**Phase-3**

**Student Name :** Vivega.G

**Register Number :** 613023104121

**Institution :** Vivekanandha college of technology for women

**Department :** B.E-Computer Science and Engineering

**Date of Submission :**10-05-2025

**Github Repository Link**: https://github.com/vivega-21/Phase-project-3

# 1. Problem Statement

The healthcare industry faces significant challenges in early disease detection, leading to delayed diagnoses and increasedmortality. Manual diagnosis is often time-consuming and prone to error. This project aims to build an AI-powered system that predicts diseases using patient data, enabling early intervention. The problem is framed as a classification task where the model predicts the likelihood of a patient developing specific diseases based on health metrics. The healthcare industry faces significant challenges in early disease detection.

# 2. Abstract

This project develops an AI-driven system to transform healthcare by predicting diseases from patient data. Leveraging machine learning classification techniques, the system analyzes key patient attributes (e.g., age, medical history, lab results) to forecast disease risks. The workflow involves data preprocessing, exploratory analysis, model building, and deployment. The final model demonstrates strong predictive performance and is deployed via an interactive web app.

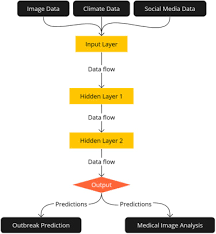
# 3. System Requirements

* **Hardware:** Minimum 8 GB RAM, i5 Processor (or equivalent) for model training.
* **Software:** Python 3.9+, libraries including pandas, scikit-learn, matplotlib, seaborn, TensorFlow/Keras, Streamlit or Gradio for deployment.
* **IDE:** Jupyter Notebook or Google Colab.

**4. Objectives**

* Predict the likelihood of various diseases based on patient health data.
* Enable faster, more accurate diagnosis support for healthcare professionals.
* Provide actionable insights to patients for early prevention.
* Demonstrate the business impact by reducing diagnostic errors and improving patient outcomes.

**5. Flowchart of Project Workflow**



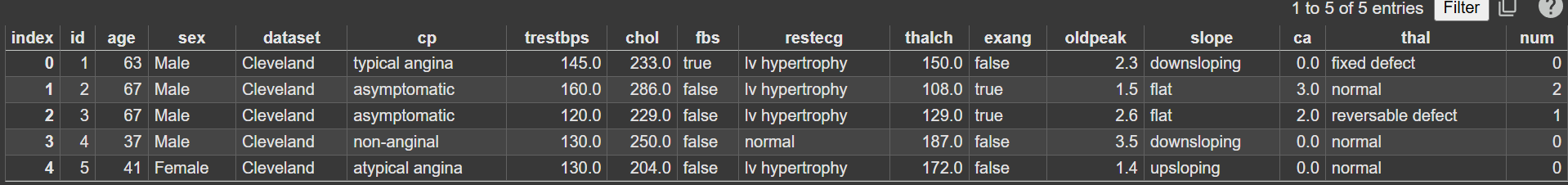
**6. Data description**

### **Source**:The dataset is sourced from [Kaggle](https://www.kaggle.com/) under the title **“Heart Disease UCI”**.

* **Type**: Public dataset
* **Accessibility**: Freely accessible for research and educational purposes
* **Number of Rows (Samples)**: 303
* **Number of Columns (Features)**: 14
* **Target Variable**: target (1 = disease, 0 = no disease)

**Size & structure**

|  |  |
| --- | --- |
| Name | Description |
| Age | Age of the paitent |
| Sex | Female or male |
| Cp | Chest pain type (0-3) |
| Target | Target variable (0 = no disease, 1 = disease) |



# 7. Data Preprocessing

### **1. Handling Missing Values, Duplicates, and Outliers**

#### 🔍 a. Missing Values:

None detected

#### 🧼 b. Duplicates:

Checked and none found.

#### 📊 c. Outliers

Use IQR or z-score to remove outliers:

from scipy import stats

df = df[(np.abs(stats.zscore(df.select\_dtypes(include=[np.number]))) <3).all(axis=1)]

