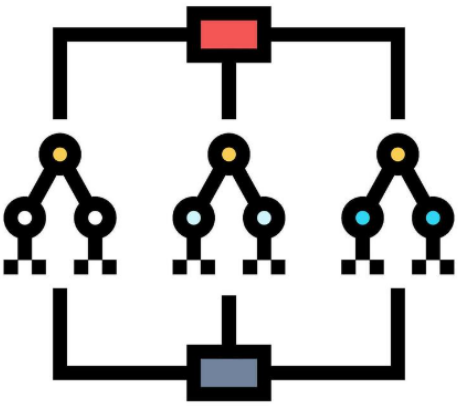
Random forest classifier for heart disease prediction

In this project, we will develop a classifier using the Random Forest machine learning algorithm to predict the presence of heart failure based on specific medical measurements.

The data and medical information used in this project are taken from this shared dataset: https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction



NIKOLAOS BELIMPASAKIS

B.Sc. | Computer Engineering and Informatics
University of Patras

Components

Information about heart failure and the dataset used

Cardiovascular diseases (CVDs) are the number 1 cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Four out of 5 CVD deaths are due to heart attacks and strokes, and one-third of these deaths occur prematurely in people under 70 years of age. Heart failure is a common event caused by CVDs and this dataset contains 11 features that can be used to predict a possible heart disease.

People with cardiovascular disease or who are at high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidaemia or already established disease) need early detection and management wherein a machine learning model can be of great help.

The dataset used in this project contains clinical records related to heart failure prediction. It consists of 12 attributes, including 11 input features and 1 target variable. Each instance represents a patient and includes the following variables [1].

1. Age: age of the patient [years]
2. Sex: sex of the patient [M: Male, F: Female]
3. ChestPainType: chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
4. RestingBP: resting blood pressure [mm Hg]
5. Cholesterol: serum cholesterol [mm/dl]
6. FastingBS: fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
7. RestingECG: resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
8. MaxHR: maximum heart rate achieved [Numeric value between 60 and 202]
9. ExerciseAngina: exercise-induced angina [Y: Yes, N: No]
10. Oldpeak: oldpeak = ST [Numeric value measured in depression]
11. ST_Slope: the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
12. HeartDisease: output class [1: heart disease, 0: Normal]

This dataset was created by combining different datasets already available independently but not combined before. In this dataset, 5 heart datasets are combined over 11 common features which makes it the largest heart disease dataset available so far for research purposes. The five datasets used for its curation are:

- Cleveland: 303 observations
- Hungarian: 294 observations
- Switzerland: 123 observations
- Long Beach VA: 200 observations
- Stalog (Heart) Data Set: 270 observations

Total: 1190 observations
Duplicated: 272 observations

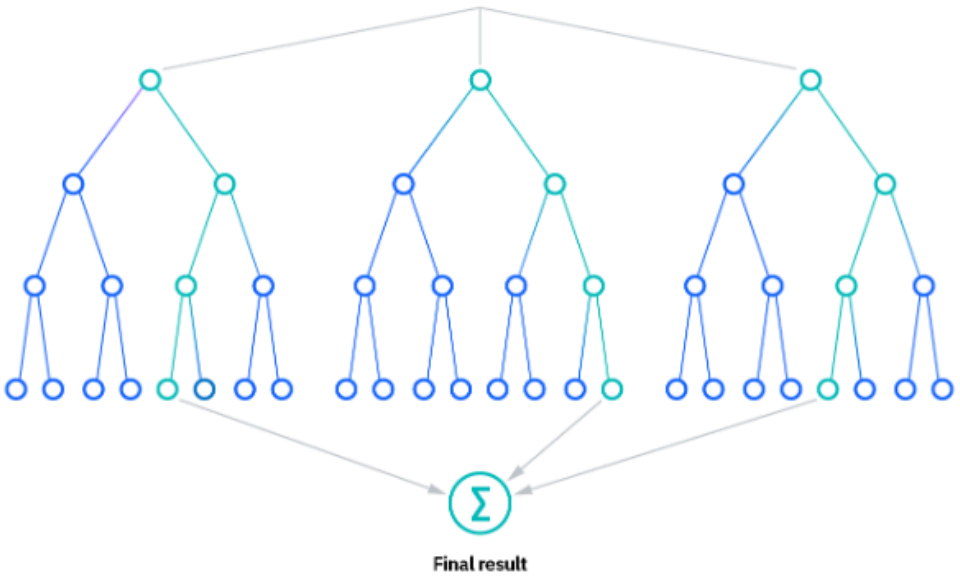
```
Final dataset: 918 observations
```

The Random Forest ML algorithm

Random Forest is a widely-used ensemble machine learning algorithm that constructs multiple decision trees during training and outputs the mode of the classes (classification) or mean prediction (regression) of the individual trees. This approach enhances predictive accuracy and controls overfitting by combining the results of various uncorrelated decision trees.

