Heart Disease Prediction

Milestone 1,2

Team 5

Seif Amjad Sobeih Heidar	23p0145
Abdelrahman Wael Maher	23p0155
Ahmed Hossam Abdallah	23p0109
Youssef Ahmed Farouk	23p0054
Youssef Farid Haddad	23P0113
Marwan Wagih Mohammed	23P0247

Table of Contents

1. Introduction

Heart disease remains one of the leading causes of mortality worldwide, creating a strong need for early detection systems that can assist healthcare providers in identifying high-risk patients. Machine learning has become an essential tool in this area due to its ability to analyze patterns in medical data and produce accurate predictions.

This project aims to build a predictive model that determines whether a patient has heart disease based on several clinical features.
Milestone 1 focuses on:

- Understanding and describing the dataset
- Preprocessing and cleaning the data
- Performing Exploratory Data Analysis (EDA)

This milestone establishes the foundation for more advanced modeling in later stages.

2. Dataset Description

This project uses a structured **Heart Disease dataset** commonly utilized in medical predictive analysis. It includes demographic information, health measurements, and results from various medical tests.

The dataset contains both **categorical** and **numerical** features, each representing important aspects of a patient's cardiovascular health.

2.1 Categorical Features

Feature	Medical name	Values	Meaning
Sex	sex	1,0	1 : male 0 : female
Cp	Chest Pain	0, 1, 2, 3	0 = typical angina 1 = atypical angina 2 = non-anginal pain 3 = asymptomatic
Fbs	Fasting Blood Sugar	1,0	1 = fasting blood sugar > 120 mg/dl
Restecg	Resting Electrocardiogram	0,1,2	0 = normal 1 = ST-T abnormality 2 = left ventricular hypertrophy
Exang	Exercise-Induced Angina	0,1	0 = no, 1 = yes
Slope	Slope of peak exercise ST segment	0,1,2	0 = upsloping 1 = flat 2 = down sloping
Ca	Major Vessels Colored by Fluoroscopy	0,1,2,3,4 -Value 4 is suspicious	Count of major vessels colored by fluoroscopy
Thal	Thallium Stress Test Result	0,1,2,3 -0 often indicates missing	1 = normal 2 = fixed defect 3 = reversible defect
target	Heart Disease Diagnosis	1,0	1 = disease, 0 = no disease

2.2 Numerical Features

Feature	**Medical Name**	Dataset Range
age	**Age**	**29-77**
trestbps	**Resting BP**	**90–200 mmHg**
chol	**Cholesterol**	**120–570 mg/dl**
thalach	**Maximum HR**	**70–200 bpm**
oldpeak	**ST Depression**	**0–6.5**

3. Data Preprocessing

3.1 Dataset Structure and Data Types

The dataset contains 303 entries and 14 features, as shown by the df.info() output. All columns are numerical (non null), with 13 features stored as integers and one feature (oldpeak) stored as a float. Because the dataset contains only numeric values, no label encoding or categorical-to-numeric conversion was required at this stage. The presence of fully numeric data also simplifies downstream modeling and preprocessing operations.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       303 non-null    int64
 1   sex       303 non-null    int64
 2   cp        303 non-null    int64
 3   trestbps  303 non-null    int64
 4   chol      303 non-null    int64
 5   fbs       303 non-null    int64
 6   restecg   303 non-null    int64
 7   thalach   303 non-null    int64
 8   exang     303 non-null    int64
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64
 11  ca        303 non-null    int64
 12  thal      303 non-null    int64
 13  target    303 non-null    int64
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

3.2 Summary Statistics

A descriptive overview using df.describe() provides detailed statistical information for each feature, including mean, standard deviation, quartiles, and minimum/maximum values. For example, the average patient age is 54.36 years, the average resting blood pressure is 131 mmHg, and cholesterol levels range widely from 126 to 564 mg/dl. These statistics help identify the general distribution and potential abnormalities in each feature.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000

3.3 Missing Values Analysis

A check for missing data using df.isnull().sum() confirms that the dataset contains no missing values in any of the 14 features. This means no data imputation or cleaning was required to fill gaps, allowing the preprocessing pipeline to proceed directly to duplication and distribution analysis.

```
age         0
sex         0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalach     0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
target      0
dtype: int64
```

3.4 Duplicate Record Handling

Duplicate detection using df.duplicated().sum() revealed one duplicated entry. The duplicated row was identified and displayed using df[df.duplicated(keep=False)], confirming that two identical records existed. These were removed using df.drop_duplicates(). Eliminating duplicates helps ensure that no patient record is unintentionally overweighted during model training.

```
df.duplicated().sum()
```

```
1
```

```
df[df.duplicated(keep=False)]
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
163	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1
164	38	1	2	138	175	0	1	173	0	0.0	2	4	2	1

