Name: UWIMANA Diane

Regno*:* 25RP21687
Course: ITLML801: Machine Learning
Level: 8, Year 4 IT

README

Heart Disease Risk Prediction System

CHUB - Clinical Decision Support System

Project Overview

This project implements a machine learning-based clinical decision support system for predicting heart disease risk levels in patients.

Features

- 5-class heart disease classification (No Disease, Very Mild, Mild, Severe, Immediate Danger)

- Multiple ML models with hyperparameter tuning

- Flask REST API for model serving

- Responsive web interface for clinical use

PROJECT REPORT

Heart Disease Risk Prediction System with Flask Deployment

Introduction

Cardiovascular diseases are among the leading causes of mortality worldwide. Early and accurate risk assessment is critical in reducing complications and improving patient outcomes. This project presents an end-to-end **Heart Disease Risk Prediction System** developed for CHUB (a referral and teaching hospital). The system leverages machine learning techniques to classify patients into five heart disease risk categories and deploys the best-performing model as a Flask-based web application for clinical decision support.

Objectives

The main objectives of this project were:

- To load, explore, and analyze a structured heart disease dataset of 5000 patients.

- To perform full Exploratory Data Analysis (EDA) including statistics and visualizations.
- To design robust preprocessing pipelines for numerical and categorical data.
- To train, tune, and compare multiple classification models.
- To evaluate the best-performing model in detail.
- To persist the trained model and deploy it using a Flask REST API.
- To build a responsive frontend interface for medical staff.

Dataset Description

The dataset consists of **5000 patient records**, each described by **13 clinical and demographic features**, including age, blood pressure, cholesterol level, chest pain type, ECG results, exercise-induced angina, number of affected vessels, and thalassemia status. The target variable represents heart disease risk level with five classes:

- 0: No Disease
- 1: Very Mild
- 2: Mild
- 3: Severe
- 4: Immediate Danger

Data Loading and Inspection

The dataset was loaded using the Pandas library. Initial inspection included:

- Displaying the total number of samples and features.
- Viewing the first five rows of the dataset.
- Checking for missing values across all features.

These steps ensured data integrity before further analysis.

Exploratory Data Analysis (EDA)

EDA was performed to understand the structure and distribution of the data:

5.1 Statistical Analysis

- Dataset shape (instances and features).
- Data types of all features.
- Descriptive statistics (mean, median, standard deviation) for numerical features.

- Class distribution counts and percentages.

Class Balance Analysis

The percentage contribution of each heart disease class was calculated. By comparing the largest and smallest class sizes, the imbalance ratio was analyzed to determine whether the dataset was balanced or imbalanced.

Visualizations

The following visualizations were generated:

- Bar plot showing distribution of the five heart disease classes.
- Correlation heatmap for numerical features.
- Box plots comparing age across heart disease classes.
- Box plots comparing cholesterol levels across classes.
- Horizontal bar chart showing percentage of missing values per feature.

These visualizations provided insights into relationships between features and disease severity.

Data Preprocessing

Train-Test Split

The dataset was split into training (80%) and testing (20%) sets using **stratified sampling** to preserve class balance.

Feature Identification

- Numerical features were identified and counted.
- Categorical features were identified and counted.

Preprocessing Pipelines

- Numerical pipeline: Missing value imputation (median) and feature scaling using StandardScaler.
- Categorical pipeline: Missing value imputation (most frequent) and OneHotEncoding with unknown category handling.

The pipelines were combined using a **ColumnTransformer**, ensuring consistent preprocessing for both training and testing data. Post-transformation checks confirmed:

- No missing values remained.
- All transformed features were numeric.

Model Training and Evaluation

Models Trained

The following models were trained and tuned using GridSearchCV:

- Multi-Layer Perceptron (MLP / ANN)
- Random Forest Classifier
- Support Vector Machine (SVM)
- K-Nearest Neighbors (KNN)
- Gradient Boosting Classifier

Hyperparameter Tuning

Each model was wrapped in a full pipeline (preprocessing + classifier) and optimized using GridSearchCV. For each model, the following were recorded:

- Best hyperparameters
- Best cross-validation accuracy
- Training time
- Training accuracy
- Testing accuracy

Model Comparison

A comparison table was created with the following columns:

- Model
- Best CV Accuracy
- Train Accuracy
- Test Accuracy
- Overfitting Gap
- Status (Overfit / Underfit / Best Fit)

Models were sorted by test accuracy to identify the best-performing model.

Best Model Selection and Detailed Evaluation

The **Random Forest Classifier** achieved the best balance between performance and generalization and was selected as the final model.

Classification Report

A full classification report was generated, showing precision, recall, and F1-score for all five classes.

Confusion Matrix

A confusion matrix was computed and visualized to analyze misclassifications between heart disease categories.

Clinical Interpretation

- The class with the highest precision indicates reliable identification of that risk level.
- The class with the lowest recall highlights cases that may require improved sensitivity.
- This analysis helps clinicians understand where the model performs strongly and where caution is needed.

Feature Importance

Since Random Forest supports feature importance, the most influential features were extracted and plotted, providing insights into key clinical factors contributing to heart disease risk.

Model Persistence

The final model and related artifacts were saved in the deployment/ directory:

- model.pkl: Full preprocessing and classification pipeline.
- feature_columns.txt: Ordered list of input features.
- class_names.txt: Ordered list of class labels.

Two verification steps were performed:

- Reloading the model and comparing actual vs. predicted values.
- Predicting outcomes for new patient samples and displaying probabilities.

Flask API Deployment

A Flask REST API was developed with the following endpoints:

- Displays model and API status.
- Api/predict Accepts patient data in JSON format and returns predicted class, confidence score, and per-class probabilities.
- The API includes input validation and error handling to ensure robustness.

Frontend Interface

- A responsive single-page HTML interface was created using HTML, CSS, and Bootstrap. The frontend allows medical staff to:
- Enter patient clinical data.
- Submit data to the Flask API.

View predicted heart disease risk level with color-coded severity.

View probability distribution across all five classes.

The interface was tested across desktop, tablet, and mobile devices.

Testing and Validation

Three patient cases were tested to verify consistency between:

Backend model predictions.

Flask API responses.

Frontend displayed results.

Identical outputs confirmed correct integration of the system.

Conclusion

This project successfully delivered a complete end-to-end Heart Disease Risk Prediction System. The integration of machine learning, robust preprocessing, model evaluation, and web deployment demonstrates how data-driven tools can support clinical decision-making. The system is reliable, interpretable, and ready for real-world deployment within a hospital environment.