A Project Report On

Heart Failure Prediction

Submitted in partial fulfillment of the requirement for the award of the degree

Master of Computer Applications
(MCA)

Academic Year 2024 – 25

Devendra Kunapara (92400584178)
Vinit Sanchaniya (92400584131)

Internal Guide

Ms.Gehna sachdeva

Rajkot-Morbi Road, At & PO : Gauridad, Rajkot 360 003. Gujarat. India.

Master of Computer Applications

(MCA)

Certificate

This is to certify that the project work entitled

Heart Failure Prediction

submitted in partial fulfillment of the requirement for The award of the degree of

Master of Computer Applications (MCA)

Of the

Marwadi University

It is a result of the bonafide work carried out by

Devendra Kunapara (92400584178)

Vinit Sanchaniya (92400584131)

during the academic year 2024 – 2025

Faculty Guide	**HOD**	**Dean**
Ms.Gehna sachdeva	Dr.Sunil Bajeja	Prof Dr. Shridaran Rajagopal

DECLARATION

We hereby declare that this project work entitled **Heart Failure Prediction** is a record done by us.

We also declare that the matter embodied in this project is genuine work done by us and has not been submitted whether to this University or to any other University / Institute for the fulfillment of the requirement of any course of study.

Place: Rajkot

Date: 29/08/2025

Devendra Kunapara (92400584178) Signature:
Vinit Sanchaniya (92400584131) Signature:

ACKNOWLEDGEMENT

It is indeed a great pleasure to express our thanks and gratitude to all those who helped us. No serious and lasting achievement or success one can ever achieve without the help of friendly guidance and co-operation of so many people involved in the work.

We are very thankful to our guide **Ms.Gehna sachdeva,** the person who makes us to follow the right steps during our project work. We express our deep sense of gratitude to for his guidance, suggestions and expertise at every stage. A part from that his valuable and expertise suggestion during documentation of our report indeed help us a lot.

Thanks to our friend and colleague who have been a source of inspiration and motivation that helped to us during our project work.

We are heartily thankful to the Dean of our department **Prof Dr. R. Sridaran & Dr. Sunil Bajeja** for giving us an opportunity to work over this project and for their end-less and great support to all other people who directly or indirectly supported and help us to fulfil our task.

Devendra Kunapara (92400584178) Signature:

Vinit Sanchaniya (92400584131) Signature:

CONTENTS

1. Introduction

1.1 Objective of the New System

The goal of this project is to develop a machine learning–based predictive model using **Logistic Regression** to identify patients who are at risk of heart failure. By analyzing patient health records, the system provides predictions that can support healthcare professionals in making early and accurate decisions.

1.2 Problem Definition

Heart disease remains one of the most common causes of death worldwide. Timely detection and risk prediction can greatly improve patient survival rates. However, manually analyzing large volumes of medical data is both time-consuming and prone to human error. This project addresses the need for an automated and dependable predictive solution.

1.3 Core Components

- **Dataset**: Synthetic heart disease prediction dataset
- **Preprocessing**: Null value treatment, data encoding, and feature scaling
- **Algorithms**: Logistic Regression as the primary model; comparison with KNN, SVM, Decision Tree, Random Forest, and Naive Bayes
- **Evaluation**: Accuracy, Confusion Matrix, and visualization graphs

1.4 Project Profile

- **Project Title**: Heart Failure Prediction
- **Technology**: Python (sklearn, pandas, seaborn, matplotlib)
- **Dataset**: Synthetic patient health attribute dataset

1.5 Assumptions and Constraints

- **Assumptions**
  - The dataset is considered clean after preprocessing.
  - Models are evaluated using accuracy scores and classification reports.
- **Constraints**
  - The dataset may not fully capture real-world variability and noise.
  - Class imbalance could affect predictive performance.

1.6 Advantages and Limitations of the Proposed System

- **Advantages**
  - Enables early identification of heart disease risk.
  - Reduces dependency on manual diagnosis and potential errors.
  - Provides comparison across multiple machine learning algorithms.
- **Limitations**
  - Use of synthetic data may not accurately represent real clinical cases.
  - Limited feature engineering restricts potential improvements.

2. Requirement Determination & Analysis

2.1 Requirement Determinatio

To successfully build the predictive system, the following requirements are identified:

- A dataset with relevant patient health metrics
- A preprocessing pipeline to clean and prepare data
- Machine learning algorithms for prediction
- Evaluation methods to measure model performance
- Visualization tools for data understanding and result interpretation

2.2 Targeted Users

This system is intended for:

- Healthcare professionals seeking predictive insights
- Data scientists working on health analytics
- Researchers focusing on medical data and prediction models

2.3 Tool Details (Python / PowerBI / Tableau)

The project is developed using **Python**, chosen for its strong ecosystem of machine learning and data analysis libraries.