```
df = df.drop_duplicates()
```

```
df.duplicated().sum()
```

```
0
```

3.5 Separation of Categorical and Numerical Features

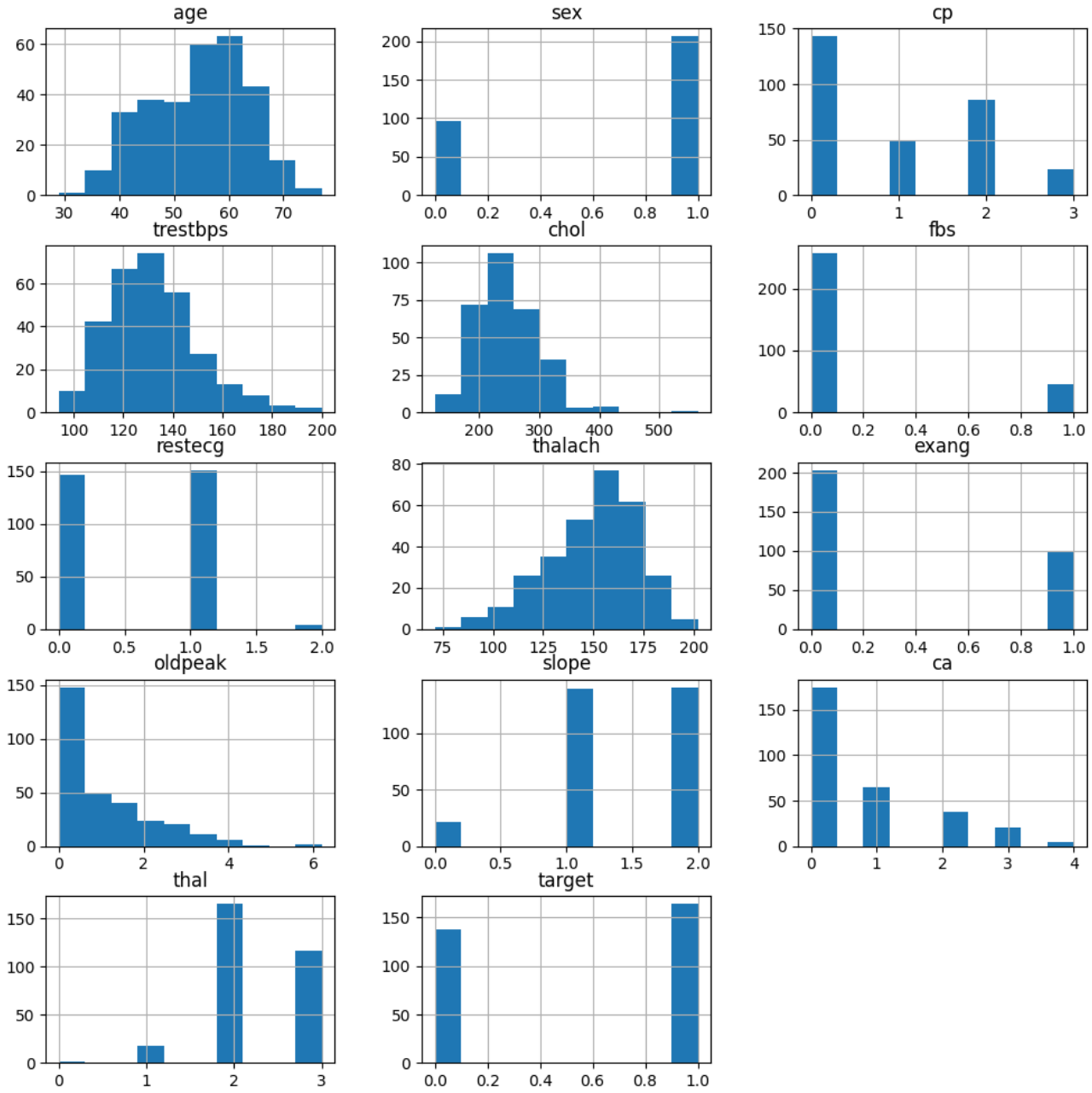
The project separates features based on their number of unique values:

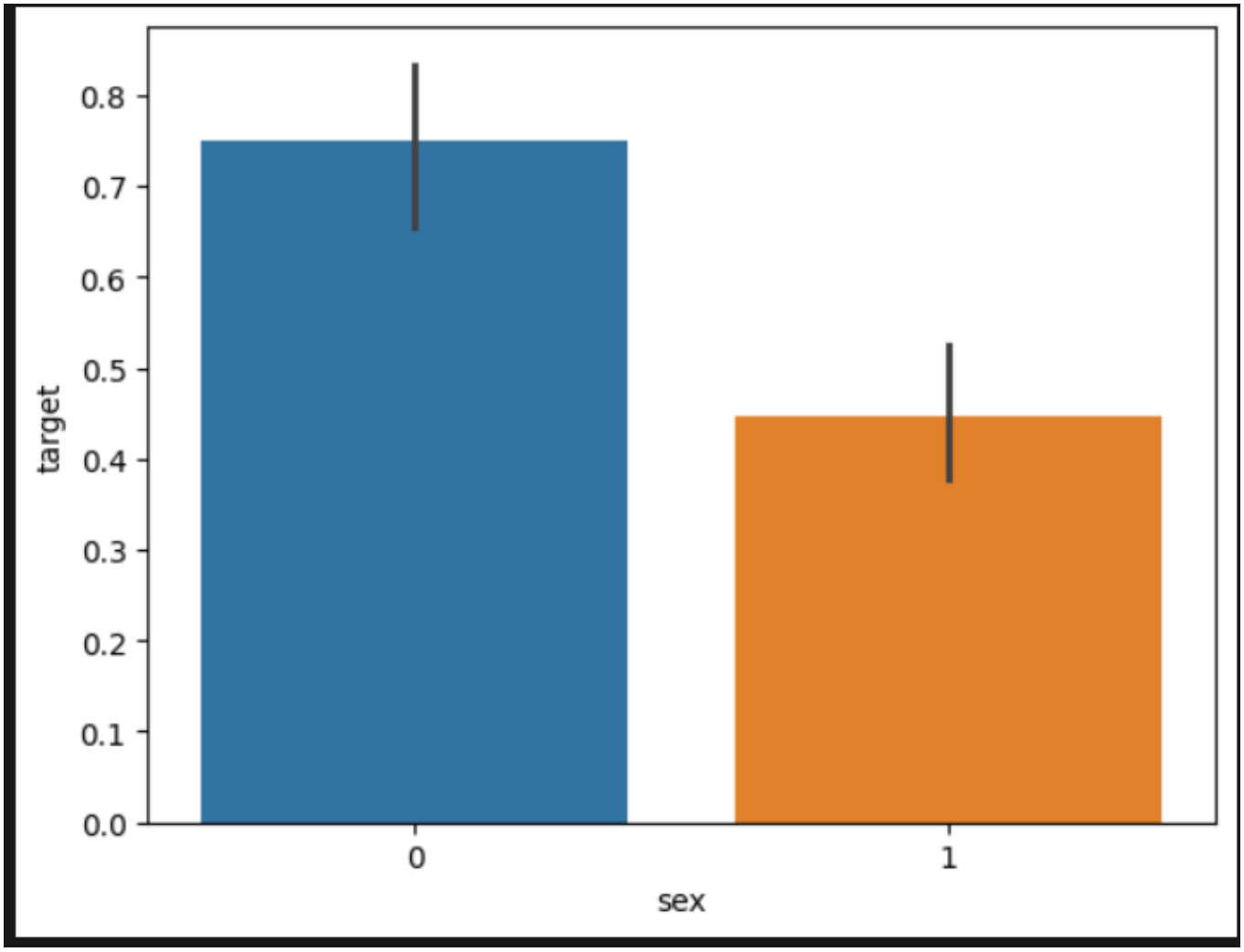
- Categorical features: features with fewer than 10 unique values
  → sex, cp, fbs, restecg, exang, slope, ca, thal, target
- Numerical features: features with higher variability
  → age, trestbps, chol, thalach, oldpeak

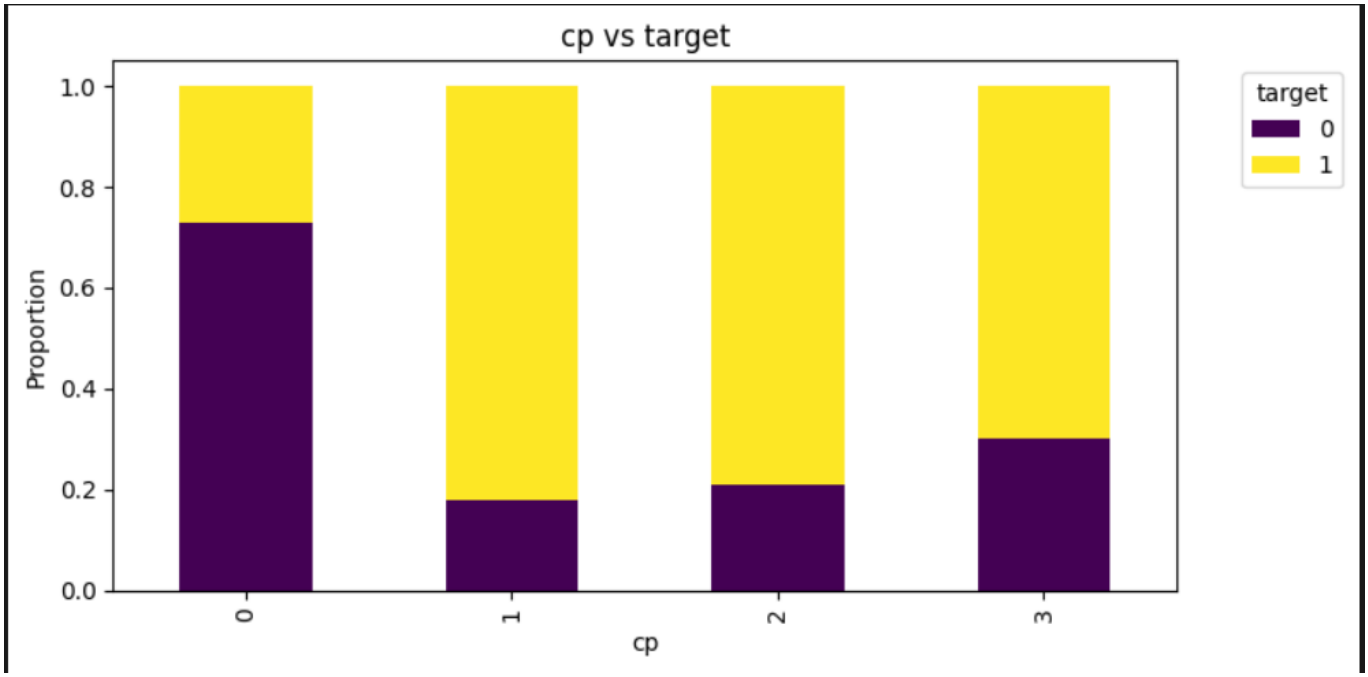
4. Exploratory Data Analysis

4.1 Distribution Analysis (Histograms & Countplots)

Histograms were used to display the distribution of numerical features, showing that age is mostly clustered between 45 and 65 years, cholesterol and resting blood pressure have a wide spread, and oldpeak is right-skewed. Countplots were used for categorical variables such as chest pain type, ECG results, and exercise-induced angina, confirming clear patterns and distinct category frequencies. Additionally, histogram plots were created using *hue* = *target* to differentiate how each feature behaves for patients with and without heart disease.

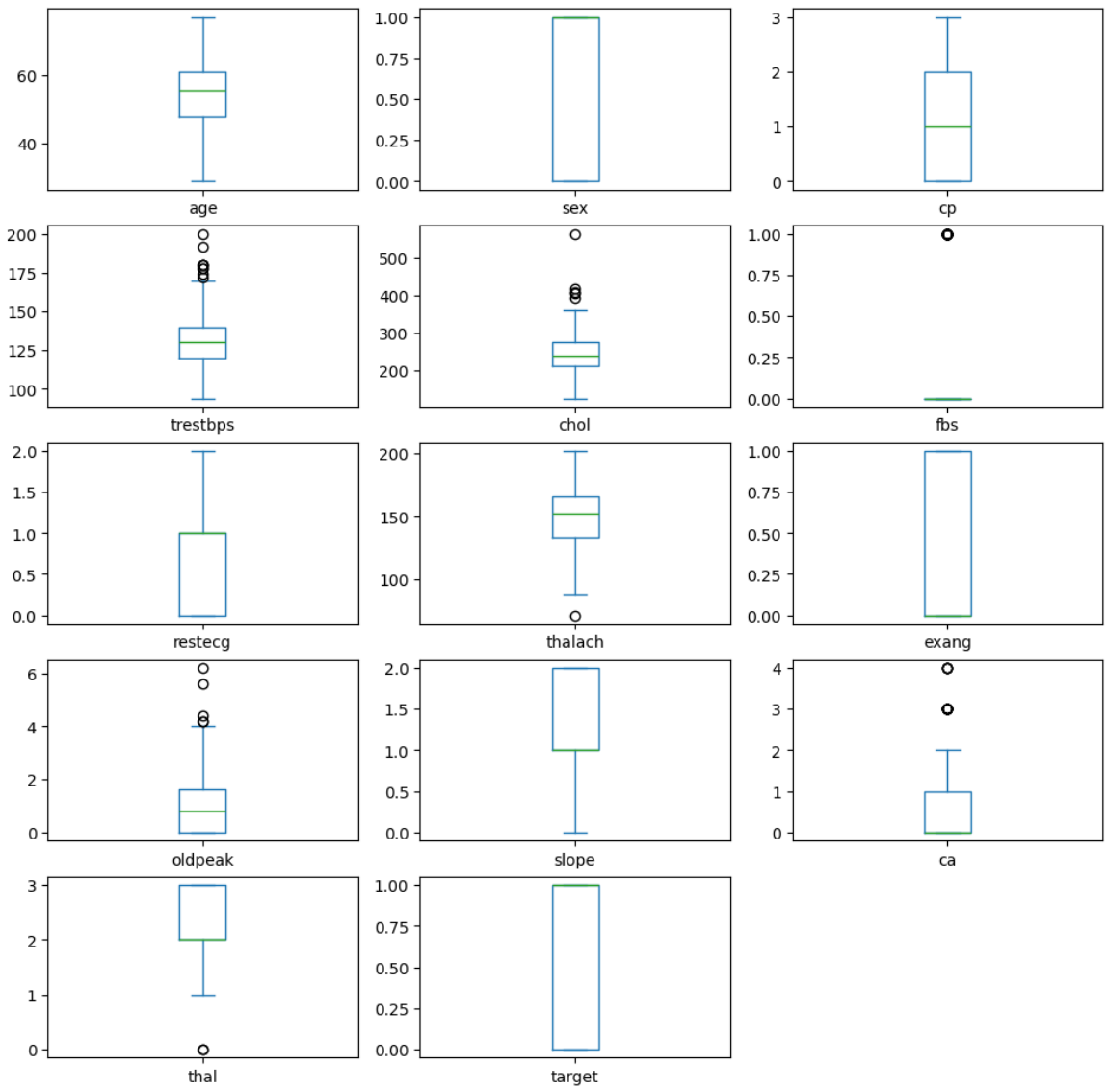

age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
target
0
1
sex


cp vs target
1.0
0.8
0.6
0.4
0.2
0.0
Proportion
0
1
2
3
cp
target
0
1

4.2 Outlier Detection (Boxplots)

Boxplots were used to detect outliers in the numerical features, revealing noticeable high cholesterol outliers and variability in oldpeak, while features like resting blood pressure and maximum heart rate showed moderate dispersion. These boxplots helped identify which variables may require scaling or careful handling during the modeling phase. Boxplots were also generated with *hue = target*, allowing us to compare feature distributions for diseased versus non-diseased patients and visually assess differences between the two groups.



4.3 Handling Invalid Values

4.3.1 No Imputation (Keeping Them as Outliers)

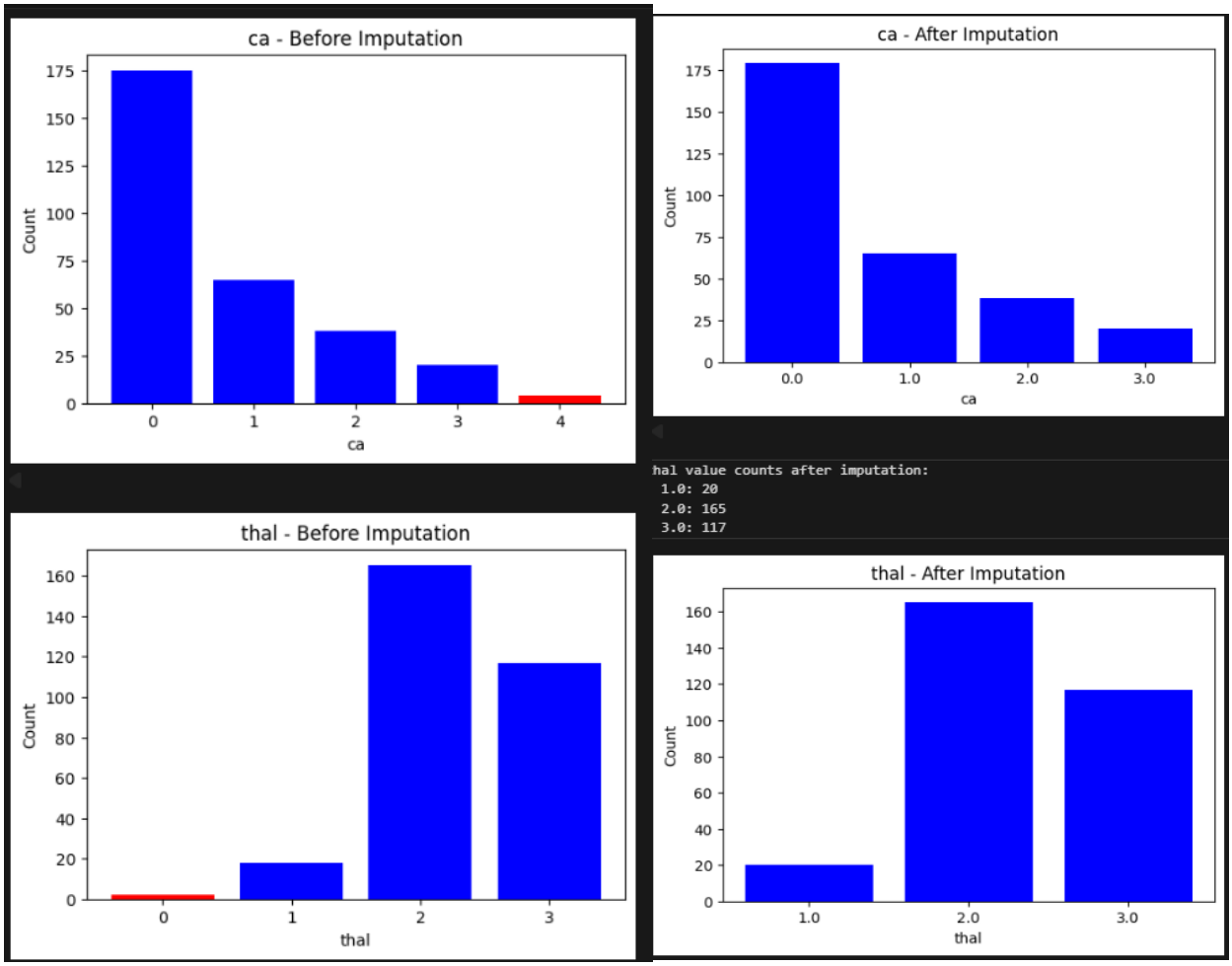
