Heart Disease Detection Project

Haidy Mohamed 2305309

Hagar Amr 2305197

Arwa Ayman 230137

Overview

1. Project Description The Heart Disease Detection Project is designed to analyze patient health indicators to predict the risk of heart disease using both a rule-based expert system (Experta) and a machine learning model (Decision Tree Classifier in Scikit-Learn). The project encompasses data preprocessing, visualization, and a structured folder system to enhance clarity and usability.

2. Requirements & Implementation Steps

Step 1: Dataset Processing

- Load the dataset using Pandas.
- Handle missing values by filling them with mean/median or dropping incomplete rows.
- Normalize numerical features (e.g., blood pressure, cholesterol) using MinMaxScaler.
- Encode categorical variables using One-Hot Encoding.
- Select the most important features using correlation analysis.
- Save the cleaned dataset as `cleaned_data.csv`.

Step 2: Data Visualization

- Generate a statistical summary and visualize distributions using Pandas and Seaborn.
- Create a correlation heatmap to analyze feature relationships.
- Use histograms and boxplots to identify data distribution and outliers.
- Develop a feature importance plot to rank attributes based on their impact on heart disease prediction.

Step 3: Implement Rule-Based Expert System (Experta)

- Define at least 10 rules for heart disease risk assessment (e.g., high cholesterol and age > 50 indicate high risk).
- Develop a knowledge base using Experta.
- Implement an inference mechanism to assess risk based on user input.
- Enable user input functionality for symptom-based risk prediction.

Step 4: Build Decision Tree Model (Scikit-Learn)

- Split data into an 80/20 train-test set.
- Train a Decision Tree Classifier on the dataset.
- Perform hyperparameter tuning for optimal performance.
- Evaluate the model using accuracy, precision, recall, and F1-score.
- Save the trained model using joblib.

Step 5: Compare Expert System and Decision Tree Model

- Validate both models using an unseen dataset.
- Compare accuracy and other performance metrics.
- Analyze the interpretability of decision trees versus human-defined rules.

Step 6: Integration & GitHub Upload

- Organize the codebase into logical folders.
- Document setup instructions and usage examples.
- Upload the complete project to GitHub, including a well-structured README file.

3. Deliverables

- Cleaned and preprocessed heart disease dataset.
- Data visualization notebook with insights.
- Rule-based expert system implemented with Experta.
- Decision Tree model with hyperparameter tuning.
- Accuracy comparison report.
- Well-structured codebase.
- Comprehensive project documentation.

Your `data_processing.py` script effectively handles dataset loading, missing value imputation, normalization, categorical encoding, feature selection, and saving the cleaned data. Here are a few suggestions for improvement:

1. **Error Handling**: Add try-except blocks around critical operations (e.g., file loading, transformations) to catch potential errors.
2. **Logging**: Instead of just printing messages, consider using the `logging` module for better debugging and tracking.
3. **Parameterization**: Allow users to specify input/output file paths via command-line arguments or function parameters.
4. **Feature Selection Enhancement**: Instead of just plotting the heatmap, consider selecting features based on a correlation threshold to improve model performance.

This script is a **data preprocessing and analysis pipeline** for a heart disease dataset. It includes the following key functionalities:

1. **Dataset Loading**
    a. Checks if the dataset file exists before loading it.
2. **Missing Value Handling**
    a. Fills numerical columns with their median values.
    b. Fills categorical columns with their most frequent values (mode).
3. **Data Normalization**
    a. Uses `MinMaxScaler` to scale numerical features.
4. **Categorical Encoding**
    a. Applies One-Hot Encoding to categorical features.
5. **Feature Selection**
    a. Selects the top correlated features with the target variable.
6. **Data Visualization**
    a. Generates correlation heatmaps, histograms, and boxplots for better data understanding.
7. **Feature Importance Analysis**

    a. Uses `RandomForestClassifier` to determine the most important features.
8. **Data Saving**
    a. Saves the processed dataset to a CSV file.
9. **Main Execution Flow**
    a. Performs the preprocessing steps, selects important features, visualizes the data, and analyzes feature importance.

Enhancements Over a Basic Preprocessing Script

More Robust Missing Value Handling (Different strategies for numerical & categorical data)
Feature Importance with Machine Learning (Using Random Forest)
Extensive Data Visualization (Multiple plots for better data exploration)
Feature Selection Based on Correlation (Reduces dimensionality for better model performance)

This script is part of a **Heart Disease Expert System** built using **Experta**. It allows users to input their health data and assesses their heart disease risk using predefined rules.

