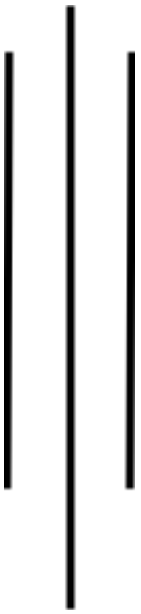
School of Engineering & Technology

Asian Institute of Technology

AT82.03 - Machine Learning



Date: 10-31-2023

Heart Disease Classification: Project Proposal

Submitted To
Dr. Chaklam Silpasuwanchai

Submitted By:

Anawat Pradabsuk	**st124039**
Ritik Sareen	**st123844**
Tanzil AL Sabah	**st123845**
Chatpol Phonyen	**st123992**
Munthitra Thadtjapong	**st124022**
Md Shafi Ud Doula	**st124047**

Index

Introduction

1. Background **2**
2. Motivation **2**

Related Work

1. Health Monitoring Wearables **2**
2. Cardiolog **3**
3. Framingham Heart Study **3**

Problem **4**

Solution

1. Rationale **4**
2. Value Proposition **4**
3. Target Audience **5**
4. Requirement Elicitation **5**
5. Feasibility **5**
6. Risks **5**

Mockups **6-8**

System Design & Architecture **8**

1. Training Process **9**
2. Version Controlling **10**
3. Deployment Process **10-12**

Use Case & Sequence Diagram **12-13**

Team Assignment **13**

Evaluation

1. Performance **14**
2. Human **14**

Reference **15**

Introduction

1. Background

Cardiovascular diseases (CVDs) remain the leading cause of death worldwide, claiming approximately 17.9 million lives annually. Traditional diagnostic tools, such as ECGs and angiograms, have been instrumental but come with limitations: they can be invasive, expensive, and sometimes inaccessible to many, especially in remote or underserved regions. In recent years, the fusion of digital technology with healthcare has ushered in an era where vast amounts of health-related data, including metrics from wearable devices, have become available.

2. Motivation

The potential of machine learning in revolutionizing heart disease detection is immense. With its capability to identify patterns and make accurate predictions from large datasets, machine learning promises earlier and more precise detection of heart ailments. Leveraging datasets like the UCI Heart Disease dataset can pave the way for crafting sophisticated prediction models. The goal is simple yet profound: early detection, timely interventions, and a significant reduction in the global impact of heart disease.

Related works

1. Health Monitoring Wearables

Wearables like Apple Watches, Fitbit devices can incorporate ECG monitoring and can notify users of irregular heart rhythms, potentially indicative of atrial fibrillation.

Figure-1: A physical fitness monitors by Fitbit

2. Cardiolog

Offers an ECG analysis platform powered by AI to detect arrhythmias and other conditions.

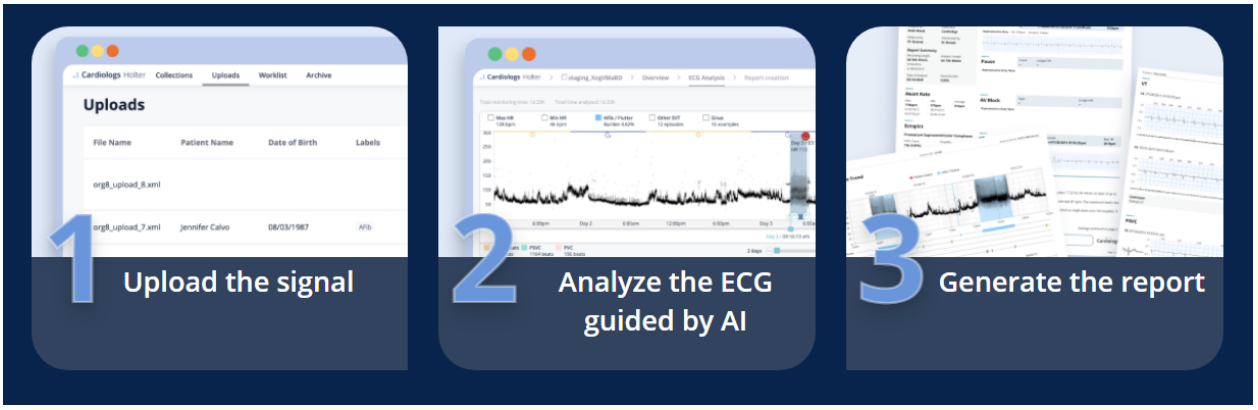


Figure-2: CardioLog AIbanalyzing and interpreting cardiovascular data, offering insights and generate report

3. Framingham Heart Study

A long-term, ongoing cardiovascular cohort study that has been a source for various ML projects due to its comprehensive datasets.

