survey highlights the existing research, tools, and frameworks that have influenced and inspired the development of .

1. **Traditional Document Retrieval Systems:** Conventional document retrieval techniques are largely dependent on keyword-based search mechanisms. Tools like Apache Lucene and Elasticsearch have long been used for full-text search capabilities, which allow users to retrieve documents based on the occurrence of specific terms.
2. **Introduction of Semantic Search:** To address the limitations of keyword-based systems, researchers have explored semantic search, which attempts to understand the meaning behind user queries. Technologies such as TF-IDF (Term Frequency-Inverse Document Frequency), Latent Semantic Analysis (LSA), and Latent Dirichlet Allocation (LDA) were early efforts.
3. **Transformer Models and BERT (Bidirectional Encoder Representations from Transformers):** Which can understand natural language process. BERT uses attention mechanisms to consider the fucontext of a word by looking at the words before and after it, making it exceptionally good at tasks like question answering and semantic similarity.

2. System Analysis

Proposed System

uses machine learning to predict heart attack risk based on patient data like age, heart rate, diabetes, smoking, and family history. It enables early and accurate detection through real-time predictions. The model uses the XGBoost algorithm for high accuracy and handles class imbalance with SMOTE to better predict high-risk cases. Optuna is used to automatically tune model parameters, improving performance. The data is also scaled and features are selected for better accuracy. The system has a web interface built with Flask, where users enter health details and instantly receive a “Low Risk” or “High Risk” result. It is fast, reliable, and helps users take timely health action.

The heart attack risk prediction system consists of the following key modules:

DATASET MODULE:

This module handles data collection and preprocessing. It uses health parameters such as age, blood pressure, cholesterol, etc. Missing values are handled, categorical data is encoded, and feature selection is applied to improve prediction accuracy.

USER INPUT MODULE

Users enter their health data via a web interface. Inputs include heart rate, age, diabetes status, smoking habits, alcohol usage, exercise time, and diet type. This data is then passed to the model for prediction.

TRAINING MODULE

This module trains the XGBoost classifier using the preprocessed dataset. It applies SMOTE for class balancing and uses Optuna for automatic hyperparameter tuning to optimize the model's performance.

PREDICTION MODULE

The trained model predicts heart attack risk based on the user's input. It returns either a “Low Risk” or “High Risk” result, providing immediate feedback via the web interface.

Feasibility Study

Feasibility study is an essential part of the system analysis process. It helps in evaluating whether the proposed system is practically achievable and worth implementing in terms of cost, technology, and user acceptance. Its focus on privacy, personalization, and real-time predictions ensures strong acceptance among users and aligns well with modern healthcare goals.

Technical Feasibility

Technically, the system is highly feasible like Python, Flask, and XGBoost, along with libraries such as SMOTE and Optuna for advanced data handling and model optimization

Social Feasibility

From a social perspective, the system promotes proactive health monitoring and awareness, increasing accessibility to predictive healthcare tools via a user-friendly web interface.

