Heart Failure Classification

Yahia Ibrahim AlKaranshawy, Hossam Osama Iraqi, Ali Hassan Ali

March 13, 2025

1 Problem statement

It's required to implement a decision tree model, bagging model and boosting model from scratch and compare the performance of the three approaches using the heart failure classification dataset.

2 Exploratory Data Analysis (EDA)

2.1 Distribution of Numerical Features

- Histograms were plotted for numerical features to visualize their distribution with respect to the target variable (`HeartDisease`).
- **Key Observations**: Features like `Age`, `RestingBP`, and `Cholesterol` showed different distributions for patients with and without heart disease.

2.2 Distribution of Categorical Features

- Bar plots were used to visualize the distribution of categorical features like `Sex`, `ChestPainType`, and `ExerciseAngina`.
- **Key Observations**: Males were more likely to have heart disease, and certain chest pain types were more associated with heart disease.

2.3 Correlation Matrix

- A heatmap of the correlation matrix was plotted to identify relationships between numerical features.
- **Key Observations**: Features like—Oldpeak— showed moderate correlations with the target variable.

3 Data Preprocessing

3.1 Data Loading and Initial Exploration

- **Dataset Overview**: The dataset contains 918 rows and 12 columns. The initial exploration revealed no missing values, but some columns like `RestingBP` and `Cholesterol` had zero values, which were replaced with `NaN` and later imputed.
- **Data Types**: The dataset contains both numerical and categorical features. Categorical features were encoded using `LabelEncoder`.

3.2 Handling Missing Values

- **Imputation**: Zero values in `RestingBP` and `Cholesterol` were replaced with `NaN` and imputed using KNN imputation.
- **Null Values**: After imputation, the dataset had no missing values.

3.3 Feature Engineering

- **Scaling**: Three scalers were used: `StandardScaler`, `MinMaxScaler`, and `RobustScaler`. Each scaler was applied to the dataset before training the models.
- **Outlier Detection**: Boxplots were used to visualize outliers in numerical features, but no significant outliers were removed.

3.4 Data Splitting

The dataset was split into training (70%), validation (10%), and test (20%) sets. The target variable was transformed to {-1, 1} for compatibility with some models.

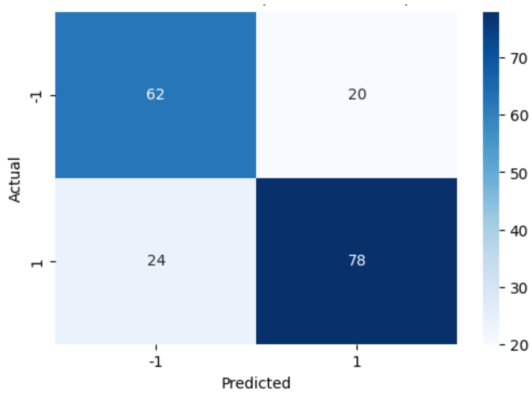
4 Results

4.1 Test Accuracy & F1 Scores

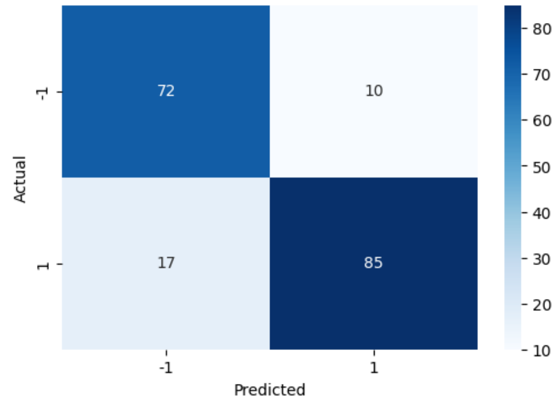
Model	**Accuracy**	**F1 Score**	**Scaler Used**
Decision Tree	0.7609	0.7800	StandardScaler
Bagging (DT)	0.8533	0.8629	StandardScaler
Boosting (DT)	0.8587	0.8700	StandardScaler
KNN	0.8587	0.8585	MinMaxScaler
Logistic Regression	0.8478	0.8480	StandardScaler
Fully Connected NN	0.8370	0.8374	MinMaxScaler