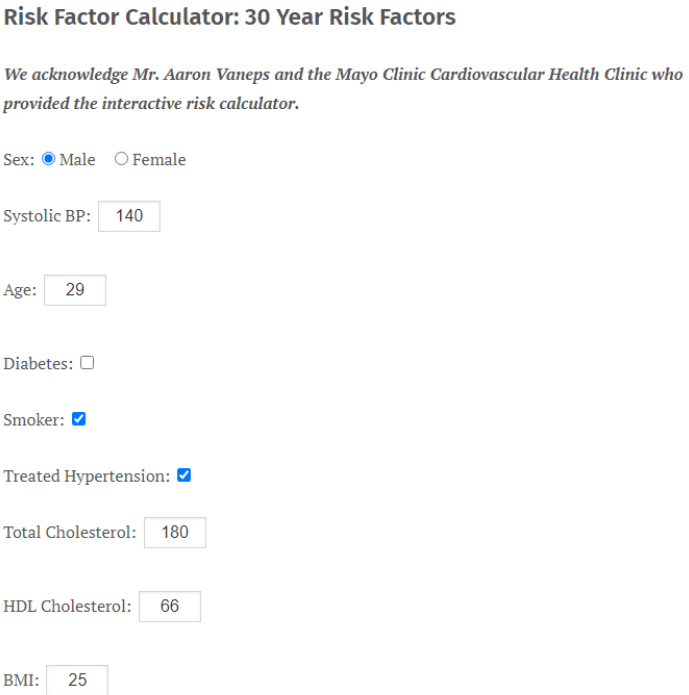


Figure-3: Framingham Heart Study API UI

Problem

- Many health wearables provide insights into heart rate and variability, but these metrics are not always linked directly to clinical outcomes. There's a gap in wearables being a wellness tool versus a medically validated diagnostic aid.

- ECG Analysis by AI is just one piece of the puzzle. Combining it with other clinical metrics can potentially yield a more holistic understanding of a patient's cardiovascular health.

- The Framingham Heart Study, though groundbreaking, primarily includes data from residents of Framingham, Massachusetts. There's a potential bias as the dataset might not reflect the global diversity of heart disease risk factors and manifestations.

- Some of the data might not align with modern lifestyle factors and risks associated with heart diseases today.

Leveraging the UCI dataset's more recent data, in conjunction with modern machine learning techniques, can yield predictive models attuned to current lifestyle factors and medical paradigms.

Solution

1. Rationale

Our product from the UCI Heart Disease Data is a diagnostic tool applying supervised machine learning to determine whether a patient is prone to have heart disease. The idea behind this product is to utilize the UCI Heart Disease Data Set to produce an effective classification model that patients and healthcare providers may use so that it could help to detect and diagnose heart disease likelihood, ultimately leading to enhanced patient outcomes and life saving heart surgery.

2. Value Proposition

- Early Detection: It is possible to identify the probability of cardiac problems, resulting in available intervention and treatment.

- Accurate Classifications: The machine learning model applies a comprehensive dataset to create an accurate classification based on given features.

- User-Friendly Interface: The product's user-friendly interface makes it accessible to patients as well as healthcare professionals.

- Data-Driven Insights: It provides insights and information from the dataset that can help a better comprehension of the factors affecting heart disease.

Target Audience

- Cardiologists and Healthcare Professionals: To assist in diagnosing and managing heart disease in patients.
- Patients: To evaluate their risk of heart disease and take proactive steps for precaution.

Requirement Elicitation

- To classify the likelihood of heart disease based on a machine learning model.
- Add a user-friendly interface for simplicity of usage.
- Offer explanations for model predictions.
- Update the model frequently to increase precision.

Feasibility

Developing a machine learning model for heart disease prediction is feasible within a 2-month timeframe. The UCI Heart Disease Data Set is well-documented and suitable for model training. The availability of machine learning libraries and frameworks simplifies the development process. However, maintaining data privacy and security may require additional time and resources.

Risks

- Data Privacy Concerns: Handling sensitive medical data may raise privacy and security issues.
- Model Accuracy: The chosen model may face challenges in delivering accurate predictions
- Data Quality: The dataset's quality and completeness could impact the model's performance.
- Ethical Considerations: Ensuring fairness and avoiding bias in classifications is crucial.
- User Adoption: Healthcare professionals and patients may not adopt the tool for various reasons, including distrust in AI-based systems or a lack of awareness.

