## ABSTRACT

Heart disease is a leading cause of death worldwide, making early detection crucial for reducing

mortality rates. This project focuses on developing a machine learning-based heart disease prediction system that analyses various health metrics to predict an individual's risk of heart disease. Using a dataset of patient data, the system employs algorithms like Logistic Regression, Random Forest, and Support Vector Machines (SVM) to train a model capable of assessing the likelihood of heart disease. The system features a user-friendly web interface, allowing users to input their health data—such as age, cholesterol levels, and blood pressure— through a simple form. The processed data is then analysed by the model to provide a real-time risk assessment. This project can assist healthcare providers in early detection, thus enabling timely medical intervention and potentially improving patient outcomes. Heart disease remains a significant global health challenge, contributing to high mortality rates despite advances in medical science. This project aims to harness the power of machine learning to create a heart disease prediction system that assists in early diagnosis. Leveraging clinical data such as age, cholesterol levels, resting blood pressure, and other cardiovascular indicators, the model provides an accurate prediction of an individual’s likelihood of developing heart disease. The system integrates advanced machine learning techniques, including Logistic Regression, Random Forest, to build a reliable classification model. The user interface is designed to be intuitive, requiring basic health inputs through a web form developed with Flask. Once submitted, the system processes the data, which undergoes necessary preprocessing steps such as normalization and encoding, before being analyzed by the prediction model. The result is a clear and actionable assessment, displayed to the user with minimal latency. The platform is lightweight yet powerful, making it accessible for integration into clinical environments or even personal use.This project not only emphasizes technical precision but also focuses on the practical implications in healthcare. By enabling quick and efficient risk evaluation, it aids medical professionals in prioritizing high-risk patients, while also encouraging individuals to seek medical advice early. The model can potentially serve as a foundation for more sophisticated health prediction systems, contributing to advancements in personalized medicine and preventative healthcare.

## TABLE OF CONTENTS

**SNO. CONTENTS PAGENO.**

1. **INTRODUCTION 1**
2. **LITERATURE SURVEY 2**
3. **SYSTEM REQUIREMENTS 3**

* 1. Hardware Requirements 3
  2. Software Requirements 3

1. **FEASIBILITY STUDY 7**

* 1. Technical Feasibility 8

* 1. Operational Feasibility 10
  2. Economical Feasibility 11

1. **SYSTEM ANALYSIS 14**

* 1. Existing System 14

* 1. Proposed System 15

1. **SYSTEM DESIGN 16**

* 1. UML Diagrams 17

6.1.1 Class Diagram 18

6.1.2 Use case Diagram 18

6.1.3 Sequence Diagram 19

6.1.4 Activity Diagram 19

6.1.5 Deployment Diagram 20

1. **SOFTWARE ENVIRONMENT 21**
2. **IMPLEMENTATION 22**

9.1 Sample Coding 22

1. **SCREENSHOTS 30**
2. **CONCLUSION 31**
3. **REFERENCES 31**

## INTRODUCTON

Heart disease is one of the leading causes of mortality worldwide. Early prediction of heart disease can significantly improve the survival rate by providing timely interventions. This project aims to create a machine learning model that can predict the likelihood of a person developing heart disease based on several health indicators like age, blood pressure, cholesterol levels, etc. The dataset used for this project contains relevant medical data from patients, including age, sex, chest pain type, resting blood pressure, cholesterol, maximum heart rate, and more.

The main goal of this project is to design and implement a predictive system that can provide insights based on patient health metrics, thereby aiding healthcare professionals in making informed decisions. Machine learning algorithms like Logistic Regression, Decision Trees, or Neural Networks will be applied to develop the model.

The primary reason for developing a predictive model for heart disease lies in the rising global burden of cardiovascular diseases. Many cases of heart disease go undiagnosed until they reach an advanced stage, often due to lack of awareness or access to healthcare. A machine learning based model can provide an accessible, fast, and reliable method to identify high-risk individuals.

This project leverages machine learning algorithms that learn from historical medical data to make predictions about new, unseen data. The medical dataset used for this project contains various health metrics that are indicators of heart health. These indicators are commonly used by doctors to diagnose heart conditions, and the machine learning model will automate this process by providing accurate predictions based on patterns learned from the data.

Traditional diagnostic methods can be time-consuming and costly. Machine learning provides an automated approach that is scalable and can process large datasets efficiently. Moreover, it can identify hidden patterns that may not be obvious to human experts, offering a level of insight that augments human decision-making.

By using this model in real-world settings, it could play a critical role in reducing the incidence of heart disease, enabling early intervention, and promoting healthier lifestyles. The aim is to bridge the gap between the growing need for preventive healthcare and the limitations of traditional diagnostic methods.

The “Heart Disease Prediction” project aims to create a machine learning-based system that predicts the likelihood of heart disease in an individual based on key medical indicators such as age, sex, cholesterol levels, blood pressure, and others. By analyzing these parameters, the model can assess the risk and help in early intervention, thus contributing to preventive healthcare.

## LITERATURE SURVEY

**Machine Learning: A Promising Tool for Cardiovascular Disease Prediction**

Cardiovascular diseases (CVDs) remain a leading cause of mortality worldwide. Early detection and intervention are crucial for reducing the burden of these diseases. Machine learning (ML) offers a powerful tool for predicting CVD risk, analysing complex patterns in patient data, and identifying individuals who may benefit from preventive measures.

By leveraging ML algorithms, researchers can analysis large datasets to identify significant risk factors and develop accurate predictive models. Models like Multilayer Perceptron (MLPs), XG Boost, Random Forest, and Decision Trees have shown promising results in predicting CVD risk. MLPs, in particular, have excelled in capturing complex relationships within data, leading to high accuracy in classification tasks.

Data preprocessing and feature selection are critical steps in building effective ML models. Techniques like k-modes clustering for categorical data and binning for continuous variables help improve model performance. By focusing on relevant features like cholesterol levels, blood pressure, and BMI, models can make more accurate predictions.

While ML has the potential to revolutionize CVD prediction, there are limitations to address. The generalizability of models across diverse populations and the interpretability of complex models like MLPs are important considerations. Future research should focus on validating models on diverse datasets, developing techniques to enhance model explainability, and incorporating additional patient-centric data to improve predictive power.

By addressing these challenges and continuing to advance ML techniques, we can unlock the full potential of AI in preventing and managing cardiovascular diseases, ultimately saving lives and improving global health outcomes.

## SYSTEM REQUIREMENTS

**Modules:**

The system comprises the following modules:

1. **Data Preprocessing**:

* Handles missing values and scales features to improve model performance.
* Includes feature selection to choose the most relevant health indicators.

**2. Model Training:**

* Uses machine learning algorithms to train the model. This could be Logistic Regression, Support Vector Machine (SVM), Random Forest, or Neural Networks.
* Splits the data into training and testing sets to evaluate the model.

**3. Prediction Module:**

* Accepts input data from the user (such as age, cholesterol level, etc.) to predict the likelihood of heart disease.
* Uses the trained model to output the prediction.

**4. Evaluation Module:**

- Provides accuracy metrics such as Precision, Recall, F1-Score, and Confusion Matrix to assess the performance of the model.