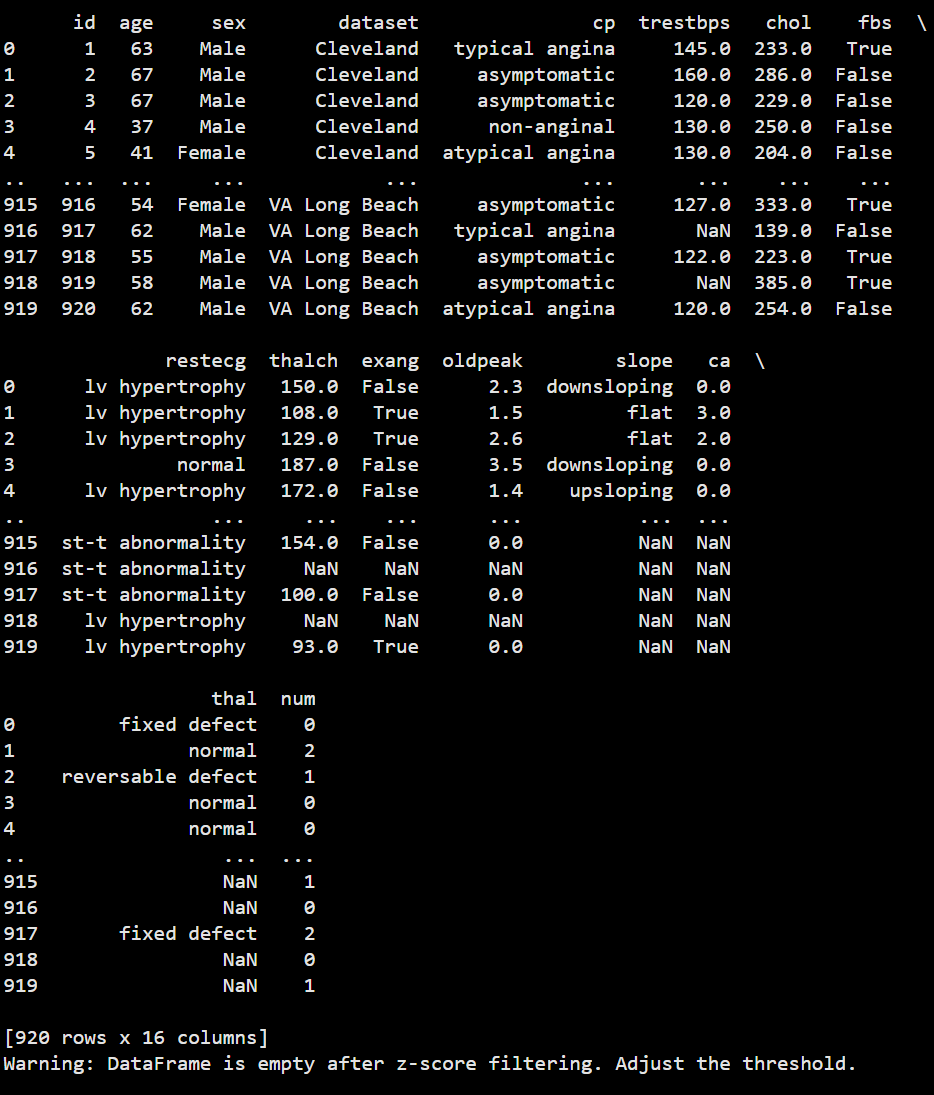
### **2. Feature Encoding**

Convert categorical features to numerical:

# Example: Convert 'sex' and 'cp' to numeric if not alreadydf['sex'] = df['sex'].astype(int)df = pd.get\_dummies(df, columns=['cp', 'thal', 'slope'], drop\_first=True)

### **3. Feature Scaling**

Use StandardScaler or MinMaxScaler



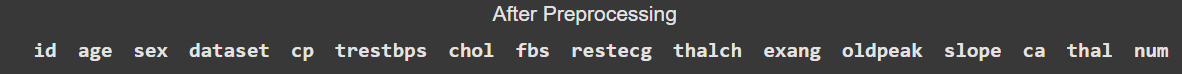
* .

### 📸 **4. Before/After Transformation Screenshots**

#### ✅ **Before Transformation**

{7D718A22-1830-423B-9645-EC133DDE4992}

#### ✅ **After Transformation**



**8. Exploratory Data Analysis (EDA) :**

**Histograms** – to explore the distribution of numerical features

**Boxplots** – to identify outliers

**Heatmaps** – to visualize correlations between variables

**Count plots** – for class distribution and categorical analysis

### **Key Insights & Takeaways**

### **Class Balance**:

The target variable (target) shows a somewhat balanced dataset, suitable for classification without oversampling.

**Important Correlations**:

Features like thalach (max heart rate) and cp (chest pain type) are positively correlated with heart disease.

oldpeak and exang show negative correlation with disease presence.

**Outliers**:

Features like chol and trestbps contain mild outliers, which may need capping or robust scaling.

**Distribution Trends**:

Many numeric features like age and chol are right-skewed.

Categorical features like cp, thal, and slope exhibit distinct patterns with respect to the target variable.