Mockups

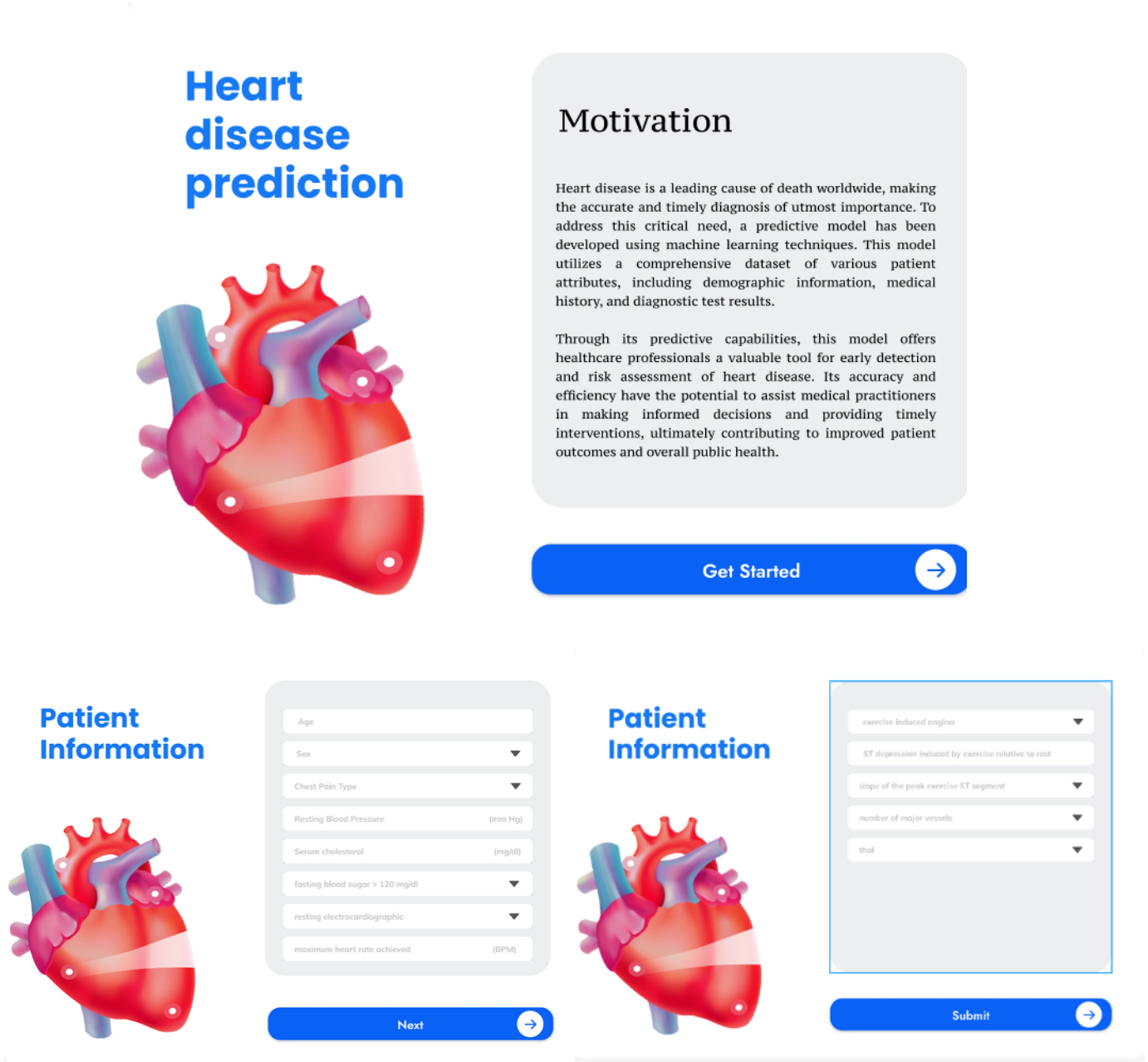


Figure-4: Desktop API UI - User input their personal information

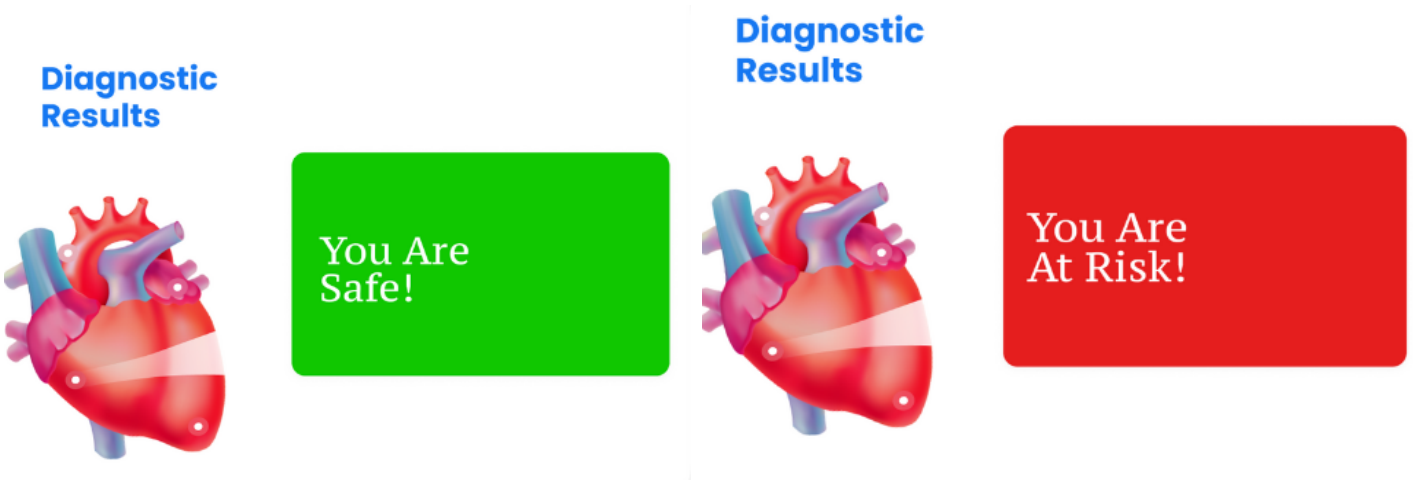


Figure-5: Desktop API UI - Return diagnostic results either at risk or safe

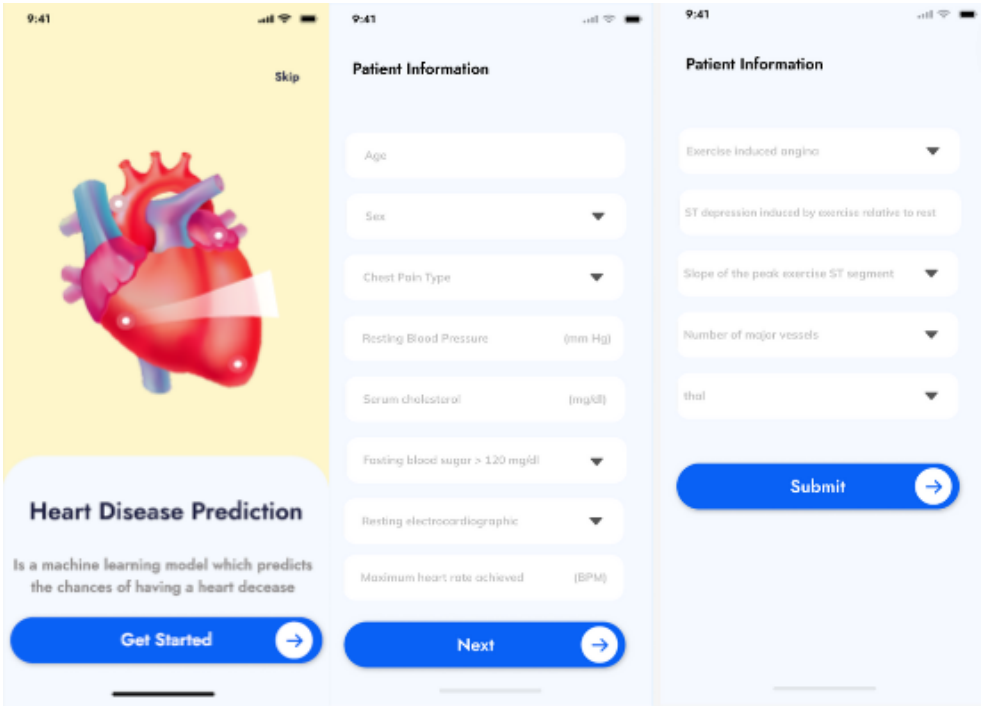


Figure-6: Phone API UI - User input their personal information

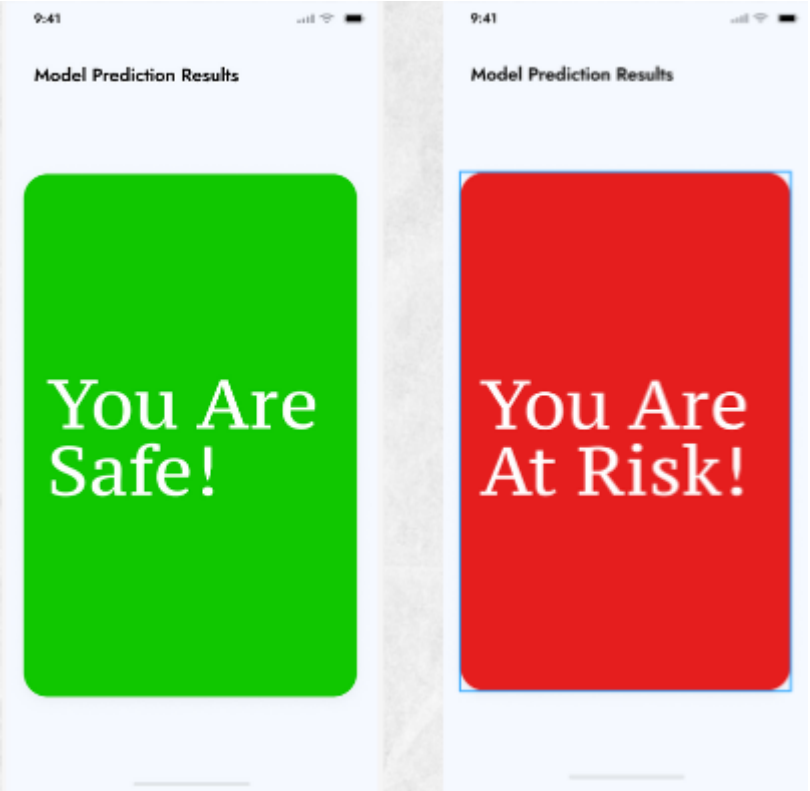


Figure-7: Phone API UI - Return diagnostic results either at risk or safe

System Design & Architecture