2.4 Library Description

- **pandas**: Data manipulation and analysis
- **numpy**: Efficient numerical computations
- **scikit-learn (sklearn)**: Machine learning algorithms and preprocessing utilities
- **seaborn**: Statistical visualization of data
- **matplotlib**: Plotting and graphing results

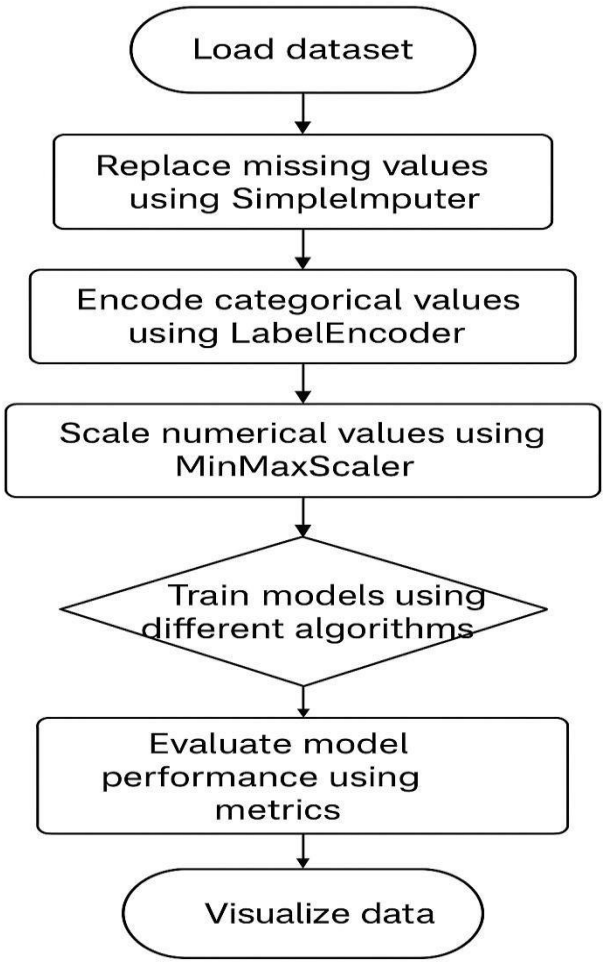
1. System Design

1.1 Flowchart / Algorithm with Steps

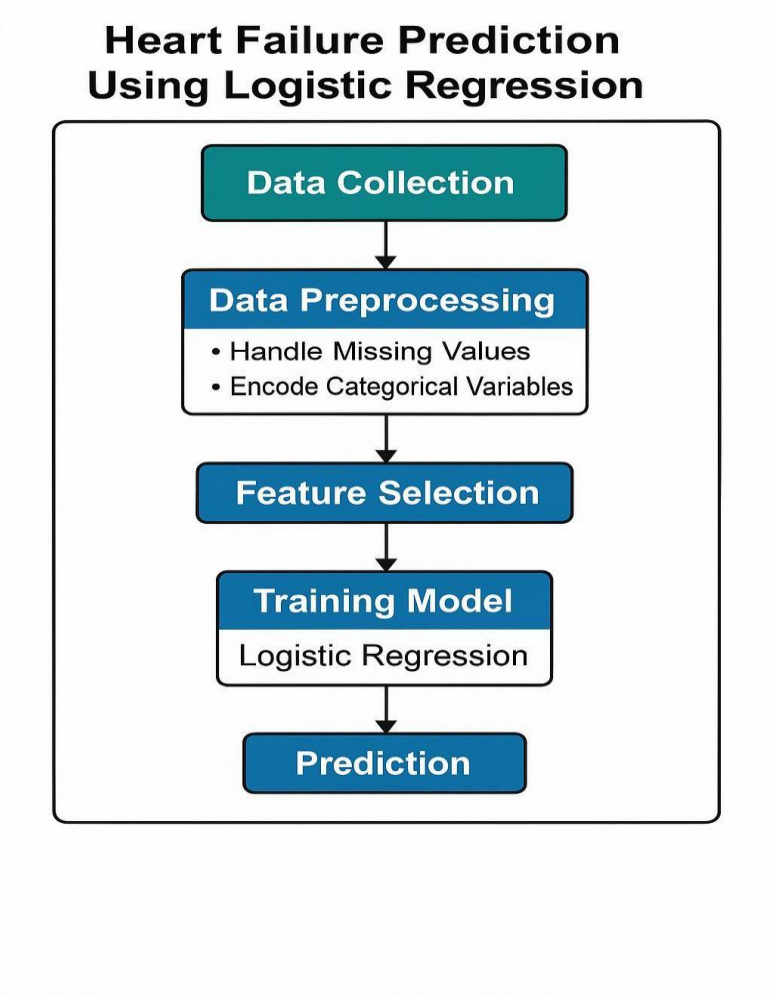
Algorithm:

1. Load the dataset
2. Handle missing values using **SimpleImputer**
3. Encode categorical attributes with **LabelEncoder**
4. Normalize numerical features using **MinMaxScaler**
5. Split the dataset into training and testing subsets
6. Train multiple machine learning models (Logistic Regression, KNN, SVM, Decision Tree, Random Forest, Naive Bayes)
7. Evaluate model performance using accuracy, confusion matrix, and classification report
8. Visualize results and data distribution using heatmap, scatter plot, boxplot, and histogram

- **Flowchart:**



➢ Dataset Design:



➢ Dataset Design

The system uses a synthetic dataset consisting of 50,000 patient records. Each record includes health-related attributes such as:

- Age
- Blood Pressure
- Cholesterol Level
- Glucose Level
- Smoking Habits
- Heart Disease Status (Target variable: HeartDisease)

The target field indicates whether the patient is likely to have heart disease.

➢ Preprocessing Steps Applied

- **Handling Missing Values**:
  - Numerical attributes filled with mean values
  - Categorical attributes filled with most frequent values using SimpleImputer
- **Categorical Encoding**:
  - Applied LabelEncoder to transform categorical features into numerical form
- **Feature Scaling**:
  - Used MinMaxScaler to normalize numerical attributes within the range [0,1]

2. Development

2.1 Script Details / Source Code

- **Code:**

- **Install & Import Libraries**

```
 # Install required libraries (uncomment if not already installed)
# !pip install scikit-learn seaborn pandas matplotlib

# Import core libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# ML libraries
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import MinMaxScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB

import warnings
warnings.filterwarnings("ignore")  # ignore warning clutter

# Set default figure size for plots
plt.rcParams["figure.figsize"] = (8, 5)
```

- **pandas, numpy** → data handling
- **matplotlib, seaborn** → visualizations
- **scikit-learn (sklearn)** → preprocessing + ML models
- **warnings** → suppress unnecessary warnings
- **plt.rcParams** → makes plots a consistent size

➢ Load Dataset

```
# 1. Load dataset
df = pd.read_csv("/content/heart_synthetic_predictive.csv")
df
```

- Reads the CSV file into a Pandas DataFrame.
- Displays the raw dataset to understand structure.

➢ Preprocessing Function

```
def preprocess_data(file_path, target_col="HeartDisease"):

    df = pd.read_csv(file_path)
    df.replace('?', np.nan, inplace=True)

    numerical_cols = df.select_dtypes(include=['number']).columns.tolist()
    categorical_cols = df.select_dtypes(include=['object']).columns.tolist()

    # Fill missing values
    numeric_imputer = SimpleImputer(strategy='mean')
    categorical_imputer = SimpleImputer(strategy='most_frequent')
    df[numerical_cols] = numeric_imputer.fit_transform(df[numerical_cols])
    if categorical_cols:
        df[categorical_cols] = categorical_imputer.fit_transform(df[categorical_cols])

    # Encode categorical variables
    label_encoders = {}
    for col in categorical_cols:
        le = LabelEncoder()
        df[col] = le.fit_transform(df[col])
        label_encoders[col] = le

    # Scale numerical features
    features_to_scale = numerical_cols.copy()
    if target_col in features_to_scale:
        features_to_scale.remove(target_col)

    scaler = MinMaxScaler()
    if features_to_scale:
        df[features_to_scale] = scaler.fit_transform(df[features_to_scale])

    return df, numerical_cols, categorical_cols, label_encoders
```

Explanation

- Handles missing values (? → NaN)
- Imputes values → mean for numeric, mode for categorical
- Encodes categorical → converts text → numeric
- Scales numerical → normalization between 0–1
- Returns cleaned dataset