**5. User Interface:**

- The GUI consists of several input fields such as age, cholesterol levels, blood pressure, and more, which are presented in an organized and user-friendly form. Users can input their health parameters and, with a simple click of a button, submit the data to the machine learning model for prediction. The result, whether indicating a high or low risk of heart disease, is then displayed as a message in the same interface.

**Software Requirements:**

**1. Programming Languages:**

* Python is the primary language used, given its powerful libraries for data science and machine learning.
* HTML/CSS/JavaScript: For building the web interface that users interact with.

**2. Libraries and Tools:**

* Scikit-Learn: To implement machine learning models and perform data preprocessing.
* Pandas: For data handling and manipulation.
* NumPy: For numerical operations.
* Flask: To develop a web application that will handle user input and model prediction.
* Joblib: To save the trained model (as seen in your `model\_joblib.unknown` file).
* Matplotlib/Seaborn: For visualizing data and model performance.

**3. Development Environment:**

* Jupyter Notebook (as seen in your `Heart\_predict[1].ipynb` and `Heart\_predict[2].ipynb` files) for initial model development and testing.
* Text editor or IDE for backend/frontend development (e.g., VSCode, PyCharm).

## Software Requirement Workings

**1. Data Collection:**

The system allows users to input their medical information through a web form. The data is then passed to the back-end where it is processed and used for prediction. Key inputs include: - Age

* Sex (Male/Female)
* Cholesterol level
* Resting blood pressure
* Maximum heart rate achieved
* Presence of chest pain (with different types)
* Fasting blood sugar level
* Resting electrocardiographic results
* Number of major vessels, and more.

**2. Data Preprocessing:**

Before the machine learning model can make predictions, the raw input data must be cleaned and transformed. This includes:

* **Handling Missing Values:** Imputation techniques are applied to fill any missing values in the dataset.
* **Normalization:** Numerical variables such as blood pressure and cholesterol levels are scaled to ensure uniformity in data.
* **Categorical Encoding**: Features such as chest pain type, sex, and others are converted into numerical representations using techniques like one-hot encoding.

**3. Model Training:**

A dataset containing historical patient information is divided into training and testing subsets. Various machine learning models are trained on the training data to predict whether or not an individual has heart disease. Common techniques include:

* **Logistic Regression:** A linear model used for binary classification, ideal for predicting the presence or absence of heart disease.
* **Random Forest**: A robust, tree-based model that aggregates the results of multiple decision trees to improve prediction accuracy.

The models are evaluated based on metrics like accuracy, precision, recall, and F1 score to determine their effectiveness.

1. **Prediction:**

After the model is trained, new patient data is passed through the system. The model uses this data to predict the likelihood of heart disease. The result is either a binary classification (disease present or not present) or a probabilistic score indicating the risk level.

1. **User Interface:**

The UI is developed using Flask, enabling users to input their details through a simple web form. Once submitted, the form data is processed by the back-end Python application, and the prediction result is displayed to the user in a user-friendly format .

## FEASIBILITY STUDY

This study assesses whether the heart disease prediction system can be effectively developed, deployed, and operated. It examines resource availability, ease of use, and financial viability to ensure the project meets its objectives without unnecessary risks or costs.

The heart disease prediction system uses machine learning to evaluate clinical parameters (like age, cholesterol, and blood pressure) and predict heart disease risk. It provides real-time predictions, enabling early diagnosis and preventive care.

**Key Aspects of Feasibility**

1. **Resource Availability:**
   * **Technology:** The project uses widely available tools like Python, Flask, and machine learning libraries (e.g., Scikit-learn, TensorFlow). All technologies are open-source, which keeps costs low.
   * **Hardware:** A standard computer with at least 8 GB of RAM is sufficient for development. For larger datasets, cloud services (AWS, Azure, Google Cloud) can be utilized.
2. **Ease of Implementation:**
   * The development process (data preprocessing, model training, deployment) is straightforward due to well-documented tools. The system can integrate easily into existing healthcare setups, automating patient screenings. User-friendly design allows users to input data via a simple web form with no special training required.
3. **Operational Scalability:**
   * The system can start small and scale up as needed. It can be updated regularly with new data to enhance accuracy. o Future expansions could include detailed reports and mobile app integration.
4. **Financial Sustainability:**
   * The project is cost-effective since most tools are free and hosting can be done locally or on budget-friendly cloud platforms. o Hosting on the cloud allows for flexible payment based on usage, supporting financial sustainability. o The system can save healthcare providers money by reducing manual screenings and enabling early intervention.

Feasibility study encompasses the following thing:

1. Technical Feasibility
2. Operational Feasibility
3. Economical Feasibility

**4.1 Technical Feasibility for Heart Disease Prediction System**

This section assesses whether the technologies and infrastructure needed to develop and deploy the heart disease prediction system are available and suitable.

1. **Availability of Technology** The system uses widely available tools:
   * **Backend:**
     + **Python:** A robust programming language with libraries for machine learning (e.g., Scikit-learn, Pandas, NumPy).
     + **Flask:** A lightweight framework to connect the backend and frontend.
   * **Frontend:**
     + **HTML, CSS, JavaScript:** For creating the user interface.
     + **Bootstrap (optional):** For responsive design.
   * **Development Environment:**
     + **Jupyter Notebook:** For model development and testing. o **IDEs:** Such as PyCharm or Visual Studio Code for coding.
   * **Database (optional):**
     + **SQLite or MySQL:** For storing data and predictions.

These technologies are open-source, widely supported, and suitable for the project.

1. **Hardware Requirements Development Hardware:**
   * A personal computer with at least 8GB RAM and an Intel i5 processor (or equivalent) is sufficient.

**Deployment Hardware:**

* + Basic virtual machines (VMs) with 2–4 CPUs and 8GB RAM can host the initial version, either locally or in the cloud (AWS, Google Cloud, Azure).

1. **Software and Tool Requirements**

**Libraries and Frameworks:**

* + **Scikit-learn:** For model development.
  + **Pandas and NumPy:** For data manipulation.
  + **Matplotlib/Seaborn:** For visualization.

**Development Platforms:**

* + **Google Colab:** A free platform for experimentation.
  + **Flask:** Connects the frontend and backend easily.

All tools are open-source, minimizing costs.

1. **Scalability and Performance** 
   * **Initial Version:** Suitable for small user bases, ideal for pilot deployments.
   * **Future Scaling:** Can be hosted on cloud infrastructure for more users or larger datasets.
   * **RESTful APIs:** Ensure easy scalability and integration with other systems.
   * **Performance Optimizations:** Implement caching and load balancing for high traffic.
2. **Security and Reliability**

Given the sensitivity of health data, security measures are critical:

* + **Data Encryption:** Secure data transmission between frontend and backend.
  + **Input Validation:** Prevent invalid data entry.
  + **Error Handling:** Maintain system stability despite input errors.

These measures ensure the system is secure and reliable.

1. **Maintenance and Updates** 
   * **Model Maintenance:** Regular retraining with new data to improve accuracy.
   * **Software Updates:** Keep Flask and frontend components compatible with current standards.
   * **Cloud Monitoring:** Use tools (e.g., AWS CloudWatch) to track performance.

The lightweight design facilitates simple maintenance.