In this section, our goal is to propose a process for training and deploying a Heart Disease Stage Prediction ML model (classification) with a primary focus on the deployment process. We have considered the following assumptions for preparing this architecture and flow diagrams-

- Keep the deployment process as simple as possible with minimal changes required.
- Maintain low cost for deployment.

We have divided this document into three sections:

1. **Training Process**
2. **Version Controlling**
3. **Deployment Process**

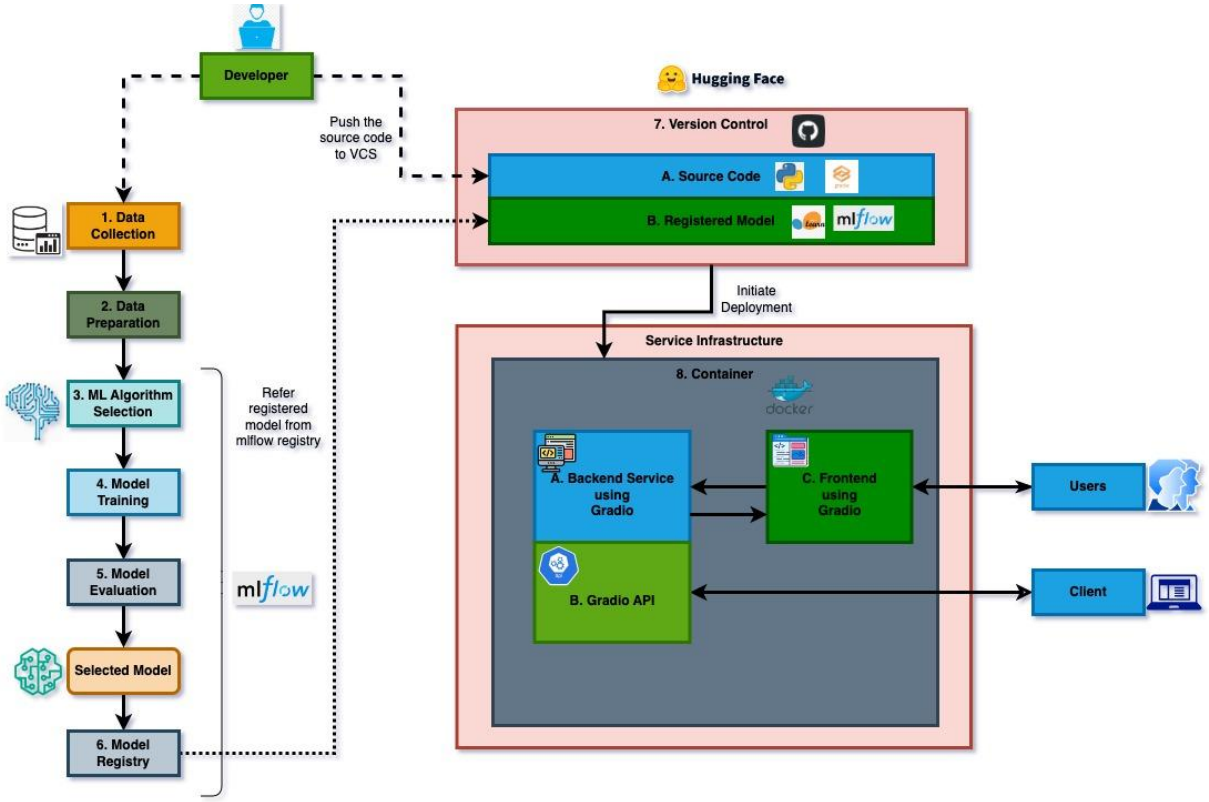


Figure-8: Process for Heart Disease Stage Prediction ML Model Training and Deployment

Infrastructure Description

Task	Platform	Specification
Model training	Linux Machine	• vCPU = 16 • Memory = 64 GB
Model deployment	Huggingface Spaces	• vCPU = 2 • Memory = 16 GB

1. Training Process

1.1. Data Collection:

- Download the dataset from data source UCI[1]
- Ensure the necessary libraries like pandas and scikit-learn are installed in the Python environment.