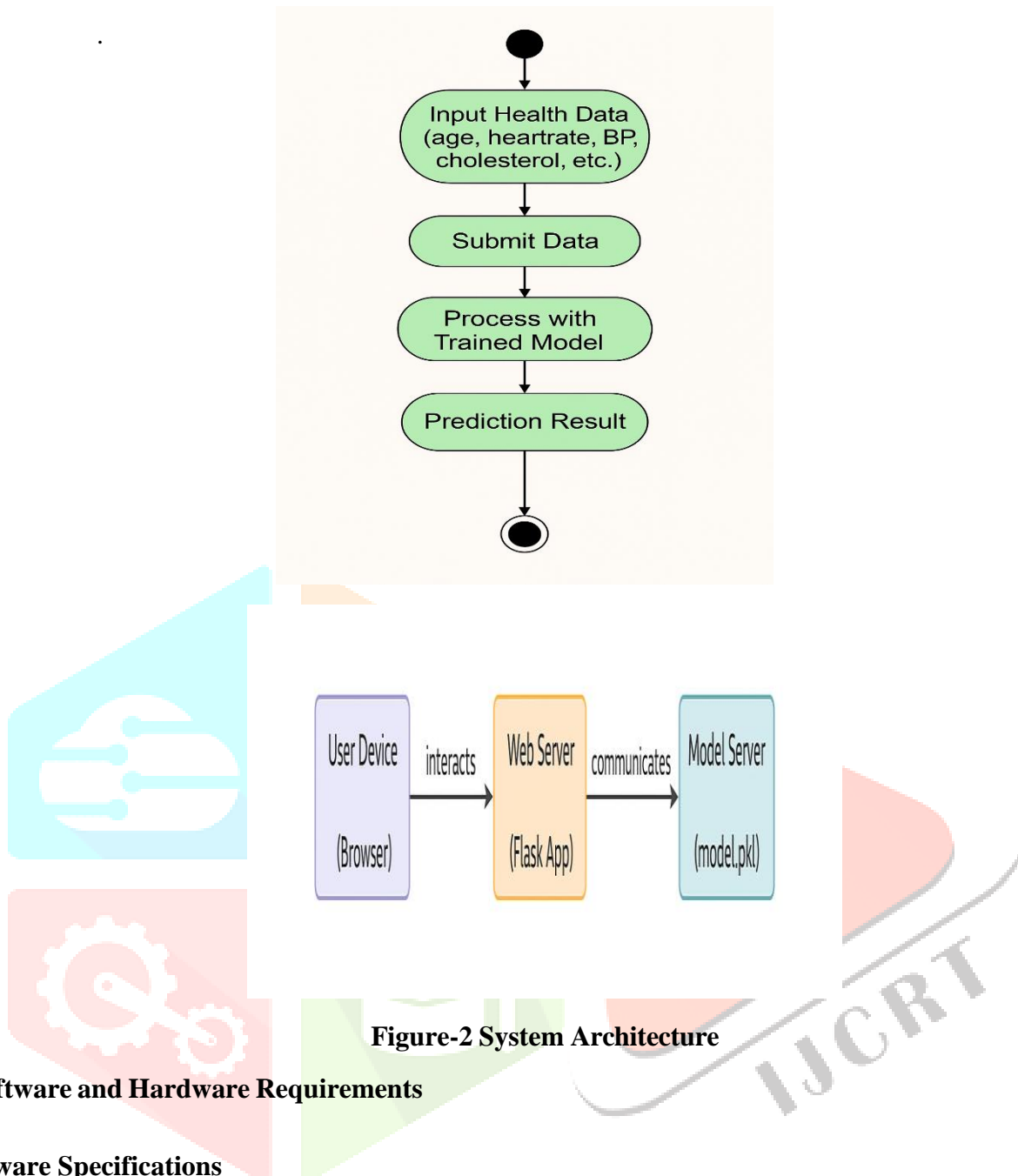


Figure-2 System Architecture

3. Software and Hardware Requirements

Software Specifications

system is built using a robust set of open-source tools and frameworks designed for natural language processing, web development, and efficient retrieval. The choice of software ensures scalability, platform independence, and offline operability.

Operating System: Windows 10 or higher

Programming Language: Python 3.8 or above Python is used for backend development due to its simplicity and rich ecosystem of AI/ML libraries.

Libraries and Frameworks

- Flask – Lightweight Python web framework for building the web interface
- PyTorch – Deep learning framework used for model inference (Flan-T5)
- Transformers (by Hugging Face) – Provides access to pre-trained language models like FlanT5
- Sentence-Transformers – Used to generate vector embeddings (semantic representation of sentences)
- FAISS – Facebook AI Similarity Search for fast and scalable vector search
- PyPDF2 – Library for extracting text from PDF documents
- docx – Library to read text from DOCX files (optional extension)

- NumPy – For efficient numerical operations and matrix handling
- Regex – For basic text cleaning and preprocessing
- HTML/CSS with Bootstrap 5 – For creating a responsive and user-friendly frontend

Development Tools

- Visual Studio Code / PyCharm (for coding and debugging)
- Postman or browser tools (for API testing)
- Git for version control Execution Environment
- Localhost (running completely offline)
- No cloud or internet dependencies required for basic usage.

Hardware Specifications

To ensure efficient execution of the Machine learning algorithms and support for processing large documents and AI model inference, the following hardware configuration is recommended:

3. System Design

Input Design: The input design in Machine learning algorithms defines how users interact with the system and how data is collected for further processing. The primary inputs to the system include document files and user queries.

Document Upload

- **Format Supported:** PDF, TXT
- **Purpose:** To allow users to upload documents from which text will be extracted and processed.
- **Input Method:** File selection via web interface (upload form)
- **Validation:** The system checks file format and size before accepting uploads.

User Query

- **Format:** Natural language question (typed in a textbox)
- **Purpose:** To retrieve specific information or summaries from the uploaded document.
- **Input Method:** Text field on the interface labelled "Ask a Question"
- **Validation:** Empty queries are rejected with a warning; basic sanitization is applied.