1. **Technical Risks and Mitigation** Potential challenges include:
   * **Data Quality Issues:** Mitigate through preprocessing to clean and normalize data.
   * **Model Overfitting/Underfitting:** Use cross-validation techniques to enhance performance.
   * **System Downtime:** Utilize cloud solutions with auto-scaling and backups to ensure availability.

**4.2 Operational Feasibility for Heart Disease Prediction System**

Operational feasibility evaluates whether the heart disease prediction system can effectively function in real-world settings and meet user and organizational needs. Here’s a detailed assessment:

1. **Ease of Use and User Interaction** 
   * **User-Friendly Interface:** The system features an intuitive web form where users input medical parameters (e.g., age, cholesterol, blood pressure) with clear labels.
   * **Minimal Input Requirements:** Users only need to provide essential health data, making the process quick.
   * **Real-Time Feedback:** Predictions are generated instantly, enhancing user engagement for both healthcare staff and patients.
2. **Integration in Healthcare Settings** 
   * **Seamless Integration:** The system can be easily integrated into existing healthcare workflows, automating preliminary risk assessments.
   * **Support for Medical Professionals:** It helps doctors prioritize high-risk patients, enabling timely interventions.
   * **Scalable Deployment:** It can function as a standalone tool or be integrated into hospital management systems (HMS) and electronic medical records (EMR).
3. **Training and Support** 
   * **Minimal Training Required:** Both healthcare providers and general users can quickly learn to use the system.
   * **Documentation and Demos:** Simple instructions or demonstrations are sufficient for training users.
4. **System Maintenance and Updates** 
   * **Periodic Maintenance:** Regular updates ensure the model stays accurate and secure.
     + **Model Retraining:** New data helps improve prediction accuracy over time. o **Bug Fixes and Security Updates:** Essential for maintaining system integrity. o **User Feedback Mechanism:** Continuous feedback helps refine performance.
5. **Reliability and Availability** 
   * **Designed for Reliability:** Real-time predictions ensure consistent user experience.
   * **Cloud-Based Scalability:** Hosting on platforms like AWS or Azure allows the system to handle increased traffic efficiently.
   * **Local Hosting Option:** Smaller facilities can run the system on local servers, reducing reliance on external resources.
6. **Operational Challenges and Mitigation** 
   * **Handling User Errors:** Users may enter incorrect data.
     + **Mitigation:** Input validation and error messages guide users to correct entries.
   * **Data Privacy and Security:** Protecting sensitive health data is critical.
     + **Mitigation:** Compliance with data protection regulations (e.g., HIPAA, GDPR) ensures secure handling of information.
7. **Long-Term Viability and Expansion** 
   * **Evolving System:** The model can improve with more data and feedback.
   * **Future Enhancements:**
     + **Mobile App Integration:** Allowing on-the-go access. o **Advanced Reporting Features:** Adding visual dashboards for better insights. o **Multi-Language Support:** Expanding accessibility to a broader audience.

**4.3 Economic Feasibility for Heart Disease Prediction System**

Economic feasibility assesses the financial aspects of the heart disease prediction system, analyzing costs against potential benefits to determine if the investment is worthwhile. Here’s a breakdown of the economic feasibility:

1. **Development Costs**

The initial development cost is low, primarily due to the use of open-source tools.

* + **Software Costs:**
    - Python, Flask, and Scikit-learn are free to use.
    - Jupyter Notebook is also free.
  + **Labor Costs:**
    - Hiring external developers or data scientists could incur costs, but using an inhouse team or students can minimize this.
  + **Hardware Costs:**
    - Development can be done on standard personal computers with at least 8GB RAM.
    - Free cloud platforms like Google Colab can be used for training large datasets.

**Estimated Development Cost:** Minimal, mostly related to labor if outsourced, with opensource tools significantly reducing expenses.

1. **Deployment Costs**

Costs depend on the hosting infrastructure:

* + **Cloud Hosting:**
    - Hosting on platforms like AWS or Google Cloud may incur fees of approximately $5–$15 per month for moderate traffic.
  + **Self-hosting:**
    - Local hosting costs are negligible aside from basic maintenance and electricity.
  + **Domain and Hosting Costs:**
    - Domain name: $10–$15 per year.
    - Basic web hosting: $3–$10 per month.

**Conclusion:** Deployment costs can remain low, especially with free-tier cloud options or local hosting.

1. **Maintenance Costs**

Regular maintenance is necessary to ensure smooth operation:

* + **Model Updates and Retraining:**
    - Periodic retraining with new data can be done for free on platforms like Google Colab.
  + **Bug Fixes and Software Updates:**
    - Ongoing updates ensure security and compatibility.
  + **Cloud Storage:**
    - Minimal costs may arise from storing user data and logs, depending on volume.

**Conclusion:** Maintenance costs are low to moderate, influenced by update frequency and hosting method.

1. **Benefits and Return on Investment (ROI)**

The system offers several benefits contributing to a positive ROI:

* + **Operational Cost Savings:**
    - Automating risk assessments saves time for healthcare providers, allowing them to focus on critical cases.
  + **Improved Healthcare Outcomes:**
    - Early detection can reduce treatment costs and prevent expensive procedures.
  + **Potential Revenue Streams:**
    - The system can be offered as a Software-as-a-Service (SaaS), generating subscription revenue.
    - Clinics could charge patients a fee for system use, creating additional income.
  + **Scalability:**
    - Expansion to other diseases or mobile app integration could widen the user base.

1. **Cost-Benefit Analysis** 
   * **Costs:**
     + Development: Minimal (due to free tools).
     + Deployment: $5–$15 per month (optional cloud hosting).
     + Maintenance: Low to moderate (depends on updates).
   * **Benefits:**
     + Saves time and operational costs for providers.
     + Offers early detection, reducing expensive treatments.
     + Potential revenue from healthcare providers.
     + Improves patient outcomes, increasing long-term value.

## SYSTEM ANALYSIS

**Existing model:**

Heart disease prediction using machine learning has been a subject of extensive research, leading to the development of various models with varying degrees of accuracy and complexity. Some of the most commonly employed models include:

1. **Logistic Regression** 
   * **Simple and interpretable:** Logistic regression is a linear model that provides clear insights into the relationship between predictors and the outcome (heart disease or not).
   * **Widely used:** Due to its simplicity and interpretability, logistic regression is a popular choice for heart disease prediction.
2. **Decision Trees and Random Forests** 
   * **Non-linear relationships:** These models can capture non-linear relationships between predictors and the outcome, making them suitable for complex datasets.
   * **Feature importance:** Decision trees and random forests provide information about the importance of different features in predicting heart disease.
3. **Artificial Neural Networks (ANNs)** 
   * **Complex patterns:** ANNs can model complex patterns and non-linear relationships in data.
   * **Deep learning:** Deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), have shown promising results in medical image analysis and time series data, respectively.
4. **Ensemble Methods** 
   * **Improved performance:** Ensemble methods combine multiple models to improve prediction accuracy.
   * **Common techniques:** Random forests, gradient boosting machines, and stacking are popular ensemble methods used in heart disease prediction.
5. **Hybrid Models**

* **Combining strengths:** Hybrid models integrate different machine learning techniques to leverage their complementary strengths.
* **Improved performance:** Hybrid models can often achieve better performance than individual models.