1.2. Data Preparation:

- **Data Exploration:**
  - Load the dataset into a pandas DataFrame.
  - Explore the data to understand its structure and content.
    There are a total 303 samples and each sample has 13 features.
    There are 4 types of target classes- integers valued from 0 (no presence) to 4.
- **Data Preprocessing:**
  - Preprocess the data, including encoding categorical variables if needed (e.g., one-hot encoding, label encoding).

- **Data Splitting:**
  - Split the data into training and testing sets to evaluate the model's performance.
  - A typical 80-20 split will be used, with 242 samples for training and 61 for testing out of the total 303.
  - Check for missing values and handle them if necessary.

1.3. ML Algorithm Selection:

- With MLflow, Choose a suitable machine learning algorithm. For heart disease prediction, consider algorithms like logistic regression, decision trees, or random forests.
- Employ MLflow to perform model selection through cross-validation. Use techniques like k-fold cross-validation (e.g., k=5 or k=10) to assess the model's performance across multiple subsets of the training data.
- MLflow will enable us to search for the best hyperparameters for the selected Algorithm. Utilize techniques like Grid Search to systematically explore different combinations of hyperparameters.
- Leveraging MLflow, this step helps in finding the best configuration for the model, which can significantly impact its performance.

1.4. Model Training:

- With MLflow, train the selected algorithm using the best hyperparameters on the training data, utilizing scikit-learn functions.
- MLflow provides a structured environment for tracking and managing the training process, ensuring the model is trained effectively.

1.5. Model Evaluation (With MLflow):

- Evaluate the model's performance on the test dataset using metrics like accuracy, precision, recall, and F1-score.
- The choice of the best model depends on specific priorities:
  - To minimize false positives, prioritize high precision.
  - To capture as many positives as possible, prioritize high recall.
  - For a balanced trade-off, consider the F1-score.

We are planning to prioritize high precision for heart disease type classification to minimize the risk of misdiagnosing a healthy patient with a specific heart disease.

2. Version controlling

While creating a space for deployment in huggingface (HF)[6], a repository is automatically created for version control for source code with the registered model of MLflow model registry of the application. Huggingface space maintains source versioning using the popular VCS git[5]. So, working with HF Space for version control is the same as using github like VCS.

3. Deployment Process

The proposed deployment process for a monolithic architecture, focusing on a popular framework Gradio.

Using Gradio

Gradio[7] is an open-source framework that simplifies the development of machine learning-based applications with minimal GUI development requirements. The framework provides a streamlined approach to deploying both the backend and frontend components of the application. To deploy using Gradio, it's essential to ensure the repository follows this specific file structure:

Figure-9: File Structure of Gradio based Deployment

- **App.py:** This Python script serves as the core of the application and comprises two distinct sections for the backend and frontend.
    - **Backend (App.py - Backend)**
        - Loads the scikit-learn model from MLflow model registry.
        - Defines the inference function.
        - Specifies input and output validation rules.
        - Handles preprocessing and postprocessing of input and output.
    - **Frontend (App.py - Frontend)**
        - Defines GUI components using predefined classes.
        - Renders the web interface using a Gunicorn server.
- **requirements.txt:** This text file includes a list of libraries required to run the application, such as scikit-learn, numpy, mlflow and others.
- **MLflow Model Registry :** This is the scikit-learn model that has been trained in the previous steps and registered in the MLflow Model Registry and is prepared for deployment after rigorous testing.

In summary, the process can be described as follows:

1. **Code Push to VCS**
   Developers push the code to the Version Control System (VCS) within Huggingface Spaces.

2. **Docker Image Creation**
   Upon code push, an automated process triggers the creation of a Docker image[8].
3. **Image Build Success**
   The build process verifies the image's successful construction.
4. **Container Initialization**
   Once the Docker image is built successfully, a container is initialized.
5. **Service Availability**
   With the container up and running, the service becomes accessible for users.
6. **User Interaction**
   Users can interact with the service through a web-based user interface (UI) or by making requests to the service's API.

Use Case & Sequence Diagram

This use case involves user input of health parameters, followed by the system's feature extraction and data classification to predict heart disease[2,3]. The results are then displayed to the user.