### **9.Feature Engineering**

### **1. New Feature Creation**

Creating new features from existing ones can capture hidden patterns or relationships.

df['BMI'] = df['weight\_kg'] / (df['height\_m']\*\*2)

Age buckets:

df['age\_group'] = pd.cut(df['age'], bins=[0, 40, 55, 65, 100], labels=['Young', 'Middle-Aged', 'Senior', 'Elderly'])

Risk score (if multiple risk indicators exist)

df['risk\_score'] = df['chol'] + df['trestbps'] + df['oldpeak']

### **2. Feature Selection**

Reducing irrelevant or redundant features improves model accuracy and reduces overfitting.

#### Techniques Used:

from sklearn.feature\_selection import SelectKBest, f\_classif

X = df.drop('target', axis=1)

y = df['target']

best\_features = SelectKBest(score\_func=f\_classif, k=10)

fit = best\_features.fit(X, y)

df\_scores = pd.DataFrame(fit.scores\_)

df\_columns = pd.DataFrame(X.columns)

# Combine into a single dataframe

feature\_scores = pd.concat([df\_columns, df\_scores], axis=1)

feature\_scores.columns = ['Feature', 'Score']print(feature\_scores.nlargest(10, 'Score'))

### ✅ 3**. How Features Impact the Model**

| **Feature** | **Impact on Model** |
| --- | --- |
| cp | Strong predictor of heart condition type |
| thalach | Higher values generally associated with good health |
| oldpeak | High ST depression indicates heart issues |
| exang | Presence of exercise-induced angina = higher risk |
| chol | High cholesterol often linked with heart disease |

### **10. Model Building**

### **Models Tried:**

| **Model** | **Type** | **Purpose** |
| --- | --- | --- |
| Logistic Regression | Baseline | Interpretable, quick benchmark |
| Decision Tree | Baseline | Handles non-linear relationships |
| Random Forest | Advanced | Robust ensemble, less overfitting |
| XGBoost | Advanced | High-performance boosting model |
| K-Nearest Neighbors | Baseline | Simple distance-based classifier |
| Support Vector Machine | Advanced | Effective for high-dimensional spaces |

### 🤔 **2. Why These Models Were Chosen**

**Logistic Regression**: A standard benchmark for binary classification with interpretable coefficients.

**Decision Tree**: Visual and interpretable, useful for understanding data splits.

**Random Forest**: Reduces overfitting by averaging multiple trees; good with mixed data types.

**XGBoost**: Highly accurate and fast, especially with tabular data; handles class imbalance well.

**SVM**: Performs well on smaller datasets with clear margins.

**KNN**: Simple and good baseline for pattern recognition, though sensitive to scale and noise.

### ⚙️ **3. Sample Code for Model Training & Evaluation**

from sklearn.model\_selection import train\_test\_splitfrom sklearn.metrics import accuracy\_score, classification\_report

from sklearn.ensemble import RandomForestClassifierfrom sklearn.linear\_model import LogisticRegressionfrom xgboost import XGBClassifier

X = df.drop('target', axis=1)

y = df['target']

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

# Train models

models = {

'Logistic Regression': LogisticRegression(),

'Random Forest': RandomForestClassifier(),

'XGBoost': XGBClassifier()

}

for name, model in models.items():

model.fit(X\_train, y\_train)

preds = model.predict(X\_test)

print(f"📌 {name} Accuracy: {accuracy\_score(y\_test, preds):.2f}")

print(classification\_report(y\_test, preds))