In the first approach, the invalid values in *ca* and *thal* were left unchanged so they could be treated later as true outliers. This method allows the unusual values (such as ca = 4 and thal = 0) to be captured by the outlier detection models rather than being removed or corrected early. The initial bar plots clearly show these rare values appearing in very small counts, highlighted in red.

4.3.2 Simple Imputation (Replacing Invalid Values With Mode)

A simple imputation method was also tested, where invalid values were replaced using the most frequent valid category in each feature. This approach quickly removes rare or impossible categories while keeping the overall distribution similar to the original data. The “after imputation” plots illustrate how the invalid classes disappear while the remaining categories maintain their general proportions.

4.3.3 KNN Imputation (Using Nearest Neighbors)

Finally, KNN Imputation was applied to *ca* and *thal* to estimate missing or invalid values based on the closest neighboring samples. This method provides a more data-aware correction, assigning new values that reflect the patterns in related features rather than relying on a single dominant category. After KNN imputation, the distributions of *ca* and *thal* appear smooth and consistent, with invalid values replaced by realistic and context-appropriate alternatives.



4.4 Categorical Outlier Detection (Rare Category Method)

Categorical outliers were identified by examining the frequency of each category and marking any value that occurred in less than 5% of the dataset. This rare-category method helps detect unusual or unexpected category values that may represent data entry errors or very uncommon patient profiles. Rows containing any rare categorical value were flagged and recorded for further inspection.

Detected 4 categorical outliers out of 302 rows (1.32% of data).

	sex	cp	fbs	restecg	exang	slope	ca	thal	target	cat_outlier_explanation
144	0	2	0	2	0	1	0.0	2.0	1	Rare category '2' in restecg (1.32% of data)