Key Functionalities:

1. **User Input Handling**
    a. Asks the user for **cholesterol, blood pressure, smoking habits, age, diabetes, BMI, exercise, family history, and stress levels**.
    b. Implements **validation** to ensure proper data entry (e.g., no negative numbers, valid categorical inputs).
2. **Expert System Execution**
    a. Creates an instance of the **HeartDiseaseExpert** system.
    b. Gathers user data into a **Patient** object and converts it into a dictionary.

c. Declares the facts to the **Expert System (Experta Engine)** and runs inference.

3. **Final Risk Assessment**
   a. The expert system evaluates the data and **determines the heart disease risk level** based on predefined rules.

Potential Enhancements:

Logging Implementation – Replace print statements with Python's `logging` module for better debugging.
GUI or Web Interface – Integrate this with **Streamlit** for a more user-friendly experience.
Dynamic Rule Expansion – Allow rules to be updated dynamically instead of hardcoding them.

his script is responsible for **training, optimizing, and evaluating** a **Decision Tree Classifier** for heart disease prediction. It follows a structured machine learning pipeline:

Key Steps:

1. **Load the Cleaned Dataset**
   a. Reads the preprocessed dataset (`cleaned_data.csv`) into a Pandas DataFrame.
   b. Displays dataset shape to confirm successful loading.
2. **Data Splitting**
   a. Splits the data into **80% training** and **20% testing** sets.
   b. Saves the split datasets (`X_train.csv`, `X_test.csv`, `y_train.csv`, `y_test.csv`) for future use.
3. **Initial Decision Tree Training**
   a. Trains a **basic Decision Tree model** without hyperparameter tuning.

4. **Hyperparameter Tuning with GridSearchCV**
    a. Optimizes **max_depth, min_samples_split, and min_samples_leaf** to prevent overfitting.
    b. Uses **5-fold cross-validation** to find the best model configuration.
5. **Model Evaluation**
    a. Predicts outcomes on the test set.
    b. Calculates **accuracy, precision, recall, F1-score**, and prints a **classification report**.
6. **Model Saving**
    a. Saves the trained model as `decision_tree_model.pkl` using **joblib** for easy reuse.

This script performs an **evaluation and explainability analysis** for two models:

1. **Decision Tree Classifier (ML-based approach)**
2. **Expert System (Rule-based approach with Experta)**

It aims to **compare the performance, interpretability, and rule patterns** of both models in heart disease risk prediction.

Key Functionalities:

1. **Load and Evaluate the Decision Tree Model**
   Loads the trained model (`decision_tree_model.pkl`).
   Loads test data (`X_test.csv`, `y_test.csv`).
   Predicts heart disease risk and evaluates **accuracy, precision, recall, and F1-score**.
2. **Load and Evaluate the Expert System**
   Ensures required features exist.

Runs predictions using the **HeartDiseaseExpert** system.
Maps textual risk levels (**Low, Medium, High**) to numeric values.
Evaluates the expert system using standard metrics.

3. **Explainability & Interpretability Analysis**
   Decision Tree Visualization – Generates and saves a decision tree plot.
   Feature Importance Analysis – Extracts and plots feature importances.
   Decision Rules Extraction – Saves the tree's textual representation.
4. **Expert System Rule Analysis**
   Extracts **patterns from predictions** (e.g., age, cholesterol, smoking).
   Analyzes the most common features influencing risk levels.
   Identifies **potential rule patterns** based on feature averages.
5. **Compare Model Disagreements**
   Identifies **cases where the Decision Tree and Expert System disagree**.
   Saves the mismatches with the exact **feature values and predictions**.
6. **Generate a Comprehensive Explainability Report**
   Comparison of decision-making approaches
   Strengths & weaknesses of each model
   Suggestions for a hybrid approach combining expert rules with ML insights
   Concludes which model performs better and why

Potential Enhancements

Add logging for structured debugging instead of print statements.
Enhance hybrid modeling by integrating expert rules as additional ML features.
Deploy findings via a **Streamlit dashboard** for interactive visualization.

This script defines an **expert system for heart disease risk assessment** using the **Experta** library. It applies a set of **rules based on medical factors** to classify patients into **Low, Medium, or High risk** categories.

Key Functionalities:

1. Defining Patient Facts

- **The `Patient` class** represents a patient's health profile, including:
  - **Cholesterol, Blood Pressure, Smoking, Age, Diabetes, BMI, Exercise, Family History, Stress**
  - A **default risk level** of `"Low"`

2. Rule-Based Expert System (`HeartDiseaseExpert`)

- The system **evaluates risk** based on predefined **medical conditions and thresholds**.
- **18 Rules** categorize patients into **High, Medium, or Low risk**, such as:
  - **High Risk:**
    - **Cholesterol > 240 & Age > 50**
    - **Blood Pressure > 140 & Smoking = "Yes"**
    - **Diabetes = "Yes" & Family History = "Yes"**
    - **Age > 70 & Diabetes = "Yes" & Family History = "Yes"**
  - **Medium Risk:**
    - **Cholesterol between 200-240 & Age > 40**
    - **Blood Pressure between 120-140 & Stress = "High"**