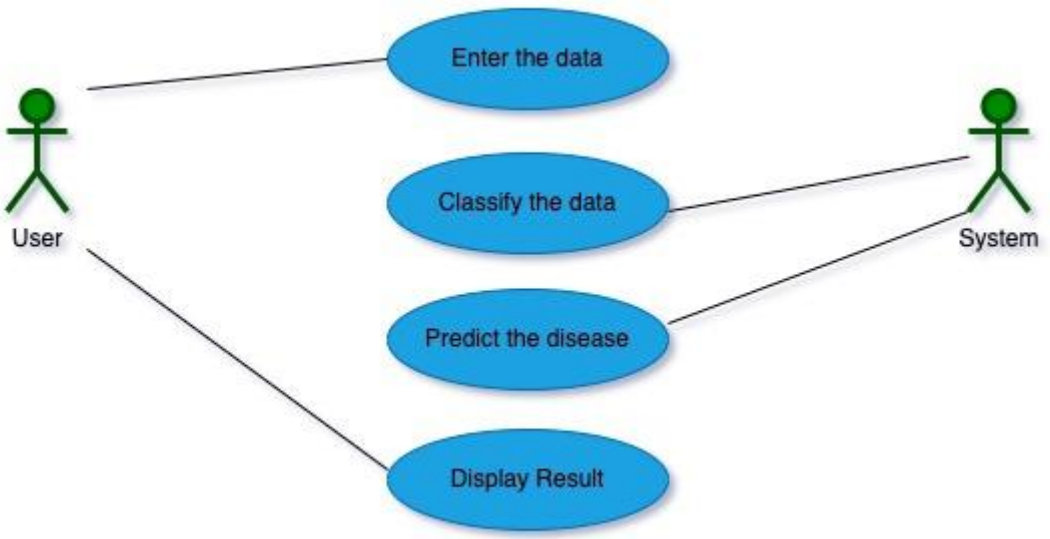


Figure-10: Heart Disease Prediction Workflow

- User Inputs Data:
  - The user provides input data, which could include various health-related data.

- System Classifies Data:
  - The system processes the user's input data and extracts relevant features or characteristics from it.

  - Using the extracted features, the system classifies the data, determining the different heart disease stages or conditions.

- System Predicts the Disease:
    - Based on the classification, the system predicts the specific heart disease stage or condition that the user may have.

- User Displays Result:
    - The system presents the prediction or result to the user, providing information about their heart disease stage based on the input parameters.

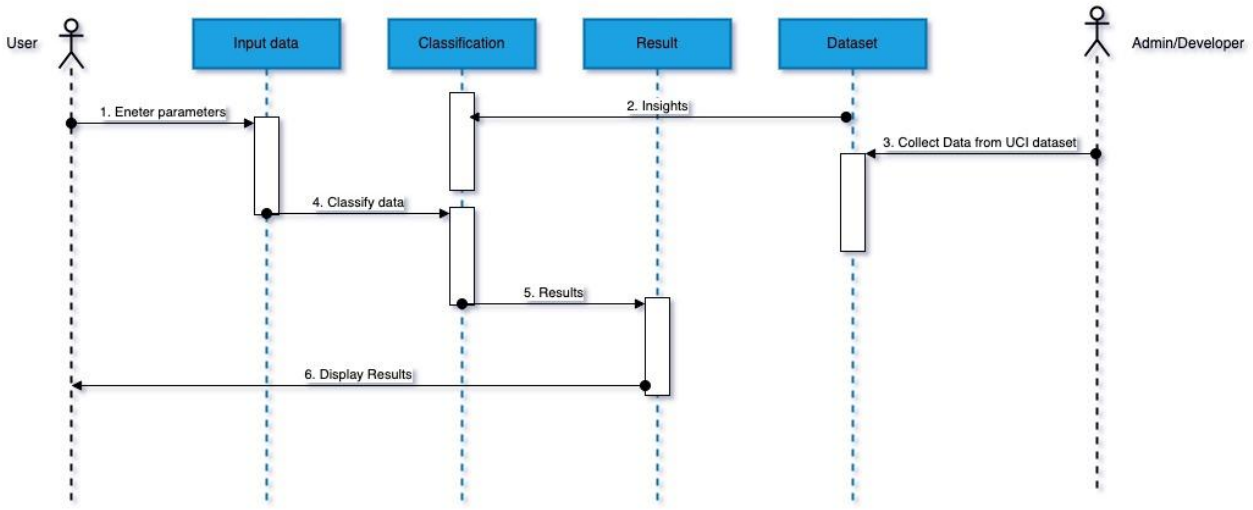


Figure-11: Heart Disease Prediction Sequence Diagram

Figure-11 the sequence diagram[2,3] shows how to perform the activity and how to respond to the user. This diagram also shows how the data is sent to the model.