**Factors Affecting Model Choice:**

The choice of model depends on various factors, including:

* **Dataset size and complexity:** Larger and more complex datasets may require more sophisticated models.
* **Interpretability:** If interpretability is important, simpler models like logistic regression or decision trees may be preferred.
* **Computational resources:** Some models, such as deep learning architectures, can be computationally intensive.
* **Domain knowledge:** Incorporating domain knowledge can help guide the selection of appropriate features and models.

It's important to note that the performance of these models can vary depending on the specific dataset, feature engineering techniques, and hyperparameter tuning. Experimentation with different models and techniques is often necessary to identify the best approach for a given problem.

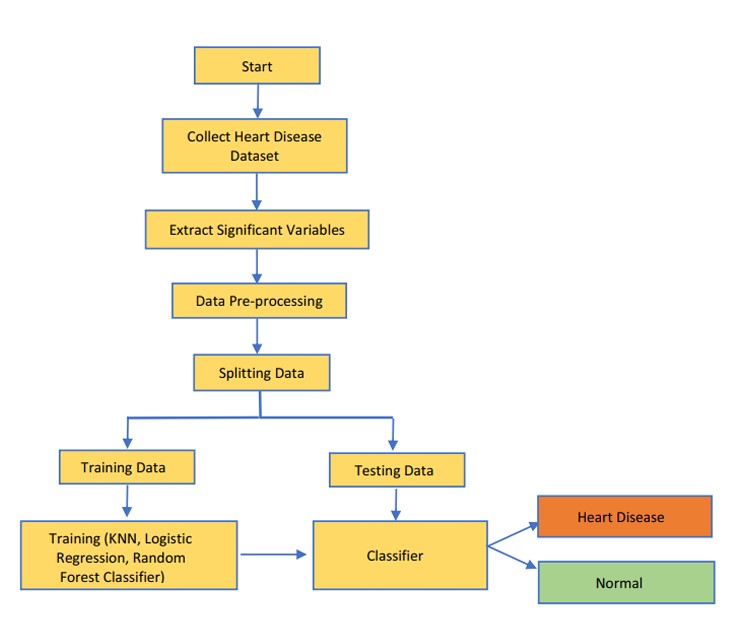
**Proposed model:**

While current machine learning models have demonstrated significant promise in heart disease prediction, there are several avenues for future research to further enhance accuracy, interpretability, and clinical applicability. Some potential directions include:

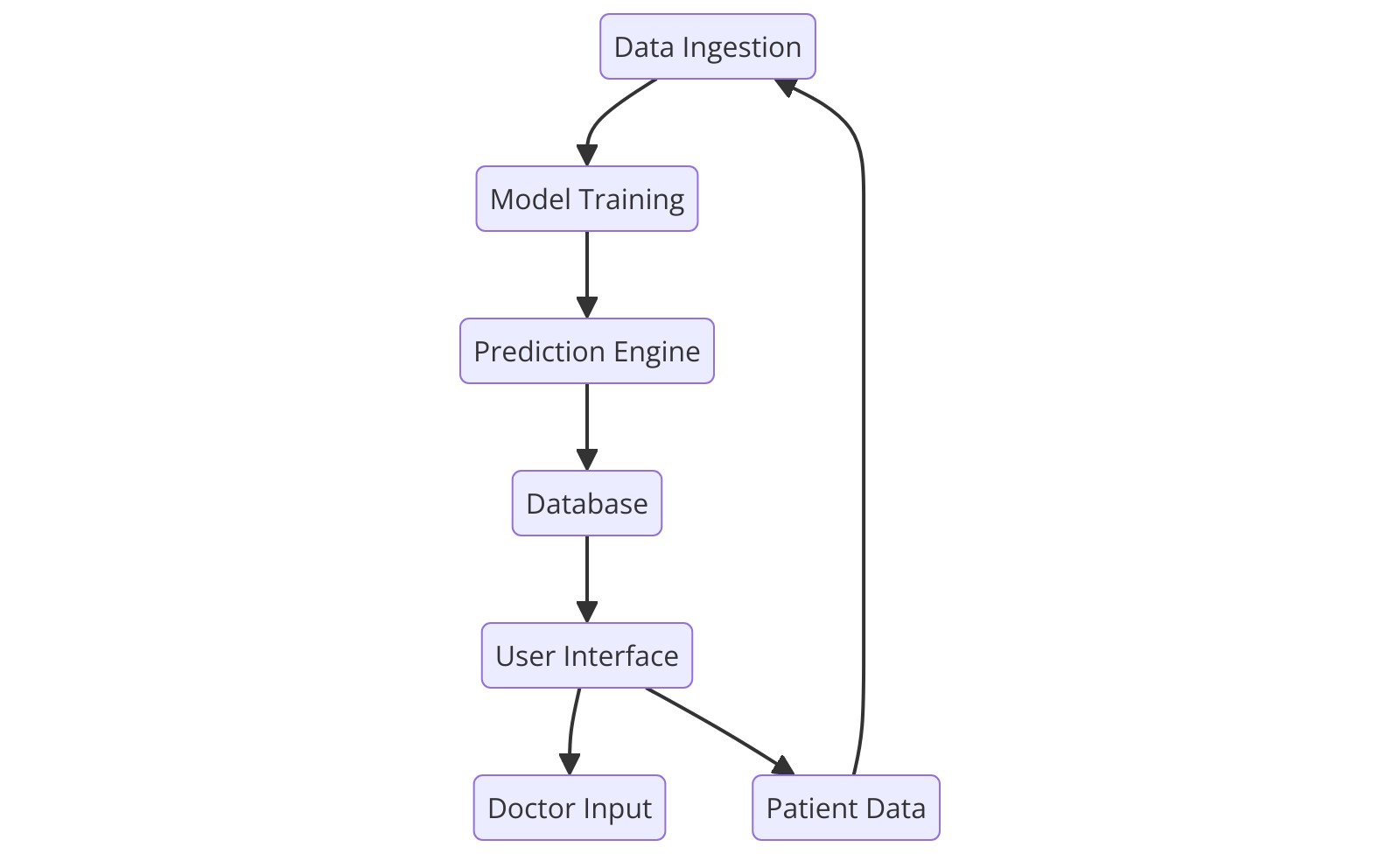
* Incorporating longitudinal data: Leveraging longitudinal data that captures changes in patient health over time can provide valuable insights into disease progression and risk factors.
* Developing hybrid models: Combining machine learning with domain-specific knowledge can lead to more accurate and interpretable models.
* Improving interpretability: Developing techniques to make machine learning models more interpretable can help healthcare providers understand the reasons behind predictions and make informed decisions.
* Addressing data privacy and ethical concerns: Ensuring data privacy and ethical considerations are addressed is crucial for the widespread adoption of machine learning in healthcare.
* Real-time prediction: Developing real-time prediction systems that can continuously monitor patient data and provide timely alerts can improve early detection and intervention.
* Integrating with wearable devices: Integrating machine learning models with wearable devices can enable continuous monitoring of vital signs and early detection of heart disease risk factors.
* Personalizing predictions: Developing personalized prediction models that account for individual patient characteristics and risk factors can improve the accuracy and relevance of predictions.