UML Diagrams

Unified Modelling Language (UML) diagrams are essential in system design as they visually represent the structure, behaviour, and interactions between different components of the system. In the XGBoot,Smote algorithms UML diagrams help to understand how data flows through the modules, how different classes interact, and how users interact with the system. They also aid developers in visualizing both high-level workflows and low-level operations, ensuring modular development and clear communication between team members.

Class Diagram highlights the structure of the key modules: data set module, user input module, training module, prediction module. Each class has a specific role—e.g., handling text extraction, indexing, or query processing.

Sequence Diagram and Activity Diagram: Which illustrate the step-by-step interaction between objects and user actions. The sequence begins with the user uploading a file, continues through the document processing pipeline, and ends with answer generation.

Deployment Diagram shows how the components are distributed across hardware environments, particularly demonstrating that everything runs locally without internet dependency. These UML diagrams together provide a complete technical view of the system's architecture and interactions.

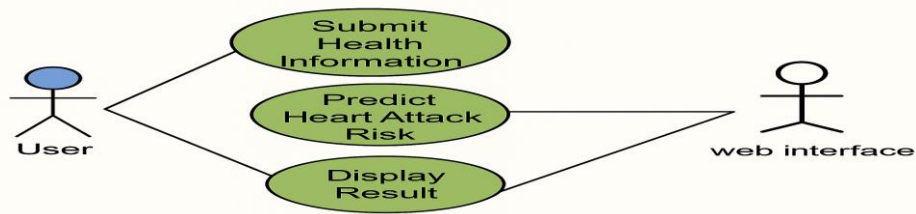


Figure-3 Use Case Diagram for System & User

4. Module Implementation

The system is divided into multiple modules, including:

- Dataset module This module handles data collection and preprocessing.
- User interface module: Users enter their health data via a web interface.,
- Training Module: This module trains the XGBoost classifier using the preprocessed dataset. It applies SMOTE for class balancing and uses Optuna for automatic hyperparameter tuning to optimize the model's performance.
- Prediction Module: It returns either a “Low Risk” or “High Risk” result, providing immediate feedback via the web interface.

All these modules are orchestrated using a Flask backend, ensuring smooth communication between the frontend and processing layers. The system also ensures data integrity and privacy by keeping all processes local, making it suitable for educational, legal, and healthcare environments where sensitive documents are handled.

5. Unit Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product.

1. Unit Testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application.

2. Integration Testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields.

3. Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

4. System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. Ex: white box testing, Black box testing.

6. Output Screens



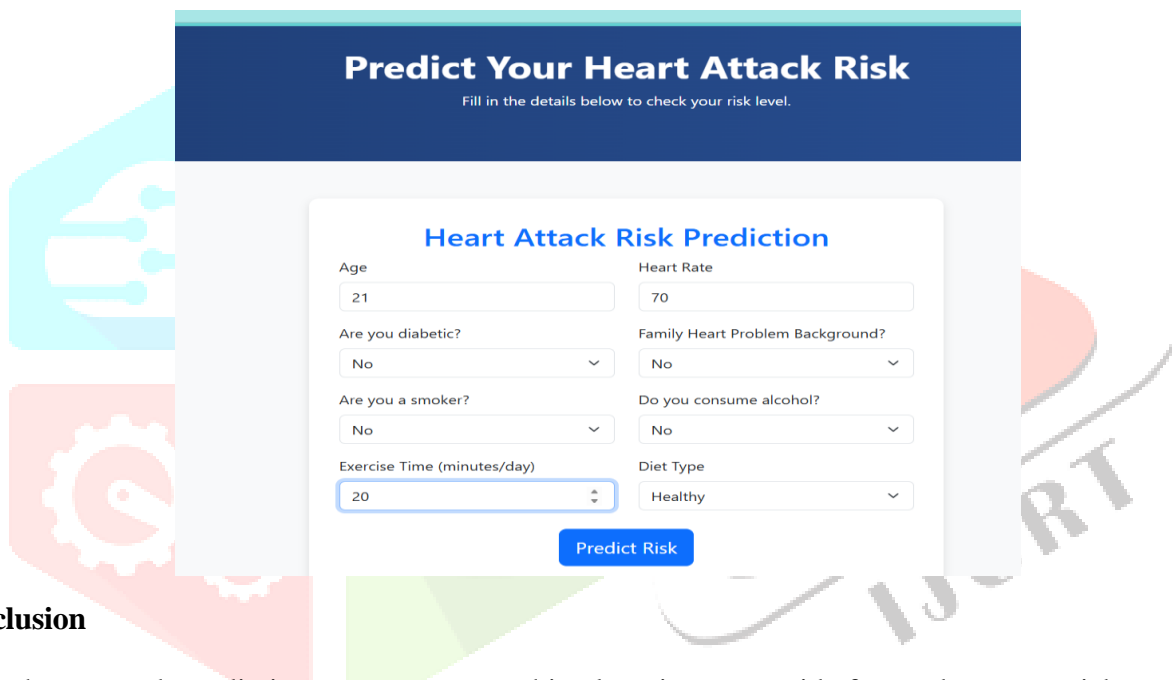
Predict Your Heart Attack Risk
Fill in the details below to check your risk level.