- **Low Risk:**
    - **Exercise = "Regular" & BMI < 25**
    - **Family History = "No" & Cholesterol < 200**

3. Predicting Risk for Patients (`predict` function)

- **Processes test data (`X_test`)** and applies rules.
- **Converts numeric data into categorical values** (e.g., `"Yes"` for smoking if `exang_1 == 1`).
- **Runs the inference engine** to determine risk.
- **Assigns the highest risk level found** (High > Medium > Low).
- **Returns a list of predictions** (`["Low", "High", "Medium", ...]`).

Potential Enhancements

Logging for Debugging – Use `logging` instead of print statements for better tracking.
Dynamic Rule Expansion – Allow rule modifications through an external file instead of hardcoding them.
Hybrid Approach – Integrate **machine learning insights** to improve expert system rules.

This script performs **data loading, cleaning, and visualization** for heart disease analysis. It prepares the dataset for further modeling by handling **missing values, duplicates, and generating insights**.

Key Functionalities:

1. Load and Inspect the Dataset

Reads the dataset from `"../data/heart.csv"`.
Displays the first few rows (`df.head()`).
Provides a **summary of dataset structure** (`df.info()`) and **statistical overview** (`df.describe()`).

2. Data Cleaning

Handles missing values (`df.dropna(inplace=True)`).
Removes duplicate records (`df.drop_duplicates(inplace=True)`).
Verifies data integrity after cleaning.

3. Data Visualization

Age Distribution – Histogram with KDE to analyze patient age spread.
Age vs. Heart Disease – Boxplot to compare age distribution between patients with and without heart disease.
Feature Correlation Heatmap – Shows relationships between variables to identify strong predictors.

4. Save Cleaned Dataset

Stores the cleaned dataset as `"../data/cleaned_data.csv"`.
Displays a success message (`"✅ Cleaned dataset has been saved successfully!"`).

Potential Enhancements

Logging Implementation – Track each cleaning step instead of using print statements.
Automated Outlier Detection – Identify and handle extreme values using IQR or Z-score methods.

More Feature Engineering – Create new insights from existing data (e.g., BMI categories, risk scores)

This script performs **data preprocessing, model training, evaluation, and saving** for a **Decision Tree Classifier** in predicting heart disease.

Key Functionalities:

1. Load and Verify Data

Reads the **cleaned dataset** from `"../data/cleaned_data.csv"`.
Checks for **missing values** and fills them with the median.
Ensures the **target column ("target")** exists to avoid errors.

2. Data Preprocessing

Separates features (X) and target (y).
Encodes categorical variables using `pd.get_dummies()`.
Scales numerical features using `MinMaxScaler` for consistency.
Splits the dataset into **80% training and 20% testing**.

3. Train a Decision Tree Classifier

Uses `max_depth=5` and `min_samples_split=4` to prevent overfitting.
Fits the model on **training data**.

4. Model Evaluation

Accuracy Calculation – Measures overall correctness.
Classification Report – Displays **precision, recall, and F1-score**.
Confusion Matrix – Visualizes model performance in predicting heart disease.

5. Save the Trained Model

Exports the trained **Decision Tree model** as `"heart_disease_model.pkl"`.

This script is a **Streamlit-based web application** for **heart disease risk prediction** using:

1. **Expert System (Rule-Based)** – Uses predefined medical rules.
2. **Decision Tree Model (Machine Learning)** – Predicts risk based on trained data.

It also provides **interactive visualizations** and a **comparative analysis of risk factors**.

key Functionalities:

1. Model and Data Loading

Decision Tree Model (`decision_tree_model.pkl`) is loaded if available.
Historical Data (`cleaned_data.csv`) is used for statistical comparisons.**2. User Input For** Users enter health details like **cholesterol, blood pressure, smoking status, age, diabetes, BMI, exercise, family history, and stress**.

3. Dual Risk Prediction

Expert System: Evaluates **predefined medical rules** to determine risk.
Decision Tree Model: Uses a **trained ML model** to predict risk.

4. Results Display

Personalized Risk Assessment

- **High Risk** → Warning with medical advice.
- **Medium Risk** → Lifestyle improvement suggestions.
- **Low Risk** → Encouragement for a healthy lifestyle.

Comparison Between Expert System & Machine Learning Mode5. Data Visualization Dashboard

Risk Distribution (Bar Chart & Pie Chart) – Compares user data to historical trends.
Personal Stats vs. Historical Averages – Highlights key differences.
Feature Importance (Decision Tree) – Shows which health factors influence predictions