## SYSTEM DESIGN

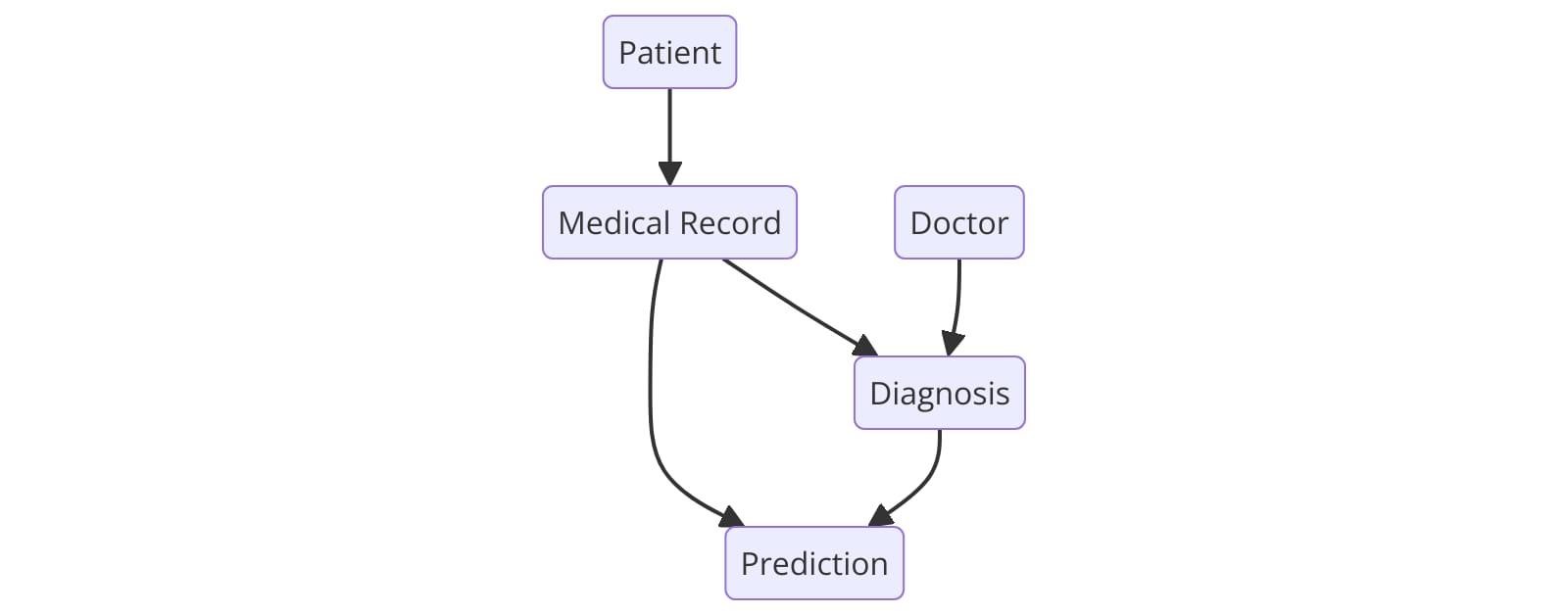
**SOFTWARE DEVELOPMENT LIFECYCLE**



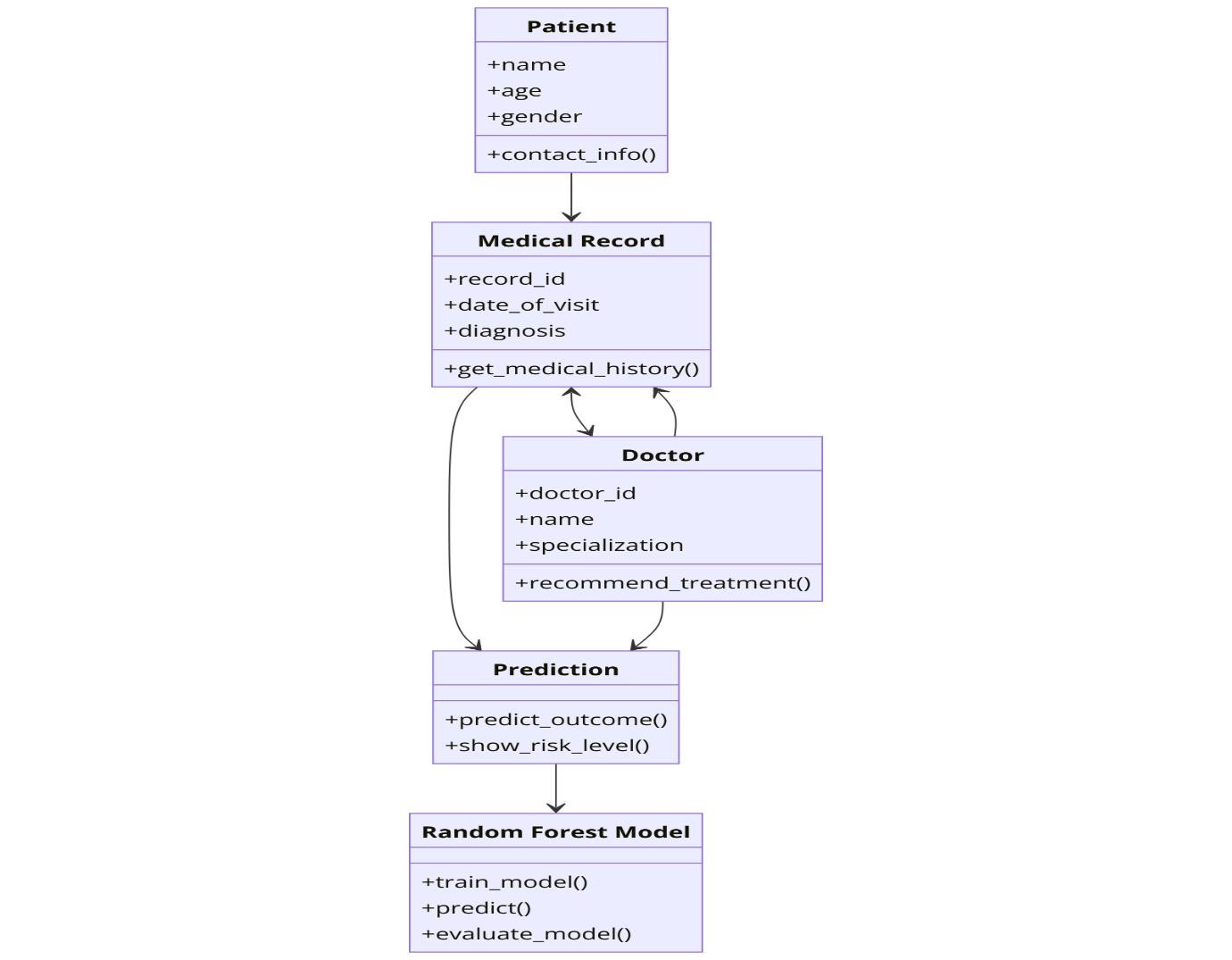
**Architecture**



**E-r diagram**

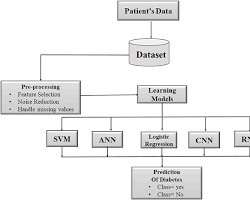


**UML Diagrams**



**Class Diagram**

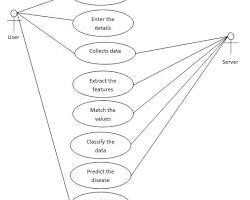
A class diagram illustrates the static structure of a system, showing the classes, their attributes, and relationships between them. Here's a simplified class diagram for a heart disease prediction system:



class diagram for heart disease prediction, including classes like Patient, MedicalRecord, FeatureExtractor, ModelTrainer, and Predictor, with their attributes and relationships