Each tree in the forest is built from a random subset of the training data, selected with replacement—a technique known as bootstrap aggregating or "bagging." Furthermore, when splitting nodes during tree construction, Random Forest considers a random subset of features, introducing additional randomness that decorrelates the decision trees and improves model robustness.

Key advantages of Random Forest include its ability to handle both classification and regression tasks, robustness to noise and overfitting, and the provision of feature importance measures, which help in understanding the influence of each feature on the prediction. However, it can be computationally intensive and less interpretable compared to single decision trees. [2]



Structure of a Random Forest model. [2]

The code

```python
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

if __name__ == "__main__":
    filename = "data.csv"

    # 1) Load the CSV directly into a DataFrame
    df = pd.read_csv(filename)

    # 2) Convert numerical columns from string to float
    numeric_cols = ['Age', 'RestingBP', 'Cholesterol', 'FastingBS', 'MaxHR', 'Oldpeak', 'HeartDisease']
    for col in numeric_cols:
        df[col] = df[col].astype(float)

    # 3) One-hot encoding for categorical columns
    categorical_cols = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
    df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

    # 4) Separate features (X) and target (y)
    X = df.drop('HeartDisease', axis=1)
    y = df['HeartDisease']

    # 5) Split into training and test sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state= 42)

    # 6) Train Random Forest
    clf = RandomForestClassifier(n_estimators=150, random_state= 42)
    clf.fit(X_train, y_train)

    # 7) Predict and evaluate
    y_pred = clf.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)

    print(f"\nRandom Forest Classifier Accuracy: {accuracy:.3f}")
```

The project's code.

The code begins by importing the necessary libraries for data handling and machine learning (lines 1–4), including pandas [3] for data manipulation and modules from scikit-learn [4] for model building and evaluation. In line 10, the dataset is loaded directly from a CSV file into a pandas DataFrame using pd.read_csv(). Numerical columns, which are initially read as strings, are explicitly converted to floats in lines 13–15 to ensure compatibility with the model. In lines 18–19, one-hot encoding is applied to categorical features using pd.get_dummies() with drop_first=True to prevent multicollinearity.

The features (X) and target variable (y) are separated in lines 22–23, with 'HeartDisease' being the target column. Next, in line 26, the dataset is split into training and testing subsets using train_test_split, reserving 20% of the data for testing and ensuring reproducibility by setting random_state=42.

A RandomForestClassifier is instantiated in line 29 with n_estimators=150, meaning the ensemble will consist of 150 decision trees. The model is trained using the training data in line 30 with the fit() method. Predictions are made on the test set in line 33, and the model's performance is evaluated using accuracy in line 34. Finally, the accuracy score is printed with three decimal precision (line 36).

This structure ensures that the data undergoes proper preprocessing before training, and the Random Forest model is built and assessed in a reproducible and systematic way.

Future work

As a continuation of this project, we can develop an application that receives patient data (medical measurements) either manually or through scanning with a camera, and displays whether the patient is at risk of developing heart failure. In addition to using a Random Forest classifier, a Large Language Model (LLM) could be integrated to support the classifier's prediction by providing a natural language explanation of the result—clarifying why the model reached that specific conclusion.

A framework that can be used for the development of the application is Django [5], and an LLM that can be easily and freely accessed via API is Google's Gemini 1.5 flash [6].

References

1. Kaggle. “Heart failure prediction dataset”. https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction/data

2. IBM. “What is random forest?”. https://www.ibm.com/think/topics/random-forest

3. Wes McKinney. “pandas: a Foundational Python Library for Data Analysis and Statistics”. https://d1wqtxts1xzle7.cloudfront.net/117768488/pyhpc2011_submission_9-libre.pdf?1724818056=&response-content-disposition=inline%3B+filename%3DAnaplastic_Lymphoma_Kinase_ALK_positive.pdf&Expires=1749056911&Signature=UZ6Qi7ScA73hPDpoHtY~gPcsLxN3h8ps-uWf7hFyA~tKFMtQq0trDFkTnAp1enmMRIOrWRdjqrEzxl7BFGhzP3JJkDY~GOSFRqHuvUqclrjTpIWf5VFs8FTEqLPcv1zryf7L6lAf~qFRvn9cfwFD1PeW-JGfXiN4KxrUl-kylqIefP1c4mmGxcIy8fKYnROP58xi1IHlByfw5y1CUkZEwEgfRqxrFtrLlBNSURJsD2d1G0L7lsjM~Yr67jisfnfpd2pTyABMoKR3Wgq1xSxRrm2MMsfD-Nju1TF6a-MFPqOeIsJedCF1JoW1Hb45gK2U75~diMjFncBznt2gofR5Ew__&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA

4. Fabian Pedregosa et al. “Scikit-learn: Machine Learning in Python”. https://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf?source=post_page

5. Django Software Foundation. “Meet Django”. https://www.djangoproject.com/

6. Google. “Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context”. https://arxiv.org/pdf/2403.05530