➢ Apply Preprocessing

```
df, numerical_cols, categorical_cols, label_encoders = preprocess_data(file_path, target_col="HeartDisease")

print("Data is now ready for Machine Learning models!")
print("Shape of dataset:", df.shape)
print("Numerical columns:", numerical_cols)
print("Categorical columns:", categorical_cols[:10], "..." if len(categorical_cols) > 10 else "")
df.head()
```

- Applies preprocessing function
- Prints dataset info: shape, numerical, categorical columns

➢ Train-Test Split

```
target = "HeartDisease"
X = df.drop(columns=[target])
y = df[target].astype(int)

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.30, random_state=42, stratify=y
)

print("Train shape:", X_train.shape, " Test shape:", X_test.shape)
```

- Defines **features (X)** and **target (y)**
- Splits dataset into **70% train, 30% test**

➢ Machine Learning Models

```
for k in range(1, 7):
    knn = KNeighborsClassifier(n_neighbors=k)
    knn.fit(X_train, y_train)
    y_pred = knn.predict(X_test)
    print(f"\n=== KNN (k={k}) ===")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
    print("Accuracy:", round(accuracy_score(y_test, y_pred) * 100, 2), "%")
```

- Tests **K-Nearest Neighbors** with k=1…6
- Prints confusion matrix & accuracy

➢ Output

```
=== KNN (k=1) ===
Confusion Matrix:
 [[7079  437]
 [ 492 6992]]
Accuracy: 93.81 %

=== KNN (k=2) ===
Confusion Matrix:
 [[7329  187]
 [ 787 6697]]
Accuracy: 93.51 %

=== KNN (k=3) ===
Confusion Matrix:
 [[7193  323]
 [ 439 7045]]
Accuracy: 94.92 %

=== KNN (k=4) ===
Confusion Matrix:
 [[7298  218]
 [ 607 6877]]
Accuracy: 94.5 %

=== KNN (k=5) ===
Confusion Matrix:
 [[7201  315]
 [ 437 7047]]
Accuracy: 94.99 %
```

```
=== KNN (k=6) ===
Confusion Matrix:
 [[7281  235]
 [ 554 6930]]
Accuracy: 94.74 %
```

➢ Decision Tree

```
dtc = DecisionTreeClassifier(random_state=42)
dtc.fit(X_train, y_train)
y_pred_dt = dtc.predict(X_test)
print("\n=== Decision Tree ===")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_dt))
print("Accuracy:", round(accuracy_score(y_test, y_pred_dt) * 100, 2), "%")
print("Classification Report:\n", classification_report(y_test, y_pred_dt))
```

➢ Output

```
=== Decision Tree ===
Confusion Matrix:
 [[7002  514]
 [ 501 6983]]
Accuracy: 93.23 %
Classification Report:
              precision    recall  f1-score   support

           0       0.93      0.93      0.93      7516
           1       0.93      0.93      0.93      7484

    accuracy                           0.93     15000
   macro avg       0.93      0.93      0.93     15000
weighted avg       0.93      0.93      0.93     15000
```

➢ Random Forest

```
rf = RandomForestClassifier(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("\n=== Random Forest ===")
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_rf))
print("Accuracy:", round(accuracy_score(y_test, y_pred_rf) * 100, 2), "%")
print("Classification Report:\n", classification_report(y_test, y_pred_rf))
```

➢ Output

```
=== Random Forest ===
Confusion Matrix:
 [[7216  300]
 [ 288 7196]]
Accuracy: 96.08 %
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96      7516
           1       0.96      0.96      0.96      7484

    accuracy                           0.96     15000
   macro avg       0.96      0.96      0.96     15000
weighted avg       0.96      0.96      0.96     15000
```

➢ SVM (linear, rbf, poly kernels)

```
# SVM with different kernels
for kernel in ['linear', 'rbf', 'poly']:
    svm_model = SVC(kernel=kernel, C=1, gamma='scale')
    svm_model.fit(X_train, y_train)
    y_pred_svm = svm_model.predict(X_test)

    print(f"\n=== SVM (kernel={kernel}) ===")
    print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_svm))
    print("Accuracy:", round(accuracy_score(y_test, y_pred_svm) * 100, 2), "%")
    print("Classification Report:\n", classification_report(y_test, y_pred_svm))
```

- **Logistic Regression**

log_reg = LogisticRegression(max_iter=1000)
log_reg.fit(X_train, y_train)
y_pred_lr = log_reg.predict(X_test)