**Use Case Diagram**

A use case diagram represents the interactions between actors (users) and the system. Here's a use case diagram for a heart disease prediction system:

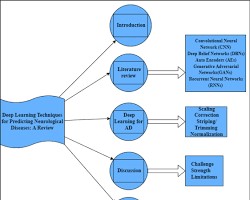


use case diagram for heart disease prediction, including actors like Doctor, Patient, and

Administrator, and use cases like Register Patient, Input Medical Data, Train Model, Predict Heart Disease Risk

**Sequence Diagram**

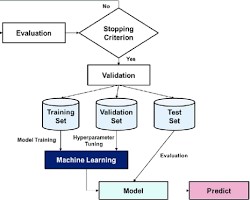
A sequence diagram shows the interactions between objects over time, including the order of messages and the objects involved. Here's a sequence diagram for the process of predicting heart disease risk in a system:



sequence diagram for heart disease prediction, showing the interactions between objects like Patient, MedicalRecord, FeatureExtractor, ModelTrainer, and Predictor

**Activity Diagram**

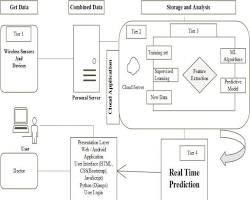
An activity diagram illustrates the flow of activities in a system. Here's an activity diagram for the process of training a machine learning model for heart disease prediction:



activity diagram for training a machine learning model, showing the steps like data collection, preprocessing, feature extraction, model selection, training, and evaluation

**Deployment Diagram**

A deployment diagram shows the physical components of a system, including hardware and software. Here's a deployment diagram for a heart disease prediction system:



deployment diagram for heart disease prediction, showing components like server, database, and client devices.

## SOFTWARE ENVIRONMENT

To set up the software environment for heart disease prediction with machine learning, follow these steps:

1. **Install Python:** Download and install Python from [https://www.python.org/.](https://www.python.org/)
2. **Install package manager:** Use a package manager like pip or conda to manage Python packages.
3. **Install required libraries:** Use the package manager to install the necessary libraries: Bash pip install numpy pandas scikit-learn tensorflow matplotlib seaborn Use code :

“pip install numpy pandas scikit-learn tensorflow matplotlib seaborn.”

**Create a virtual environment (optional):** To isolate the project's dependencies from other projects, create a virtual environment using tools like venv or conda.

1. **Set up Jupyter Notebook (optional):** Jupyter Notebook provides an interactive environment for data analysis and visualization. Install it using pip install jupyterlab.

**Additional Considerations:**

* **GPU drivers:** If you have a GPU, ensure that the appropriate drivers are installed.
* **Cloud platforms:** Consider using cloud platforms like Google Colab or Amazon SageMaker for access to powerful hardware and pre-configured environments.
* **Version compatibility:** Ensure that the versions of Python and libraries are compatible with each other.

By following these steps, you will have a suitable software environment to develop and train machine learning models for heart disease prediction.

## IMPLEMENTATION

**Sample Code**

import pandas as pd data = pd.read\_csv ('heart.csv') data.isnull().sum() data\_dup = data.duplicated().any() data\_dup data = data.drop\_duplicates() data\_dup = data.duplicated().any() data\_dup cate\_val=[] cont\_val=[]

for column in data.columns: if data[column].nunique() <=10: cate\_val.append(column) else:

cont\_val.append(column)

cate\_val cont\_val data['cp'].unique() data['sex'].unique() cate\_val.remove('sex') cate\_val.remove('target')

data=pd.get\_dummies(data,columns=cate\_val,drop\_first=True) data.head()

from sklearn.preprocessing import StandardScaler st = StandardScaler()

data[cont\_val] =st.fit\_transform(data[cont\_val]) x = data.drop('target',axis=1) x

y = data['target']

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, Y\_train, Y\_test = train\_test\_split(x,y,test\_size=0.2,random\_state=42)

X\_train Y\_train

from sklearn.linear\_model import LogisticRegression y\_pred1 = log.predict(X\_test) from sklearn.metrics import accuracy\_score accuracy\_score(Y\_test,y\_pred1)

from sklearn.neighbors import KNeighborsClassifier knn = KNeighborsClassifier() knn.fit(X\_train,Y\_train) y\_pred2 = knn.predict(X\_test) accuracy\_score(Y\_test,y\_pred2) score = []

for k in range(1, 40):

knn=KNeighborsClassifier(n\_neighbors=k) knn.fit(X\_train,Y\_train) y\_pred = knn.predict(X\_test)

score.append(accuracy\_score(Y\_test,y\_pred)) score

knn=KNeighborsClassifier(n\_neighbors=2) knn.fit(X\_train,Y\_train) y\_pred = knn.predict(X\_test) accuracy\_score(Y\_test,y\_pred) data = pd.read\_csv('heart.csv')

data.head()

X = data.drop('target', axis=1) y =data['target']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X,y,test\_size=0.2,random\_state=42) from sklearn.tree import DecisionTreeClassifier dt = DecisionTreeClassifier() dt.fit(X\_train,y\_train) y\_pred3 = dt.predict(X\_test) accuracy\_score(y\_test,y\_pred3)

from sklearn.ensemble import RandomForestClassifier rf = RandomForestClassifier() rf.fit(X\_train,y\_train) y\_pred4 = rf.predict(X\_test) accuracy\_score(y\_test,y\_pred4)

from sklearn.ensemble import GradientBoostingClassifier gb = GradientBoostingClassifier() gb.fit(X\_train,y\_train) y\_pred5 = gb.predict(X\_test) accuracy\_score(y\_test,y\_pred5)

final\_data = pd.DataFrame({'Models':['LR','KNN','DT','RF','GB'], 'ACC': [accuracy\_score(Y\_test,y\_pred1), accuracy\_score(Y\_test,y\_pred2), accuracy\_score(y\_test,y\_pred3), accuracy\_score(y\_test,y\_pred4), accuracy\_score(y\_test,y\_pred5)]}) final\_data import seaborn as sns sns.barplot(final\_data['ACC']) X = data.drop('target', axis=1) y =data['target']