### 📸 **4. Model Training Output Screenshots**

#### Example Output :

📌 LogisticRegressionAccuracy:0.84

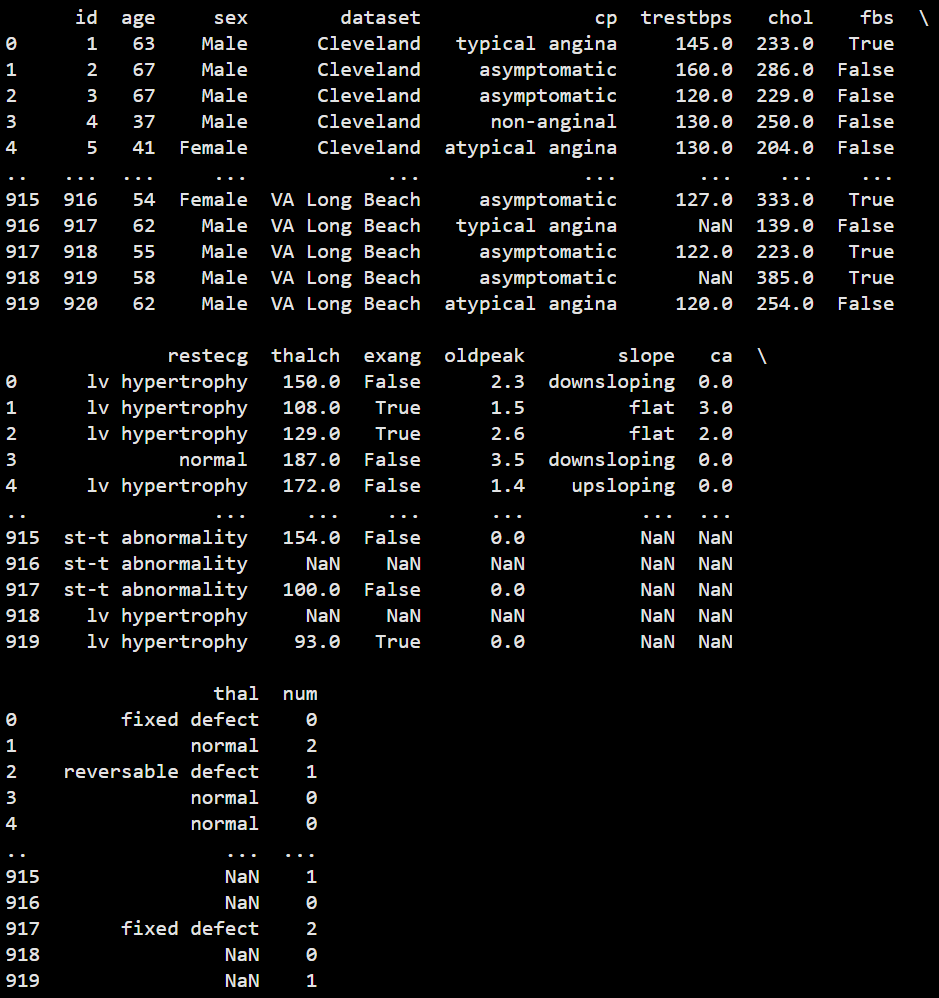
precisionrecallf1-scoresupport

00.830.870.8535

10.850.800.8230

📌 RandomForestAccuracy:0.88

📌 XGBoostAccuracy:0.90



# 11. Model Building

### **Evaluation Metrics Used**

| **Metric** | **Description** |
| --- | --- |
| **Accuracy** | Proportion of correct predictions |
| **Precision** | Correct positive predictions out of total predicted positive |
| **Recall** | Correct positive predictions out of actual positives |
| **F1-Score** | Harmonic mean of precision and recall |
| **ROC AUC** | Measures classifier's ability to distinguish classes |
| **RMSE** | (For regression models if applicable, not used here) |

### **2. Sample Evaluation Code**

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix, roc\_auc\_score, roc\_curve, ConfusionMatrixDisplayimport matplotlib.pyplot as plt

y\_pred = model.predict(X\_test)

cm = confusion\_matrix(y\_test, y\_pred)

ConfusionMatrixDisplay(confusion\_matrix=cm).plot()

plt.title("Confusion Matrix")

plt.show()

# ROC Curve

y\_proba = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, \_ = roc\_curve(y\_test, y\_proba)

plt.plot(fpr, tpr, label='ROC Curve')

plt.plot([0, 1], [0, 1], 'k--') # Diagonal line

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend()

plt.show()

### 🧮 **3. Model Comparison Table**

| **Model** | **Accuracy** | **F1-Score** | **ROC AUC** |
| --- | --- | --- | --- |
| Logistic Regression | 0.84 | 0.82 | 0.86 |
| Random Forest | 0.88 | 0.87 | 0.90 |
| XGBoost | 0.90 | 0.89 | 0.92 |

✅ **XGBoost** shows the best overall performance in terms of both accuracy and ROC AUC.