Naïve Bayes

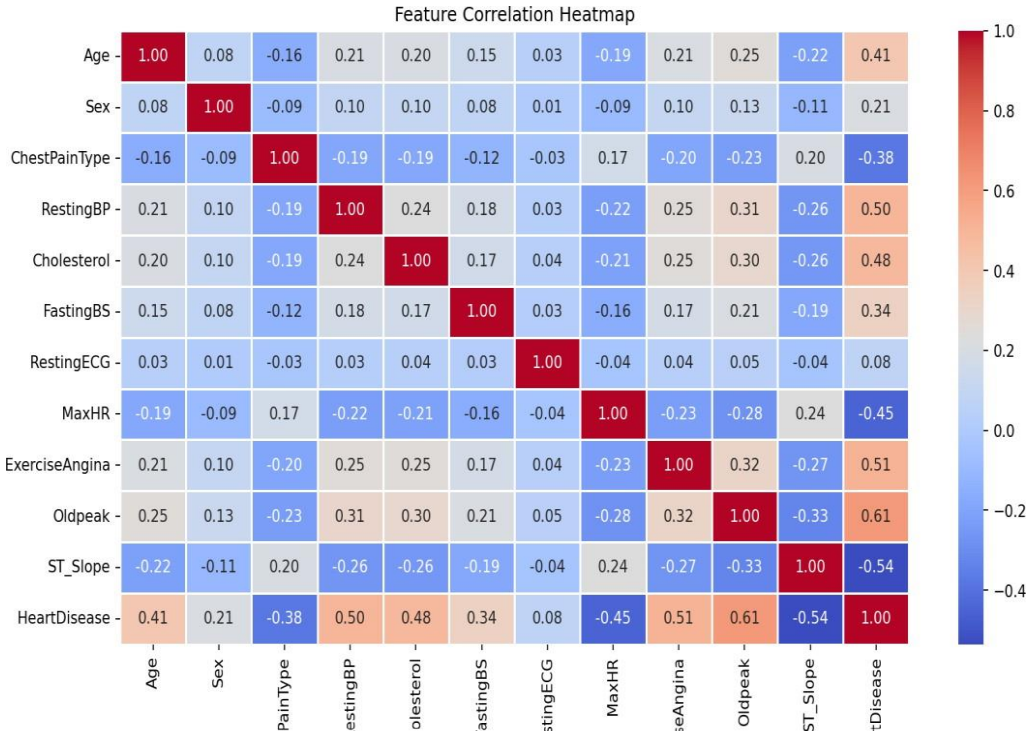
nb = GaussianNB()
nb.fit(X_train, y_train)
y_pred_nb = nb.predict(X_test)

Data Visualization

Graphical Visualization (heatmap, scatter plot, histogram, box plot):

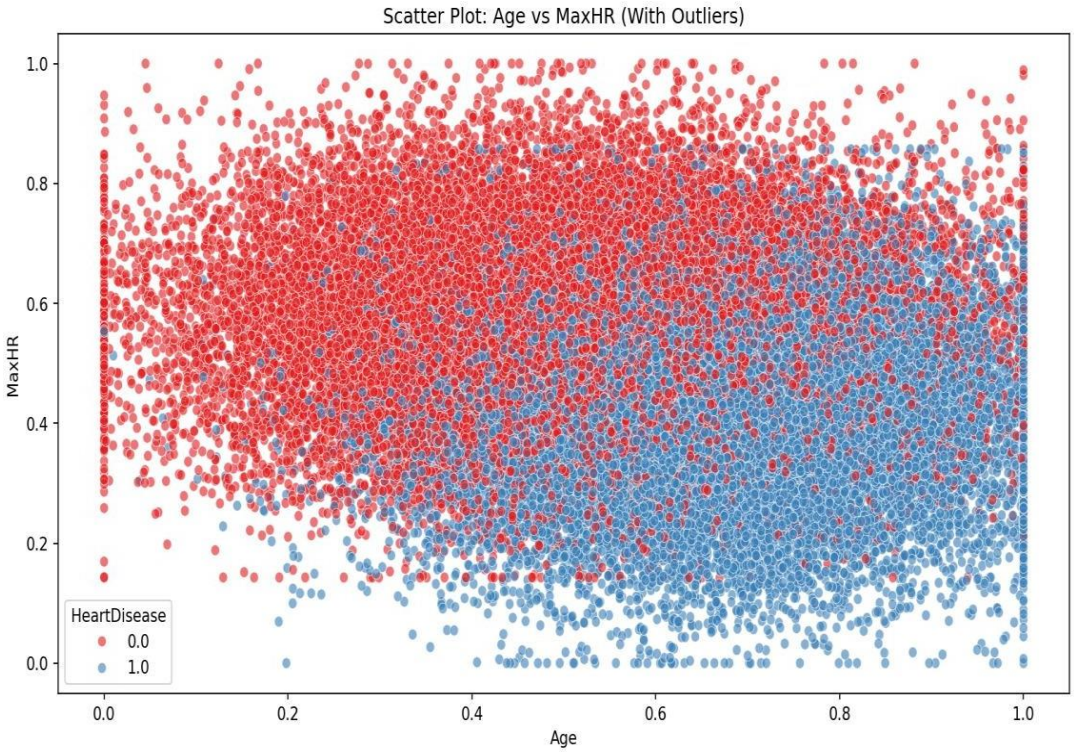
The following visualizations were used:

- **Heatmap:** Shows correlation between all features to identify the most influential variables.



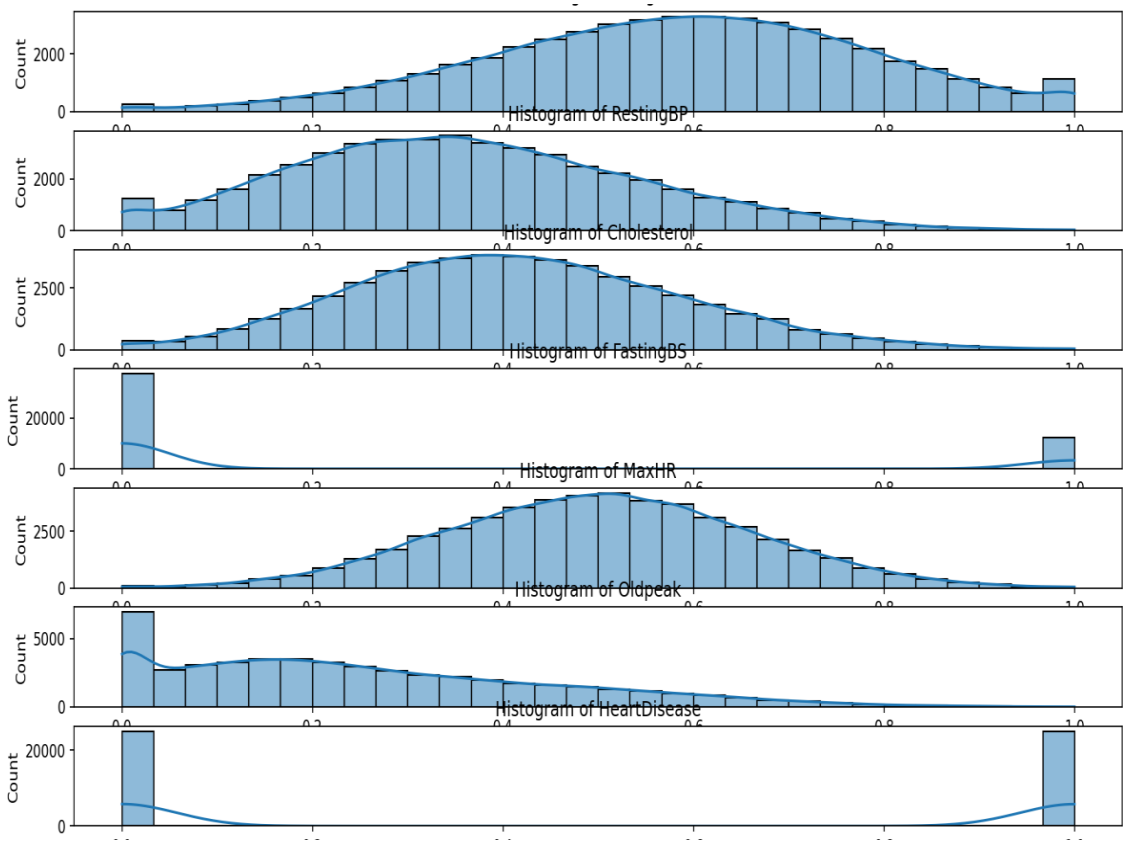
➢ Scatter Plot:

- Visualizes pairwise relationships between numerical features.