Table 1: Performance comparison of different models

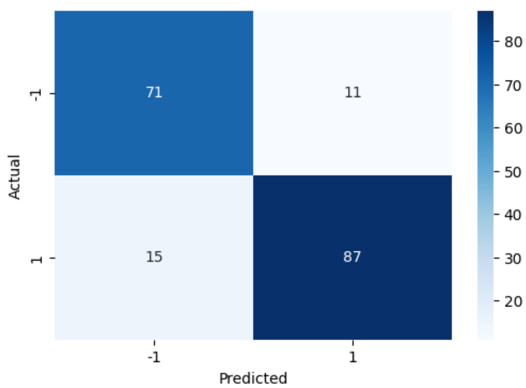
4.2 Confusion Matrices



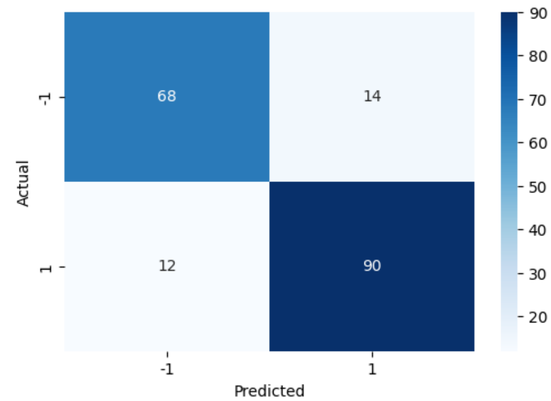
(a) Decision Tree



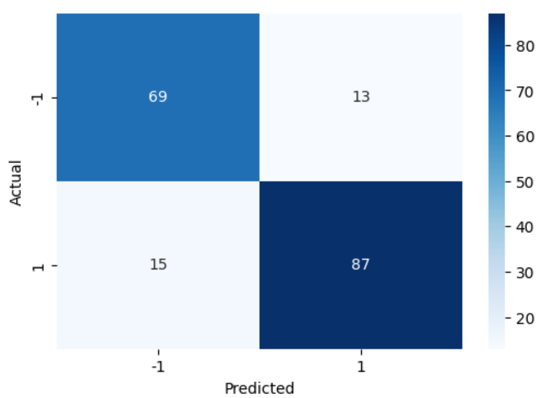
(b) Bagging (DT)



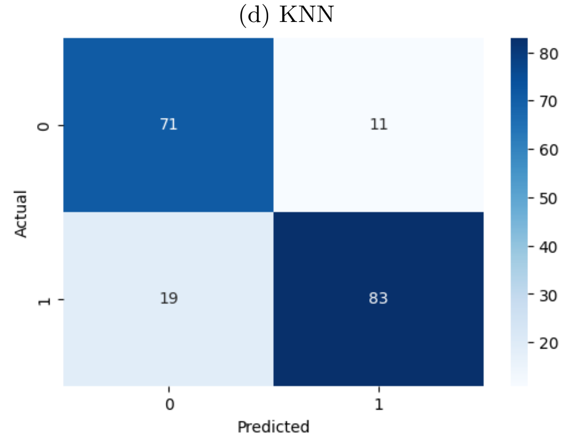
(c) Boosting (DT)



(d) KNN



(e) Logistic Regression



(f) Fully Connected Neural Network

Figure 1: Confusion Matrices for Different Models

5 Conclusion

- **Best Models:** Boosting (DT) (85.87% accuracy, 87.00% F1) and Bagging (DT) (85.33% accuracy, 86.29% F1) outperform a single Decision Tree (76.09% accuracy, 78.00% F1).
- **Why?**
  - Decision Tree overfits.
  - Bagging (DT) reduces variance → more stable.
  - Boosting (DT) reduces bias → best performance.
- **Confusion Matrix Insights:**
  - Boosting (DT) has **lowest false negatives (15)** → best for recall.
  - Bagging (DT) balances false positives & negatives well.
  - Decision Tree has **high misclassification rates**.
- **Conclusion:**
  - Choose **Boosting (DT)** for **high recall**.
  - Choose **Bagging (DT)** for **balanced performance**.
  - **Avoid a single Decision Tree** due to overfitting.

References