266	0	0	0	2	1	1	0.0	2.0	0	Rare category '2' in restecg (1.32% of data)
289	0	0	0	2	1	1	1.0	3.0	0	Rare category '2' in restecg (1.32% of data)
291	1	0	0	2	0	0	3.0	1.0	0	Rare category '2' in restecg (1.32% of data)

Figure 1: KNN and Simple

Detected 10 categorical outliers out of 302 rows (3.31% of data).

	sex	cp	fbs	restecg	exang	slope	ca	thal	target	cat_outlier_explanation
48	0	2	0	0	0	2	0	0	1	Rare category '0' in thal (0.66% of data)
92	1	2	0	1	0	2	4	2	1	Rare category '4' in ca (1.32% of data)
144	0	2	0	2	0	1	0	2	1	Rare category '2' in restecg (1.32% of data)
158	1	1	0	1	0	1	4	3	1	Rare category '4' in ca (1.32% of data)
163	1	2	0	1	0	2	4	2	1	Rare category '4' in ca (1.32% of data)
251	1	0	1	0	1	1	4	3	0	Rare category '4' in ca (1.32% of data)
266	0	0	0	2	1	1	0	2	0	Rare category '2' in restecg (1.32% of data)
281	1	0	1	1	1	1	0	0	0	Rare category '0' in thal (0.66% of data)
289	0	0	0	2	1	1	1	3	0	Rare category '2' in restecg (1.32% of data)
291	1	0	0	2	0	0	3	1	0	Rare category '2' in restecg (1.32% of data)

Figure 2:without any imputer

4.5 Isolation Forest (Numerical Outliers)

Isolation Forest was applied to the numerical features to identify observations that behave differently from the majority of the data. This algorithm isolates anomalies by randomly partitioning the feature space and identifying points that require fewer splits to isolate. The method detected numerical patterns that sharply deviated from typical patient measurements, marking them as potential outliers.

Isolation Forest outliers: 16

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
1	37	1	2	130	250	0	1	187	0	3.5	0	0.0	2.0	1
72	29	1	1	130	204	0	0	202	0	0.0	2	0.0	2.0	1
85	67	0	2	115	564	0	0	160	0	1.6	1	0.0	3.0	1
101	59	1	3	178	270	0	0	145	0	4.2	0	0.0	3.0	1
125	34	0	1	118	210	0	1	192	0	0.7	2	0.0	2.0	1
175	40	1	0	110	167	0	0	114	1	2.0	1	0.0	3.0	0
204	62	0	0	160	164	0	0	145	0	6.2	0	3.0	3.0	0
220	63	0	0	150	407	0	0	154	0	4.0	1	3.0	3.0	0
221	55	1	0	140	217	0	1	111	1	5.6	0	0.0	3.0	0
223	56	0	0	200	288	1	0	133	1	4.0	0	2.0	3.0	0