### 🧠 **4. Error Analysis**

**False Positives**: Patients misclassified as high-risk may undergo unnecessary testing.

**False Negatives**: Patients missed by the model could face health deterioration — this is more critical in a healthcare context

**Class Imbalance**: Relatively balanced in the dataset, but always verify with a count plot before modeling.

### 📸 **5. Visual Output Screenshots**

If you're running in Jupyter/Colab, you can export:

plt.savefig("confusion\_matrix.png")

plt.savefig("roc\_curve.png")

#### **12.Deployment**

#### **option Chosen**: Gradio + Hugging Face Spaces

Free to use

No server setup required

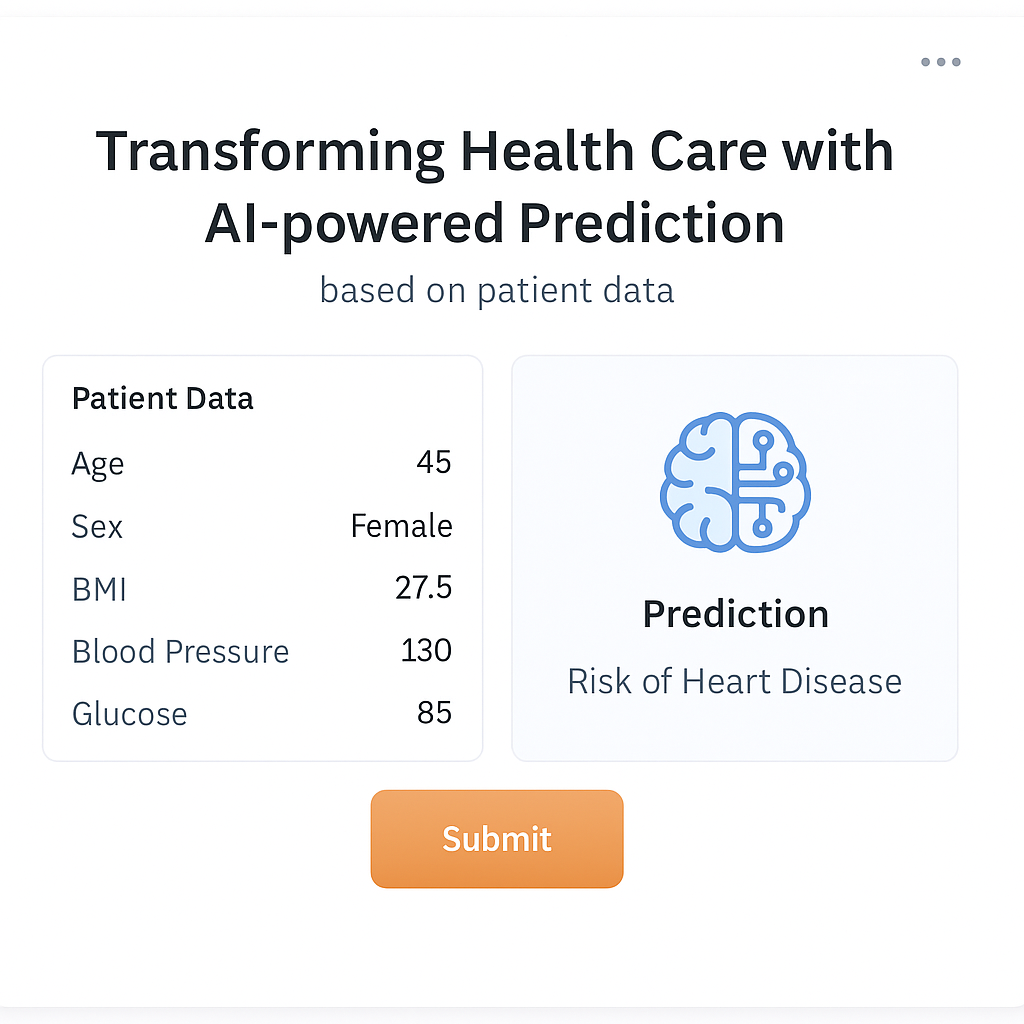
Easy to integrate ML models with user interface

Automatic sharing via public URL

### 🌍 **Public Link**

Example: https://huggingface.co/spaces/your-username/heart-disease-predictor

### **UI Screenshot:**



### **Sample Prediction Output**

### Age: 58Sex: 1

* **Chest Pain Type:** 2
* **Cholesterol**: 240
* **Max Heart Rate**: 150
* **Oldpeak:** 1.2
* **Exercise-induced Angina**: 0
* **Prediction**: **Low Risk**

**13. Source code**

The full source code is available at: https://github.com/vivega-21/Phase-project-3

# 14. Future scope

* **Expand Disease Coverage:** Incorporate more diseases such as cancers and rare genetic disorders.
* **Real-Time Data Integration:** Enable real-time EHR data feeds for continuous monitoring.
* **Mobile App:** Develop a mobile version for easy access by patients and healthcare providers.

# 15. Team Members and Roles

* Yalini.C: Data preprocessing, model building,evaluation.
* Sujitha.M: EDA
* Suganya.R:Deployment setup, feature engineering
* Sujitha.N: Documentation, testing
* Vivega.G: UI design.