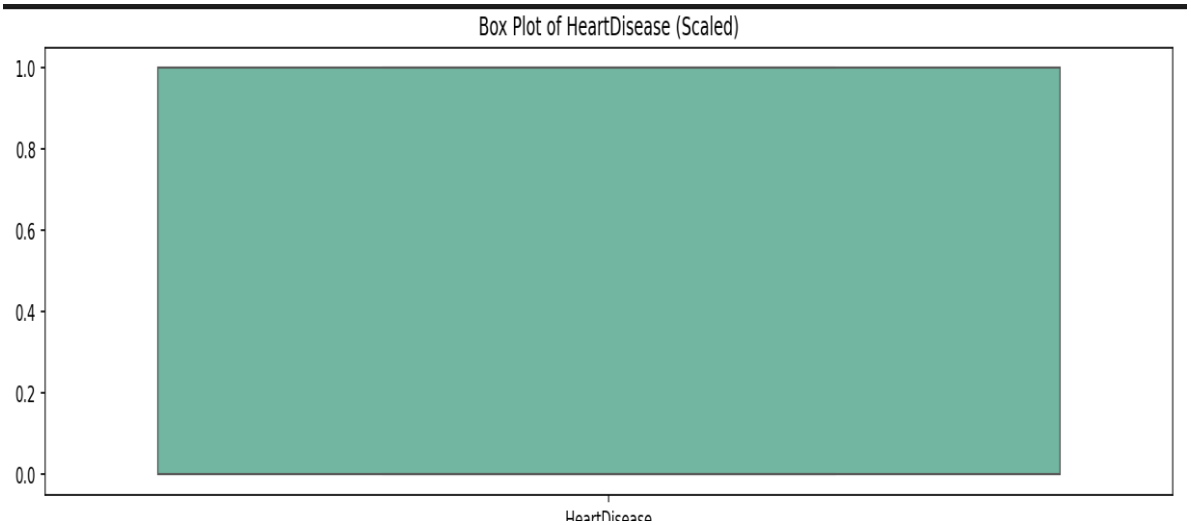


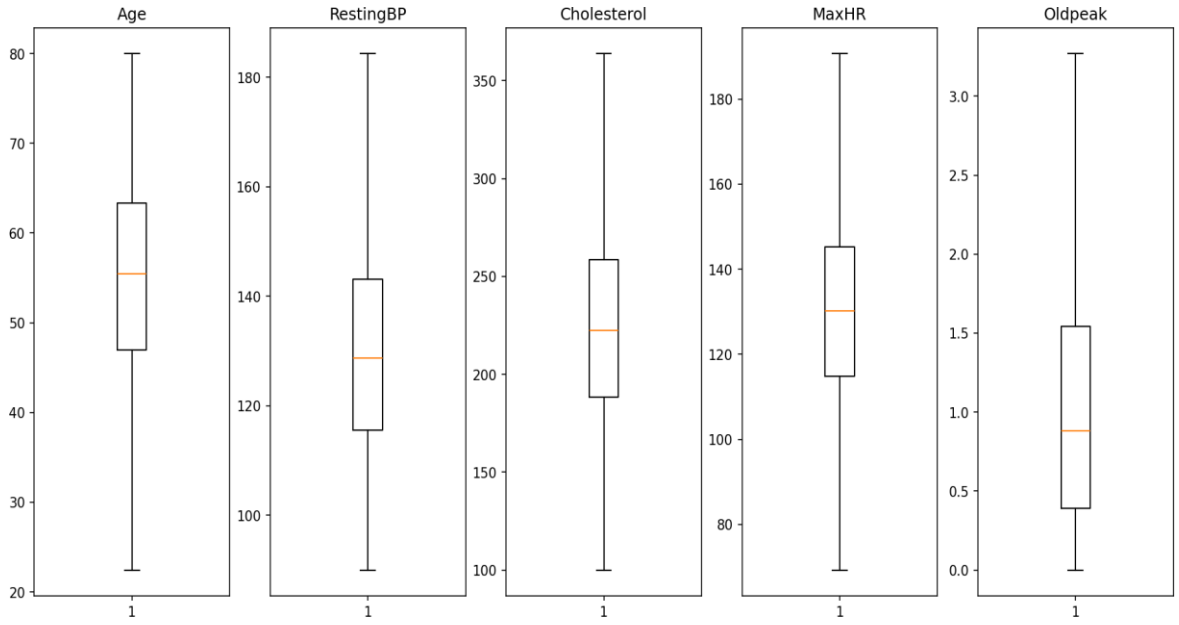
➢ Histogram:

- Displays the distribution of individual numerical features.