Team Assignment

Project: Streamlined Deployment of Heart Disease Prediction ML Model

Project Duration	11 Days
Start Date	11/1/2023
End Date	11/11/2023

Team Structure

Serial No.	Name	Responsibity
1	Anawat	Project Co-ordinator
2	Chatpool	
3	Munthitra	
4	Shafi	
5	Tanzil	
6	Ritik	

Task ID	Task Name	Subtask	Resource	Duration (day)	Start Date	End Date	Completion
	Training Process						
1	Data Collection	Download dataset from UCI data source	1. Munthitra 2. Anawat	1	11/1/2023	11/1/2023	
2	Data Exploration	Ensure necessary libraries are installed	1. Tanzil 2. Chatpol	1	11/2/2023	11/2/2023	
3		Load dataset into a pandas DataFrame, Explore data structure and content.	1. Tanzil 2. Chatpol	0.5	11/3/2023	11/3/2023	
4	Data Splitting & handing missing Value	Split data into training and testing sets, Use an 80-20 split (242 samples for training, 61 for testing)	1. Anawat	0.5	11/3/2023	11/3/2023	
5		Check for missing values and handle them if necessary	1. Anawat 2. Munthitra	0.5	11/4/2023	11/4/2023	
6	Data Preprocessing	Preprocess data, including encoding categorical variables if needed	1. Ritik 2. Shafi	0.5	11/4/2023	11/4/2023	
7	Team Meeting-1	Discuss data exploration progress, issues, splitt ratio and next steps	Team	0.5	11/5/2023	11/5/2023	
8	ML Algorithm Selection	Choose suitable ML algorithm for prediction, Perform model selection through cross-validation, Search for best hyperparameters using Grid Search	1. Chatpol 2.Tanzil 3. Shafi	0.5	11/5/2023	11/5/2023	
9	Model Training	Fit selected algorithm with best hyperparameters on the training data	1.Munthitra 2.Ritik	2	11/5/2023	11/7/2023	
10	Model Evaluation	Evaluate model's performance on the test dataset, Choose the best model based on specific priorities	Team	0.4	11/7/2023	11/7/2023	
11		Discuss model evaluation progress, issues, and next steps	Team	0.16	11/7/2023	11/7/2023	
12	Deployment Process	Use Huggingface Spaces for version control, Automatically create a repository for model and source code	1. Shafi 2. Tanzil 3. Anawat	2	11/3/2023	11/5/2023	
13		Develop deployment process with Gradio framework, Ensure the repository follows a specific file structure, Push code to Version Control System (VCS) within Huggingface Spaces, Trigger Docker image creation upon code push, Verify successful image build, Initialize a container for deployment, Ensure service availability, Enable user interaction through web-based UI or API	1. Shafi 2. Tanzil 3. Anawat	1	11/8/2023	11/8/2023	
14	Integration & System Test	Test the Application and verify and validate the system	Team	1	11/8/2023	11/5/2023	
15	Team Meeting-2	Discuss deployment progress, issues, and bug fixings	Team	1	11/9/2023	11/10/2023	
16	Project Report	Compile project report	Team	1	11/11/2023	11/11/2023	

Evaluation

1. Performance

- AUC-ROC: Measure the model's discriminatory ability and performance.
- F1 Score: Balance between precision and recall in predictions.
- Confusion Matrix Analysis: Assess true positives, true negatives, false positives, and false negatives for the model's predictive power.

2. Human

- Usability Testing: Assess ease of use, user satisfaction, and overall experience through surveys and interviews. Most important thing for this API is to be as comfortable as possible while using this API.
- Continuous Improvement: Implement user feedback to enhance the API's user-friendliness.

Reference

1. Heart Disease UCI Dataset (URL: https://archive.ics.uci.edu/dataset/45/heart+disease)

2. Intelligent Heart Disease Prediction on Physical and Mental Parameters: A ML Based IoT and Big Data Application and Analysis [URL: https://www.researchgate.net/publication/339810503_Intelligent_Heart_Disease_Prediction_on_Physical_and_Mental_Parameters_A_ML_Based_IoT_and_Big_Data_Application_and_Analysis

3. Heart Disease Prediction System Using Data Mining Technique. [URL: https://www.irjet.net/archives/V2/i8/IRJET-V2I8232.pdf)

4. scikit-learn. (URL: https://scikit-learn.org/stable/)

5. Version Control System (URL: https://git-scm.com/)

6. Huggingface Spaces. (URL: https://huggingface.co/spaces)

7. Gradio. (URL: https://www.gradio.app/)

8. Docker. (URL: https://www.docker.com/resources/what-container/)