248	54	1	1	192	283	0	0	195	0	0.0	2	1.0	3.0	0
259	38	1	3	120	231	0	1	182	1	3.8	1	0.0	3.0	0
266	55	0	0	180	327	0	2	117	1	3.4	1	0.0	2.0	0
272	67	1	0	120	237	0	1	71	0	1.0	1	0.0	2.0	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3.0	1.0	0
297	59	1	0	164	176	1	0	90	0	1.0	1	2.0	1.0	0

Figure 3:KNN, simple and without any imputer

4.6 Local Outlier Factor (LOF)

Local Outlier Factor was used to evaluate the density around each data point, flagging observations whose local density was significantly lower than that of their neighbors. This method highlights patient records that do not conform to the local structure of the data and may indicate rare or inconsistent physiological patterns.

LOF outliers: 16

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
28	65	0	2	140	417	1	0	157	0	0.8	2	1.0	2.0	1
39	65	0	2	160	360	0	0	151	0	0.8	2	0.0	2.0	1
53	44	0	2	108	141	0	1	175	0	0.6	1	0.0	2.0	1
72	29	1	1	130	204	0	0	202	0	0.0	2	0.0	2.0	1
85	67	0	2	115	564	0	0	160	0	1.6	1	0.0	3.0	1
96	62	0	0	140	394	0	0	157	0	1.2	1	0.0	2.0	1
111	57	1	2	150	126	1	1	173	0	0.2	2	1.0	3.0	1
151	71	0	0	112	149	0	1	125	0	1.6	1	0.0	2.0	1
220	63	0	0	150	407	0	0	154	0	4.0	1	3.0	3.0	0
223	56	0	0	200	288	1	0	133	1	4.0	0	2.0	3.0	0
246	56	0	0	134	409	0	0	150	1	1.9	1	2.0	3.0	0
248	54	1	1	192	283	0	0	195	0	0.0	2	1.0	3.0	0
266	55	0	0	180	327	0	2	117	1	3.4	1	0.0	2.0	0
272	67	1	0	120	237	0	1	71	0	1.0	1	0.0	2.0	0
297	59	1	0	164	176	1	0	90	0	1.0	1	2.0	1.0	0
301	57	1	0	130	131	0	1	115	1	1.2	1	1.0	3.0	0

Figure 4:KNN, simple and without any imputer

4.7 IQR Method (Interquartile Range Outliers)

The IQR method was used to detect outliers by calculating the interquartile range for each numerical feature and flagging values that fall below the lower bound or above the upper bound (Q1 – 1.5×IQR or Q3 + 1.5×IQR). This traditional statistical approach identifies extreme values in measurements such as cholesterol, resting blood pressure, and ST depression (oldpeak).

IQR-based numerical outliers: 19

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
8	52	1	2	172	199	1	1	162	0	0.5	2	0.0	3.0	1