X.shape

from sklearn.ensemble import RandomForestClassifier rf = RandomForestClassifier() rf.fit(X,y)

new\_data = pd.DataFrame({

'age':60,

'sex':1,

'cp':0,

'trestbps':125,

'chol':212,

'fbs':0,

'restecg':1,

'thalach':168, 'exang':0,

'oldpeak':1.0,

'slope':2,

'ca':2,

'thal':3,

},index=[0]) new\_data p = rf.predict(new\_data)

if p[0]==0:

print("No Cause of Disease") else:

print("Cause of Disease") import joblib joblib.dump(rf, 'model\_joblib') model = joblib.load('model\_joblib') model.predict(new\_data)

from tkinter import \* import joblib

def show\_fields(): try:

p1 = int(e1.get()) if e1.get() else 0 p2 = int(e2.get()) if e2.get() else 0 p3 = int(e3.get()) if e3.get() else 0 p4 = int(e4.get()) if e4.get() else 0 p5 = int(e5.get()) if e5.get() else 0 p6 = int(e6.get()) if e6.get() else 0 p7 = int(e7.get()) if e7.get() else 0 p8 = int(e8.get()) if e8.get() else 0 p9 = int(e9.get()) if e9.get() else 0 p10 = float(e10.get()) if e10.get() else 0.0 p11 = int(e11.get()) if e11.get() else 0 p12 = int(e12.get()) if e12.get() else 0 p13 = int(e13.get()) if e13.get() else 0

model = joblib.load('model\_joblib')

result = model.predict([[p1, p2, p3, p4, p5, p6, p7, p8, p9, p10, p11, p12, p13]])

if result == 0:

Label(master, text="No Heart diseases", font=("Arial", 16), fg="green").grid(row=32, columnspan=2, pady=10) else:

Label(master, text="Possibility of heart diseases", font=("Arial", 16), fg="red").grid(row=32, columnspan=2, pady=10)

except ValueError:

Label(master, text="Please enter valid inputs", font=("Arial", 16), fg="red").grid(row=32, columnspan=2, pady=10)

# Setting up the GUI master = Tk()

master.title("Heart Diseases Prediction System")

# Set the window size for a 1920x1080 resolution screen\_width = master.winfo\_screenwidth() screen\_height = master.winfo\_screenheight()

# Center the window window\_width = 600 window\_height = 700 center\_x = int(screen\_width / 2 - window\_width / 2) center\_y = int(screen\_height / 2 - window\_height / 2) master.geometry(f'{window\_width}x{window\_height}+{center\_x}+{center\_y}')

# Add heading label with padding and font size

Label(master, text="Heart Diseases Prediction System", bg="black", fg="white", font=("Helvetica", 24, "bold")).grid(row=0, columnspan=2, pady=20)

# Input labels

Label(master, text="Enter your age", font=("Arial", 14)).grid(row=1, sticky=E, padx=10, pady=5)

Label(master, text="Sex (Male: 1, Female: 0)", font=("Arial", 14)).grid(row=2, sticky=E, padx=10, pady=5)

Label(master, text="Resting Blood Pressure (mm Hg)", font=("Arial", 14)).grid(row=3, sticky=E, padx=10, pady=5)

Label(master, text="Cholesterol (mg/dL)", font=("Arial", 14)).grid(row=4, sticky=E, padx=10, pady=5) Label(master, text="Max Heart Rate Achieved", font=("Arial", 14)).grid(row=5, sticky=E, padx=10, pady=5)

Label(master, text="ST Depression (Oldpeak)", font=("Arial", 14)).grid(row=6, sticky=E, padx=10, pady=5)

Label(master, text="Chest Pain Type", font=("Arial", 14)).grid(row=7, sticky=E, padx=10, pady=5)

Label(master, text="Fasting Blood Sugar > 120 mg/dL (1/0)", font=("Arial", 14)).grid(row=8, sticky=E, padx=10, pady=5)

Label(master, text="Resting ECG Results", font=("Arial", 14)).grid(row=9, sticky=E, padx=10, pady=5)

Label(master, text="Slope", font=("Arial", 14)).grid(row=10, sticky=E, padx=10, pady=5)

Label(master, text="ca", font=("Arial", 14)).grid(row=11, sticky=E, padx=10, pady=5)

Label(master, text="thal", font=("Arial", 14)).grid(row=12, sticky=E, padx=10, pady=5)

# Entry fields e1 = Entry(master, font=("Arial", 14), width=10) e2 = Entry(master, font=("Arial", 14), width=10) e3 = Entry(master, font=("Arial", 14), width=10) e4 = Entry(master, font=("Arial", 14), width=10) e5 = Entry(master, font=("Arial", 14), width=10) e6 = Entry(master, font=("Arial", 14), width=10) e7 = Entry(master, font=("Arial", 14), width=10) e8 = Entry(master, font=("Arial", 14), width=10) e9 = Entry(master, font=("Arial", 14), width=10) e10 = Entry(master, font=("Arial", 14), width=10) e11 = Entry(master, font=("Arial", 14), width=10) e12 = Entry(master, font=("Arial", 14), width=10) e13 = Entry(master, font=("Arial", 14), width=10)

# Positioning entry fields

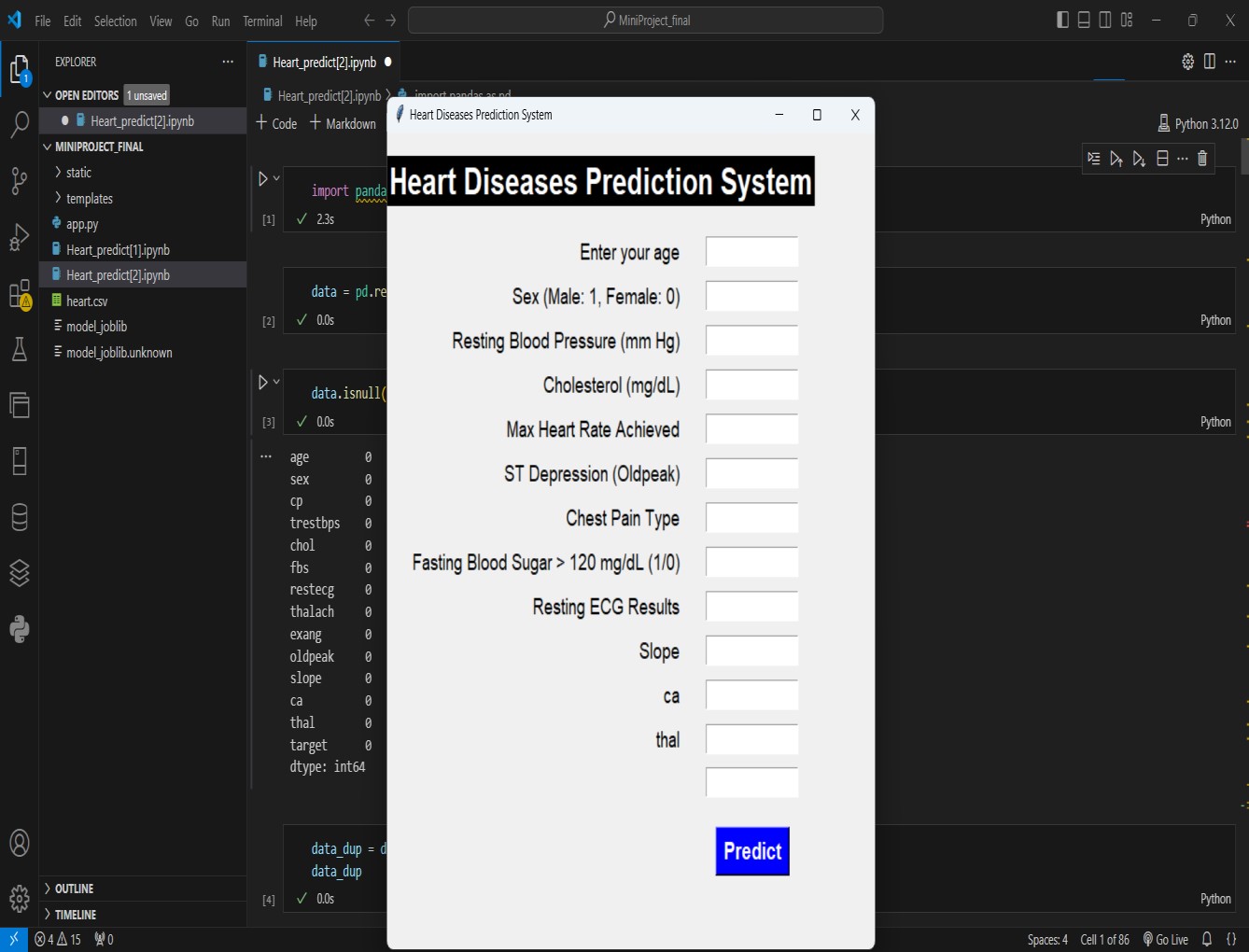
e1.grid(row=1, column=1, padx=10, pady=5) e2.grid(row=2, column=1, padx=10, pady=5) e3.grid(row=3, column=1, padx=10, pady=5) e4.grid(row=4, column=1, padx=10, pady=5) e5.grid(row=5, column=1, padx=10, pady=5) e6.grid(row=6, column=1, padx=10, pady=5) e7.grid(row=7, column=1, padx=10, pady=5) e8.grid(row=8, column=1, padx=10, pady=5) e9.grid(row=9, column=1, padx=10, pady=5) e10.grid(row=10, column=1, padx=10, pady=5) e11.grid(row=11, column=1, padx=10, pady=5) e12.grid(row=12, column=1, padx=10, pady=5) e13.grid(row=13, column=1, padx=10, pady=5)