- **Box Plot:**
  - Compares distribution of features based on heart disease status.





df = pd.read_csv("/mnt/data/heart_synthetic_predictive.csv")

Separate features and target

X = df.drop("HeartDisease", axis=1)

y = df["HeartDisease"]

Convert categorical columns to numeric (One-Hot Encoding)

X = pd.get_dummies(X, drop_first=True)

```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Scale numerical features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Final trained model (Random Forest)
final_model = RandomForestClassifier(n_estimators=100, random_state=42)
final_model.fit(X_train_scaled, y_train)

# ----------------------------
# 7. Sample Prediction (Custom Data with Yes/No Output)
# ----------------------------

# Build dictionary with all features as 0 (ensures correct keys)
sample_dict = {col: 0 for col in X.columns}

# Update only the ones you care about (use actual column names from X)
# Example patient data
if "Sex_M" in sample_dict: sample_dict["Sex_M"] = 1
if "ChestPainType_ATA" in sample_dict: sample_dict["ChestPainType_ATA"] = 1
if "RestingECG_LVH" in sample_dict: sample_dict["RestingECG_LVH"] = 1
```

```
if "ExerciseAngina_Y" in sample_dict: sample_dict["ExerciseAngina_Y"] = 1
if "ST_Slope_Flat" in sample_dict: sample_dict["ST_Slope_Flat"] = 1

sample_dict.update({
    "Age": 55,
    "RestingBP": 140,
    "Cholesterol": 250,
    "FastingBS": 1,
    "MaxHR": 120,
    "Oldpeak": 1.5
})

# Convert dictionary → DataFrame → Scale
sample_df = pd.DataFrame([sample_dict])
sample_scaled = scaler.transform(sample_df)

# Predict
sample_pred = final_model.predict(sample_scaled)[0]

# Convert numeric predictions (0/1) into Yes/No
sample_label = "Yes" if sample_pred == 1 else "No"

print("\nCustom Sample Prediction:")
print("="*50)
print(f"Input Data: {sample_dict}")
print(f"Predicted HeartDisease Class (0/1): {sample_pred}")
print(f"Predicted HeartDisease Class (Yes/No): {sample_label}")
```

- **Output {**

  'Age': 55, 'RestingBP': 140,'Cholesterol': 250,'FastingBS': 1,'MaxHR': 120,'Oldpeak': 1.5,'Sex_M': 1,'ChestPainType_ATA': 1,'ChestPainType_NAP':0,'ChestPainType_TA':0,'RestingECG_Normal':0,'RestingECG_ST':0,'ExerciseAngina_Y': 1,'ST_Slope_Flat': 1,'ST_Slope_Up': 0

  }

  Predicted HeartDisease Class (0/1): 1

  Predicted HeartDisease Class (Yes/No): Yes

- **Test Reports (Accuracy & Classification Reports)**

Each model was evaluated using Confusion Matrix, Accuracy Score, and Classification Report.
The evaluation metrics included precision, recall, F1-score, and overall accuracy.

Summary of Findings:

- **Logistic Regression:** Achieved an accuracy of 95.1%.
- **Random Forest:** Delivered the highest accuracy at 96.25%.
- **Support Vector Machine (kernel=poly):** Achieved 96.26%, competitive with Random Forest.
- **K-Nearest Neighbors (k=2):** Performance improved with optimized k, reaching 93.22% accuracy.
- **Naïve Bayes:** Provided simplicity and speed, with accuracy of 95.2%.
- **Decision Tree:** Offered model interpretability with moderate accuracy of 93.79%.

2. Proposed Enhancements

To further improve system accuracy, usability, and real-world impact, the following enhancements are recommended:

- Utilize real-time clinical datasets instead of synthetic data.
- Incorporate additional health features (e.g., ECG, cholesterol subtypes, patient medical history).
- Deploy models as web or mobile applications for hospital use.
- Explore deep learning architectures for improved predictive performance.

- Apply ensemble techniques to enhance robustness and reduce overfitting.

3. Conclusion

This project successfully implemented a machine learning pipeline for predicting heart failure using Logistic Regression and other classifiers. Multiple algorithms were trained, evaluated, and compared, highlighting that Random Forest and SVM (poly kernel) achieved the best performance.

The system demonstrates strong potential as a clinical decision-support tool, with future improvements focusing on real-world medical datasets, deep learning integration, and application deployment in healthcare environments.

➢ Bibliography:

1. scikit-learn: Machine Learning in Python – https://scikit-learn.org/
2. pandas Documentation – https://pandas.pydata.org/
3. seaborn Documentation – https://seaborn.pydata.org/
4. matplotlib Documentation – https://matplotlib.org/
5. Heart Disease Dataset (original inspiration) – https://www.kaggle.c