28	65	0	2	140	417	1	0	157	0	0.8	2	1.0	2.0	1
85	67	0	2	115	564	0	0	160	0	1.6	1	0.0	3.0	1
96	62	0	0	140	394	0	0	157	0	1.2	1	0.0	2.0	1
101	59	1	3	178	270	0	0	145	0	4.2	0	0.0	3.0	1
110	64	0	0	180	325	0	1	154	1	0.0	2	0.0	2.0	1
203	68	1	2	180	274	1	0	150	1	1.6	1	0.0	3.0	0
204	62	0	0	160	164	0	0	145	0	6.2	0	3.0	3.0	0
220	63	0	0	150	407	0	0	154	0	4.0	1	3.0	3.0	0
221	55	1	0	140	217	0	1	111	1	5.6	0	0.0	3.0	0
223	56	0	0	200	288	1	0	133	1	4.0	0	2.0	3.0	0
241	59	0	0	174	249	0	1	143	1	0.0	1	0.0	2.0	0
246	56	0	0	134	409	0	0	150	1	1.9	1	2.0	3.0	0
248	54	1	1	192	283	0	0	195	0	0.0	2	1.0	3.0	0
250	51	1	0	140	298	0	1	122	1	4.2	1	3.0	3.0	0
260	66	0	0	178	228	1	1	165	1	1.0	1	2.0	3.0	0
266	55	0	0	180	327	0	2	117	1	3.4	1	0.0	2.0	0
272	67	1	0	120	237	0	1	71	0	1.0	1	0.0	2.0	0
291	58	1	0	114	318	0	2	140	0	4.4	0	3.0	1.0	0

Figure 5:KNN, simple, without any imputer

4.8 Regression-Based Outlier Detection (Residual Analysis)

A linear regression–based method was applied specifically to the oldpeak feature. The three numerical features most strongly correlated with oldpeak were used to train a regression model. Residuals were then computed for each observation, and the top 5% of points with the highest residual values were flagged as outliers. This approach identifies values that do not fit the expected relationship between oldpeak and its related predictors.

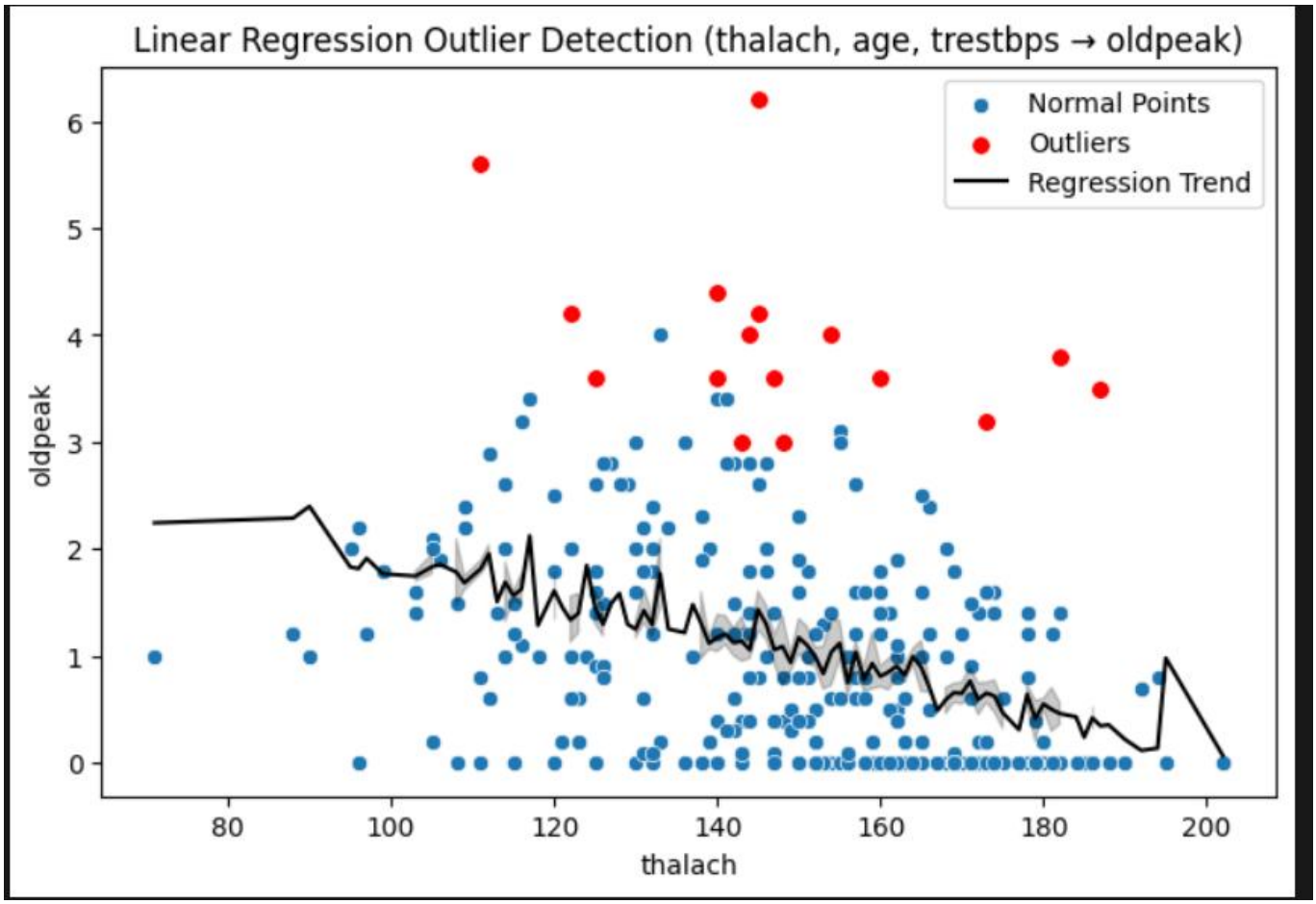


Figure 6:KNN,simple and without any imputer

Comparison between outlier methods

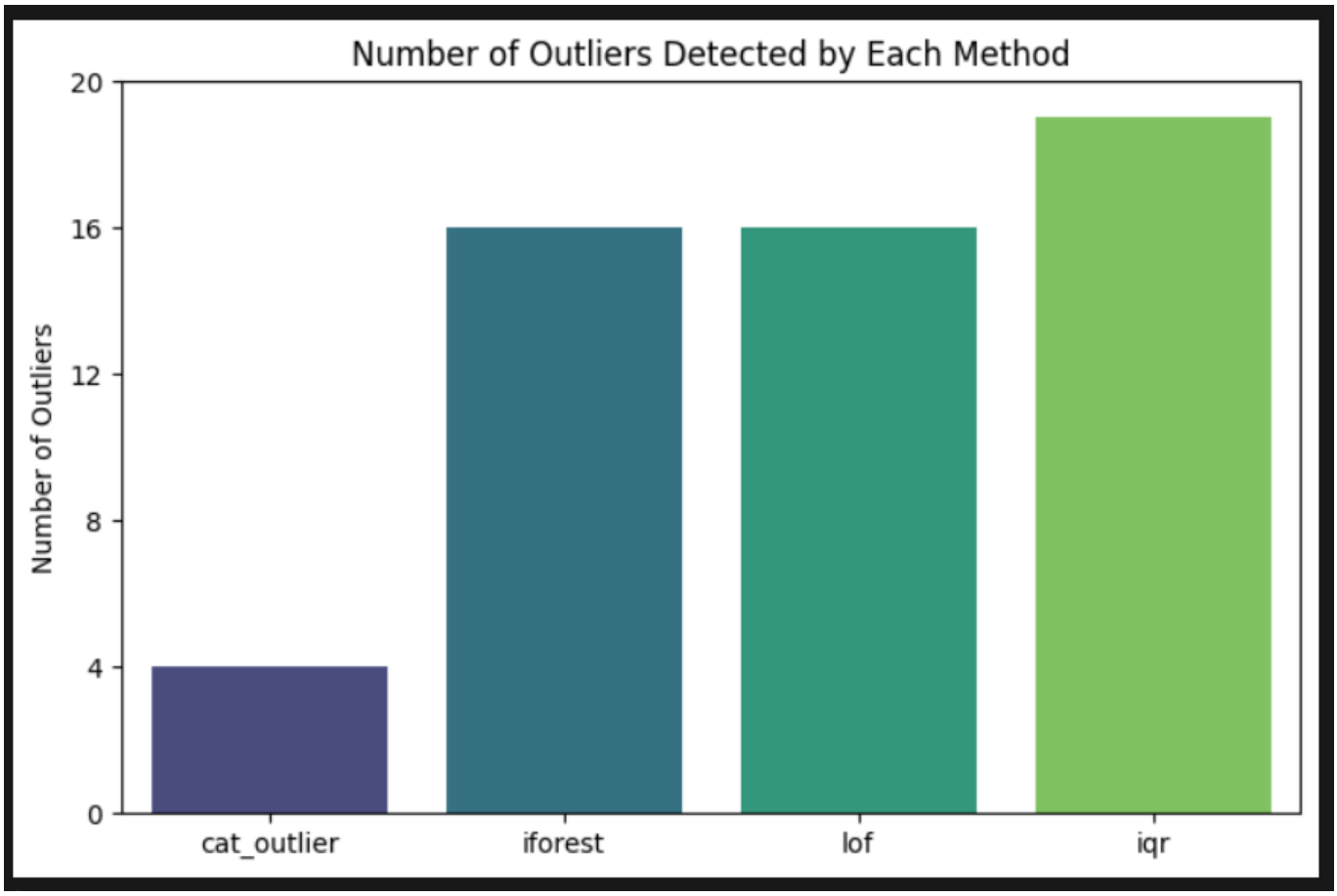
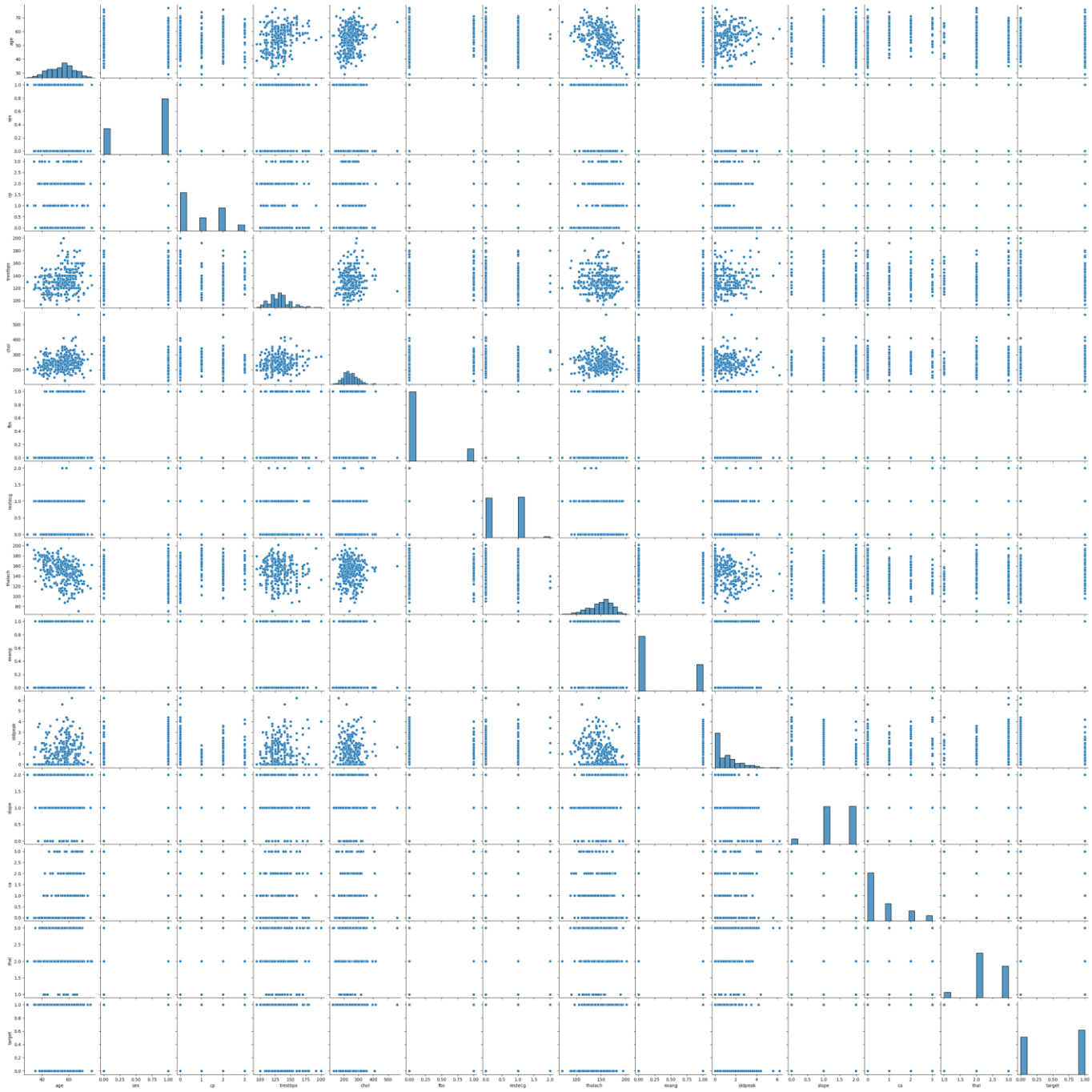


Figure 7:outlier comparison

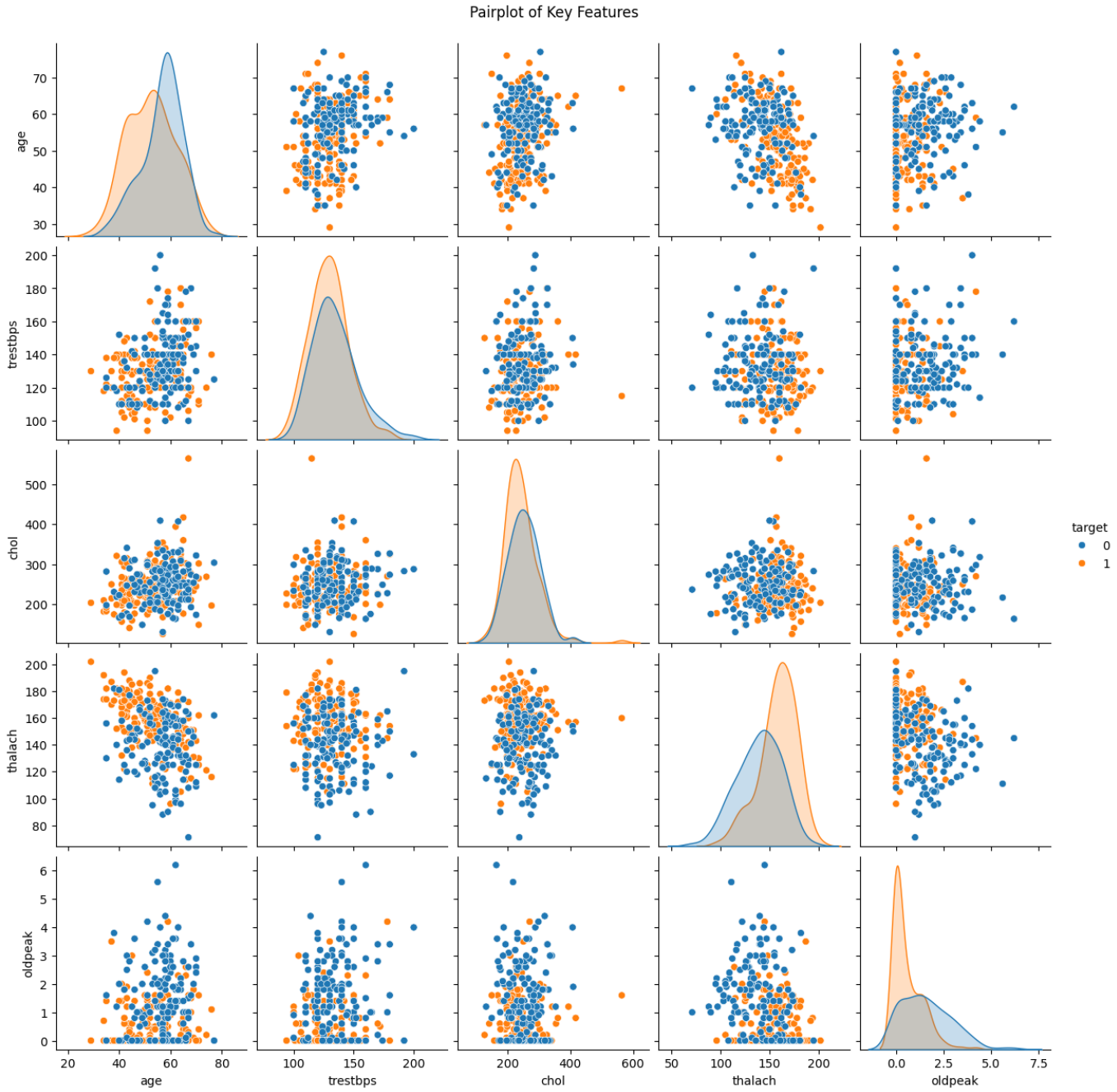
4.9 Full Pairplot of All Features

A full pairplot was generated to visualize the pairwise relationships between all features in the dataset. This comprehensive grid of scatterplots and histograms highlights how variables interact with one another and reveals general patterns such as the negative relationship between age and maximum heart rate. Although dense, the full pairplot helps identify broad trends and potential class separation when colored according to the target variable.