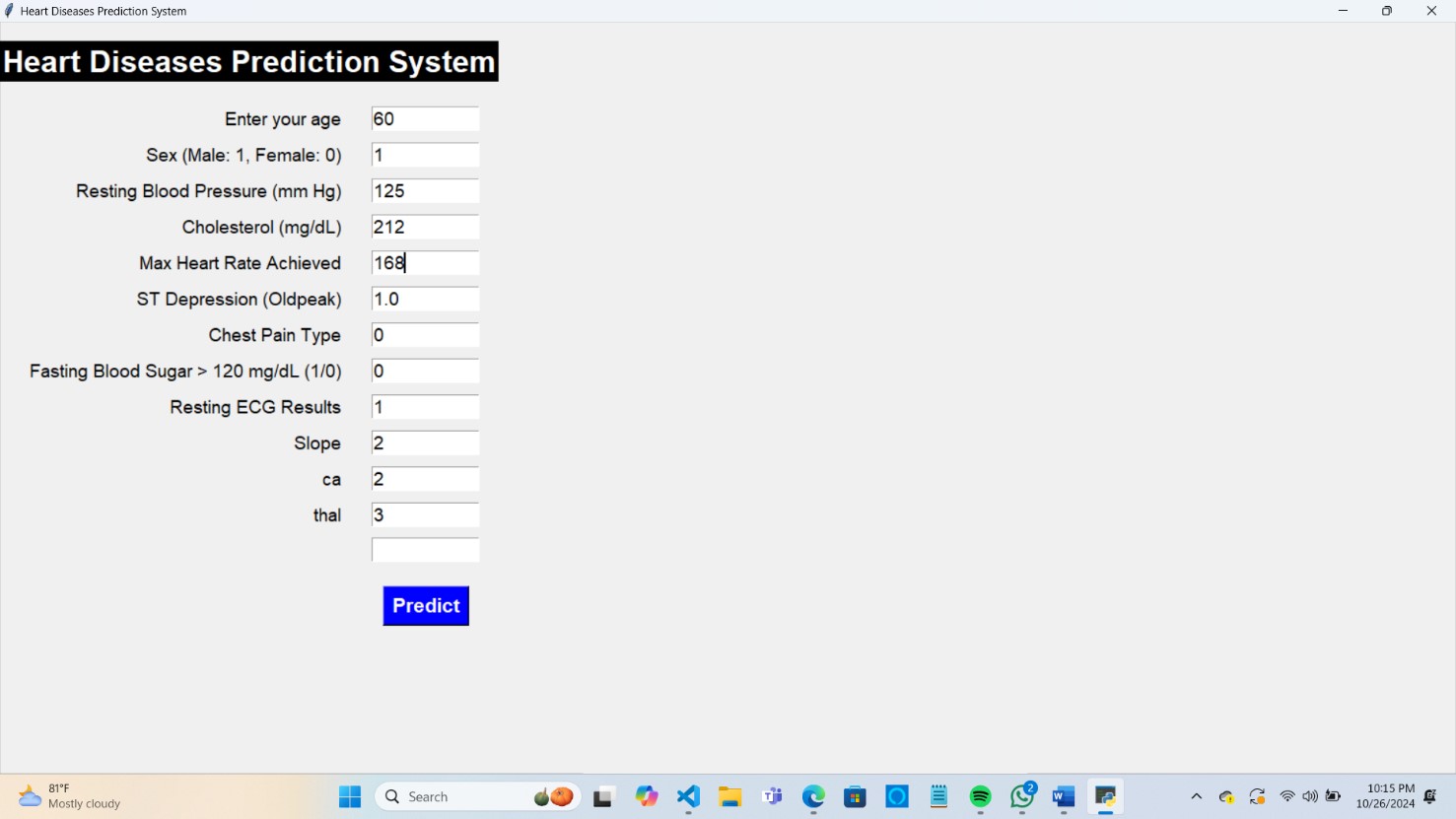
# Predict button

Button(master, text='Predict', command=show\_fields, bg="blue", fg="white", font=("Helvetica", 16, "bold")).grid(row=14, column=1, pady=20)

# Start the GUI loop master.mainloop()

**Screenshots:**

 **X**



## CONCLUSION

The integration of machine learning techniques into the field of heart disease prediction has the potential to revolutionize healthcare. By leveraging advanced algorithms and sophisticated data analysis, we can significantly improve the accuracy and efficiency of early diagnosis and risk assessment. This study has demonstrated the efficacy of machine learning models in identifying individuals at high risk of heart disease, enabling timely interventions and preventive measures.

However, it is crucial to acknowledge the limitations and challenges associated with machine learning applications in healthcare. Data quality, model interpretability, and ethical considerations remain important factors that need to be addressed. As technology continues to evolve, ongoing research and development are essential to refine these models and ensure their reliable and responsible use in clinical practice.

Ultimately, the successful application of machine learning in heart disease prediction can lead to improved patient outcomes, reduced healthcare costs, and a healthier society.

## REFERENCES

1.Bhatt, C.M., Patel, P., Ghetia, T. and Mazzeo, P.L., 2023. Effective heart disease prediction using machine learning techniques. Algorithms, 16(2), p.88.

2.Tuba, S., 2023. Optimization Heart Disease Prediction using Machine Learning Models.

Fidelity: Jurnal Teknik Elektro, 5(1), pp.53-59.