Heart Attack Risk Prediction

| | |
|--|---|
| Age <input type="text" value="21"/> | Heart Rate <input type="text" value="70"/> |
| Are you diabetic? <input type="text" value="No"/> | Family Heart Problem Background? <input type="text" value="No"/> |
| Are you a smoker? <input type="text" value="No"/> | Do you consume alcohol? <input type="text" value="No"/> |
| Exercise Time (minutes/day) <input type="text" value="20"/> | Diet Type <input type="text" value="Healthy"/> |

Predict Risk

Figure-4 User input



Predict Your Heart Attack Risk
Fill in the details below to check your risk level.

Heart Attack Risk Prediction

| | |
|--|---|
| Age <input type="text" value="21"/> | Heart Rate <input type="text" value="70"/> |
| Are you diabetic? <input type="text" value="No"/> | Family Heart Problem Background? <input type="text" value="No"/> |
| Are you a smoker? <input type="text" value="No"/> | Do you consume alcohol? <input type="text" value="No"/> |
| Exercise Time (minutes/day) <input type="text" value="20"/> | Diet Type <input type="text" value="Healthy"/> |

Predict Risk

7. Conclusion

The heart attack prediction system uses machine learning to provide fast and accurate risk assessments. It collects patient data and uses an XGBoost model, enhanced by SMOTE and Optuna, for reliable predictions.

The Flask-based web interface makes it easy to use, supporting early diagnosis and preventive care. This project shows how AI can make healthcare more efficient, data-driven, and personalized. The system's modular architecture, which includes components for text extraction, embedding generation, retrieval, and answer generation, ensures that it is scalable, maintainable, and flexible for future upgrades. Through a simple yet effective web interface built with Flask, users can easily upload documents and interact with the system without needing technical expertise.

8. Problem Statement

The modern retail industry is characterized by increasing complexity, demanding consumers, and fierce competition. Retailers grapple with the challenge of managing intricate supply chains that span numerous suppliers, distribution channels, and logistical processes, all while striving to meet fluctuating customer demand and optimize sales performance. Often, valuable data pertaining to supply chain operations and sales activities resides in disparate systems and formats, including spreadsheets like Microsoft Excel. This fragmented data landscape makes it exceedingly difficult for retail businesses to gain a holistic and real-time understanding of their overall performance.

This is to bridge the gap between fragmented data and actionable insights by providing a centralized and dynamic platform for analyzing key performance indicators related to both the supply chain and sales functions. By visualizing these interconnected datasets, the report will enable retailers to gain a deeper understanding of the relationships between operational efficiency and revenue generation, leading to more informed strategic and tactical decisions.

9. Scope

The feature scope of this is Using real-time data from smartwatches and fitness bands can help monitor health continuously and improve prediction accuracy. A larger and more diverse dataset will make the model more reliable for different types of people. This helps in detecting early signs of heart risks and taking timely action. Future systems could also suggest personalized lifestyle changes based on individual health data.

10. References:

1. Paul Viola and Michael Jones, "Rapid Object Detection using a Boosted Cascade of Simple Features,"
2. Buolamwini, Joy, and Timnit Gebru. "Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification,"
3. Chen, Tianqi, and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System,"
4. Chawla, Nitesh V., et al. "SMOTE: Synthetic Minority Over-sampling Technique.
5. Bird, S., Klein, E., & Loper, E. (n.d.). Natural Language Processing with Python.
6. Russell, S., & Norvig, P. (n.d.). Artificial Intelligence: A Modern Approach.
7. GeeksforGeeks. (n.d.). Python Programming Tutorials. <https://www.geeksforgeeks.org>
8. LangChain Documentation. (n.d.). <https://python.langchain.com>
9. OpenAI API Documentation. (n.d.). <https://platform.openai.com>
10. Facebook AI Similarity Search (FAISS). (n.d.). <https://faiss.ai>
11. GeeksforGeeks. (n.d.). Python Programming Tutorials. <https://www.geeksforgeeks.org>
12. Medium. (n.d.). Articles on Retrieval-Augmented Generation (RAG). <https://medium.com>.