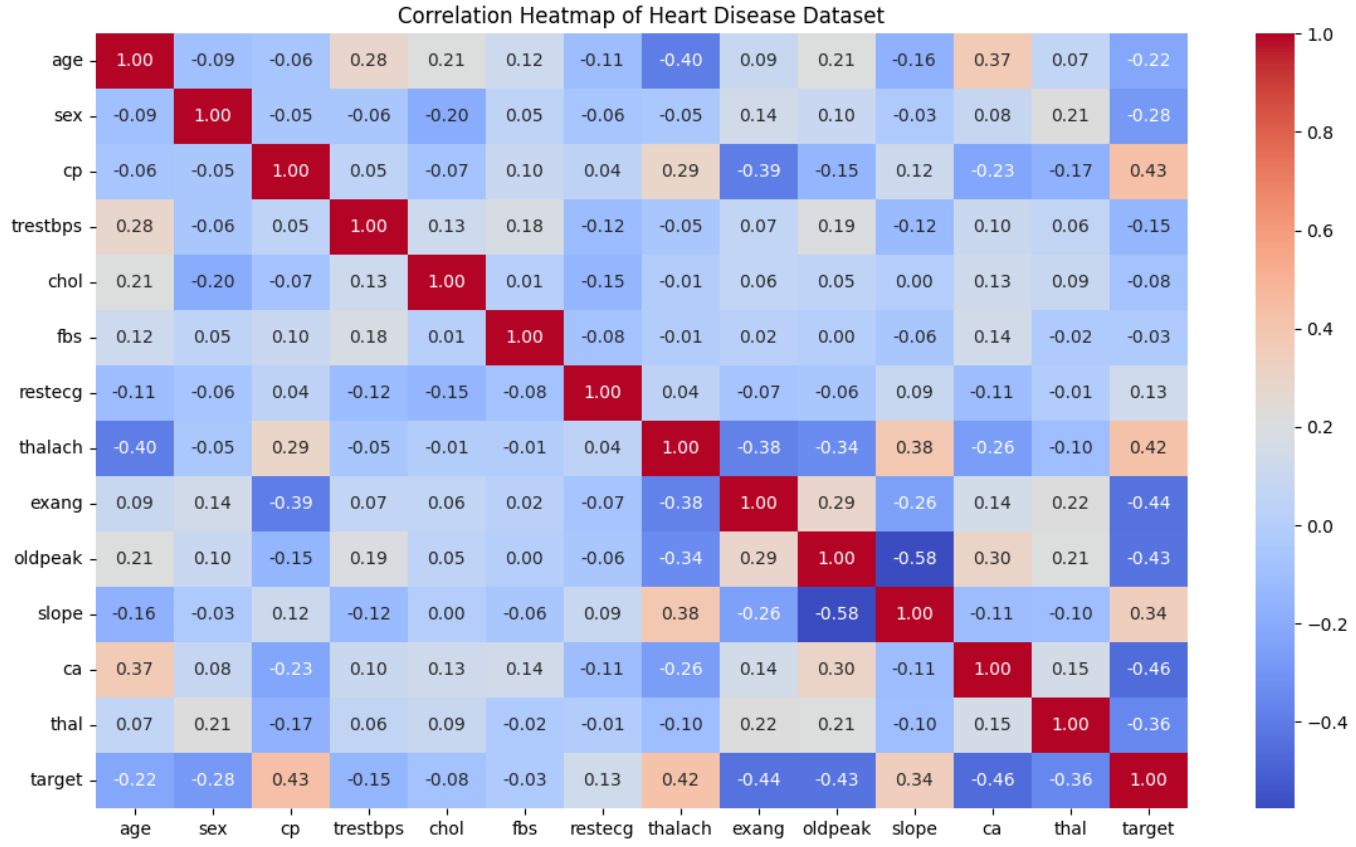
4.10 Pairplot of Key Numerical Features

A more focused pairplot was created using the most important numerical features, including age, resting blood pressure, cholesterol, maximum heart rate, and ST depression (oldpeak). When colored by the target label, the plot clearly shows that patients with heart disease tend to have higher oldpeak values and lower maximum heart rates. This simplified pairplot provides a clearer and more interpretable view of the relationships that matter most for prediction.



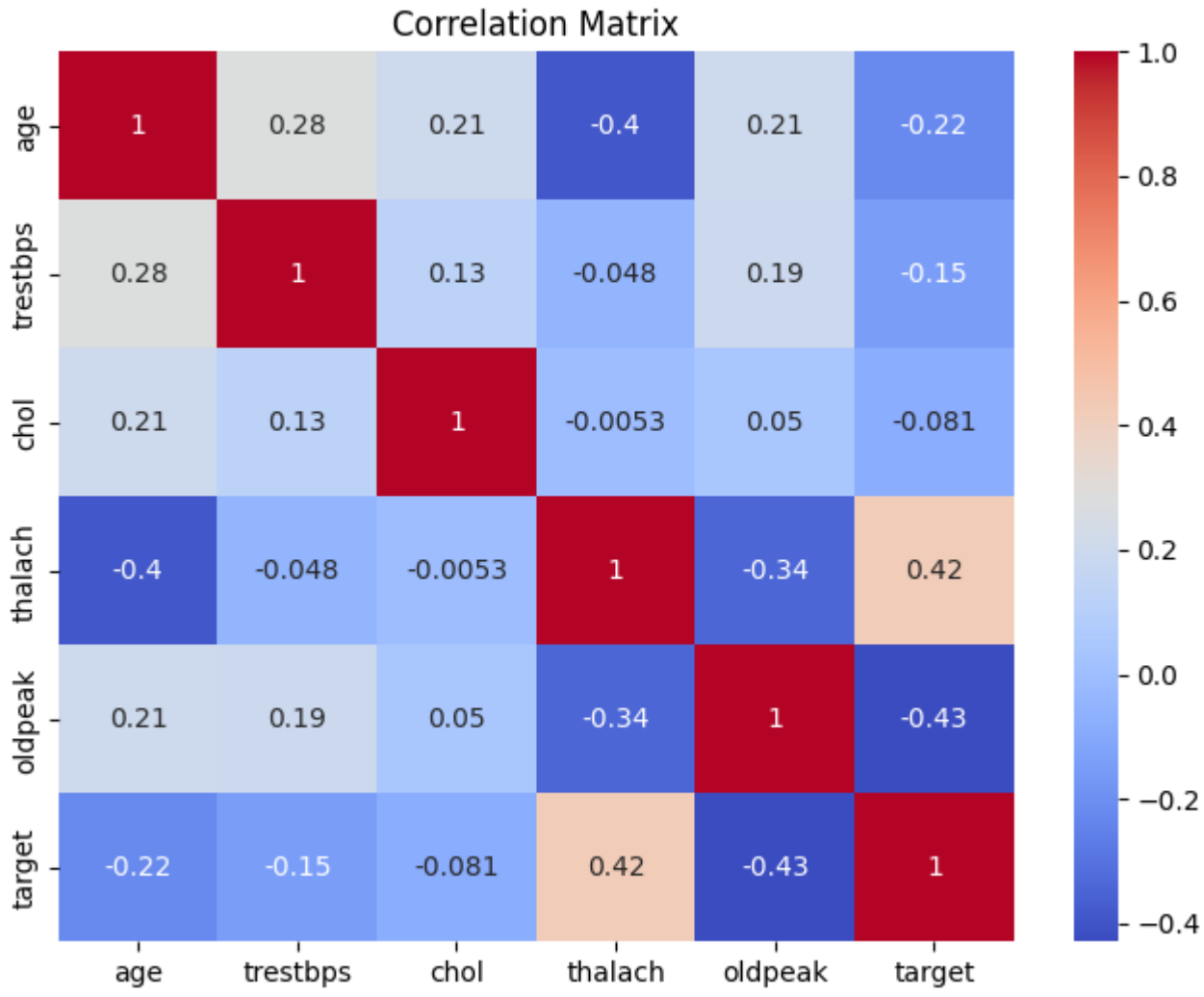
4.11 Full Correlation Heatmap

A correlation heatmap was used to examine how strongly each feature relates to all others, including the target variable. The heatmap highlights meaningful patterns, such as the positive correlation between chest pain type and heart disease, the negative correlation between maximum heart rate and age, and the strong association between oldpeak and slope. These correlations help identify which features carry greater predictive weight.



4.12 Reduced Correlation Heatmap (Key Predictors Only)

A reduced correlation heatmap focusing on age, resting blood pressure, cholesterol, maximum heart rate, oldpeak, and the target variable was generated for clearer interpretation. This condensed version reveals that oldpeak and maximum heart rate have the strongest correlations with the presence of heart disease, while cholesterol and resting blood pressure show weaker relationships. This simplified matrix helps guide feature importance and model selection.



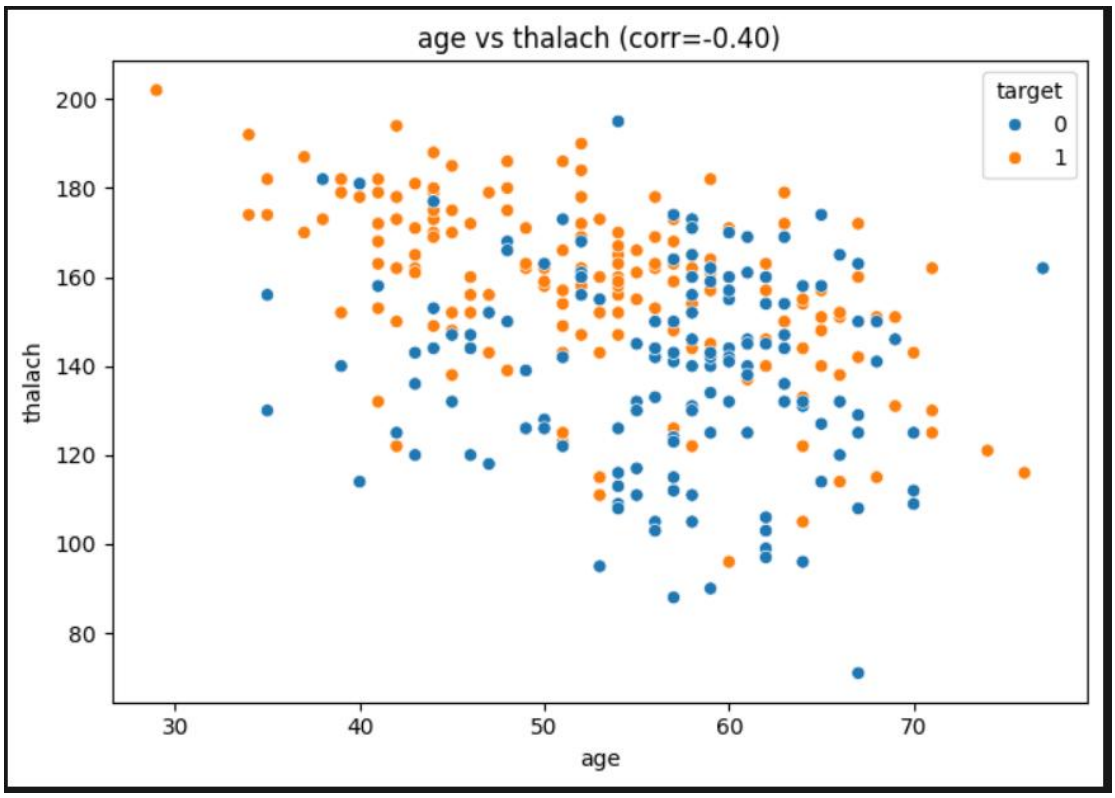
4.13 Top Correlated Features

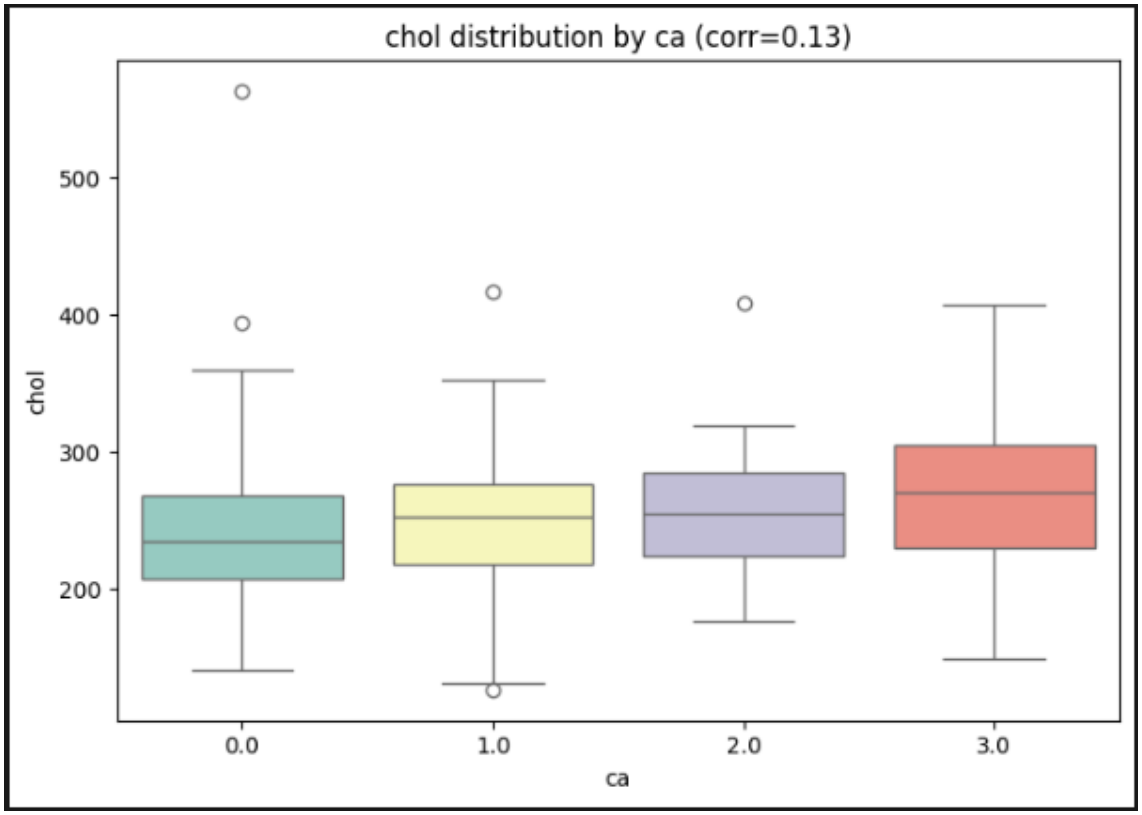
We calculated the top five strongest correlations for each feature to see which variables are most closely related. This helps highlight which measurements move together, such as age being strongly linked to maximum heart rate and chest pain type showing a strong connection with the target.

```
Top 5 strongest correlations (positive or negative) with 'age':
thalach    -0.395235
ca          0.367313
trestbps    0.283121
target     -0.221476
chol        0.207216
Name: age, dtype: float64

Top 5 strongest correlations (positive or negative) with 'sex':
target    -0.283609
thal       0.214220
chol      -0.195571
exang      0.143460
oldpeak    0.098322
Name: sex, dtype: float64

Top 5 strongest correlations (positive or negative) with 'cp':
target     0.432080
exang     -0.392937
thalach    0.293367
ca        -0.226091
thal      -0.165391
Name: cp, dtype: float64
...
cp         0.432080
oldpeak   -0.429146
thalach    0.419955
Name: target, dtype: float64
```

4.14 Visualizing Feature Relationships

For each top correlation, we generated the appropriate plot: scatterplots for numerical pairs, boxplots for numerical vs categorical features, and countplots for categorical comparisons. These charts make it easy to visually interpret how features relate to each other and how the target influences these relationships.





5. Decision Tree Classifier

5.1 Description of the Model

A Decision Tree Classifier is a supervised learning algorithm that predicts the target variable by splitting the dataset into branches based on feature conditions.
In this experiment, the model was trained using the Entropy criterion, meaning it selects splits that maximize information gain. Limiting the maximum depth and minimum leaf sizes helps prevent overfitting.

The model outputs a human-interpretable structure showing how predictions are made.

5.2 Applied Tools / Techniques

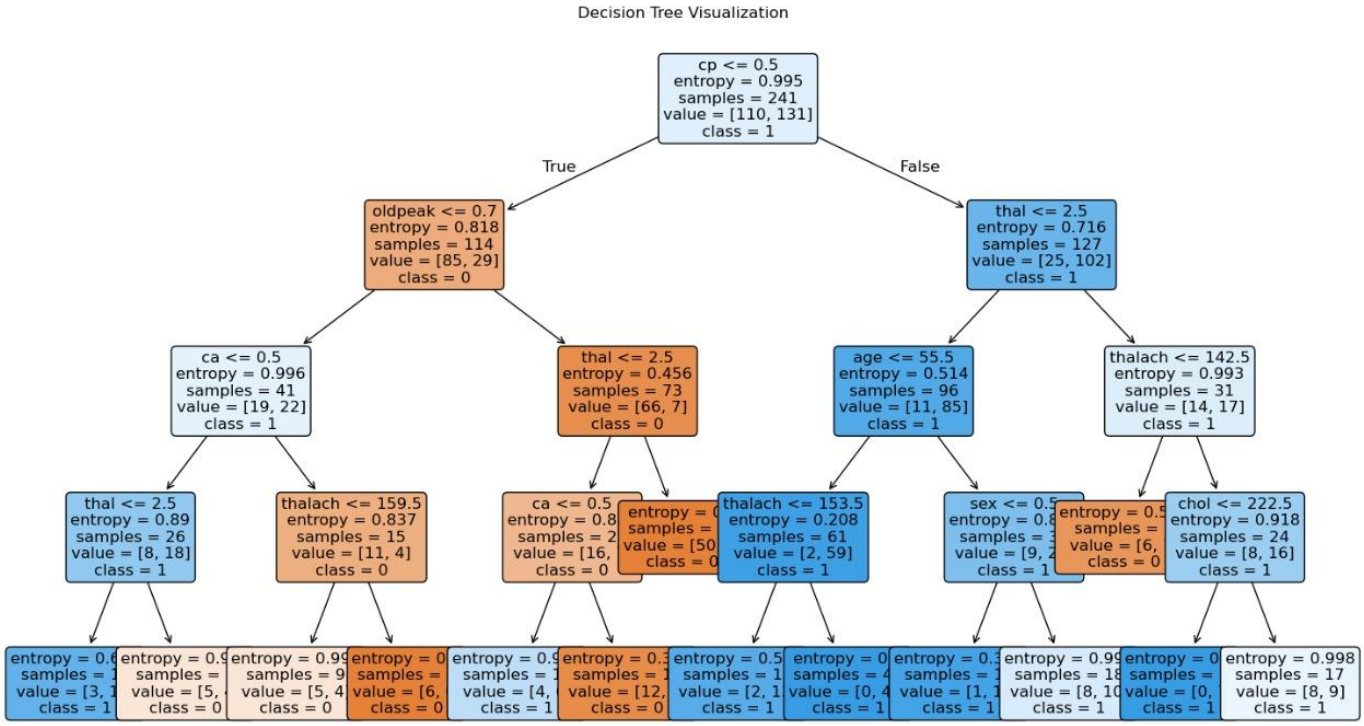
Your Decision Tree experiment used the following ML components:

- DecisionTreeClassifier with:
    - criterion='entropy'
    - max_depth=4
    - min_samples_leaf=5
    - min_samples_split=2
- Train/Test Split (80/20)
- Model Evaluation
    - Accuracy (Train & Test)
    - Confusion Matrix
    - Classification Report
- Model Visualization
    - Full Decision Tree Graph
- Cross-Validation
    - 10-fold CV
    - Average accuracy & standard deviation
    - Boxplot of CV accuracy

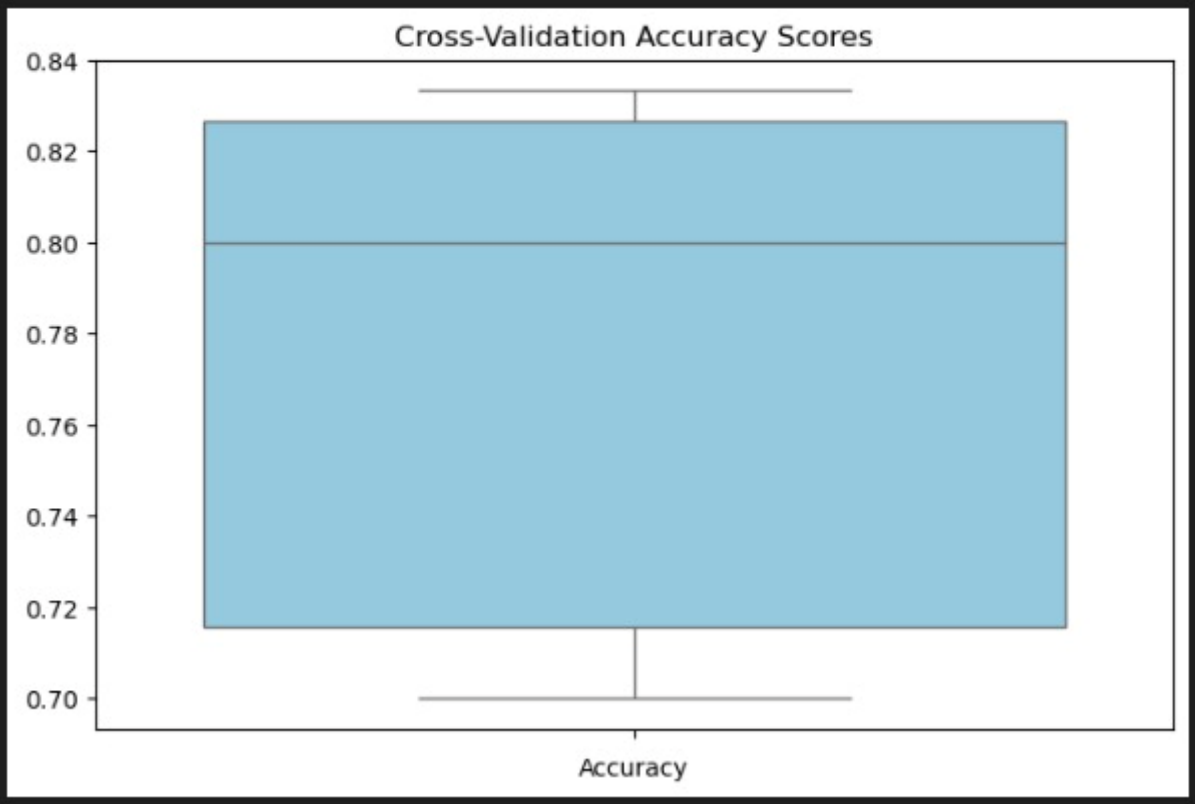
5.3 Used Libraries

- pandas – data handling and manipulation
- matplotlib / seaborn – visualization
- scikit-learn – model training, evaluation, and cross-validation

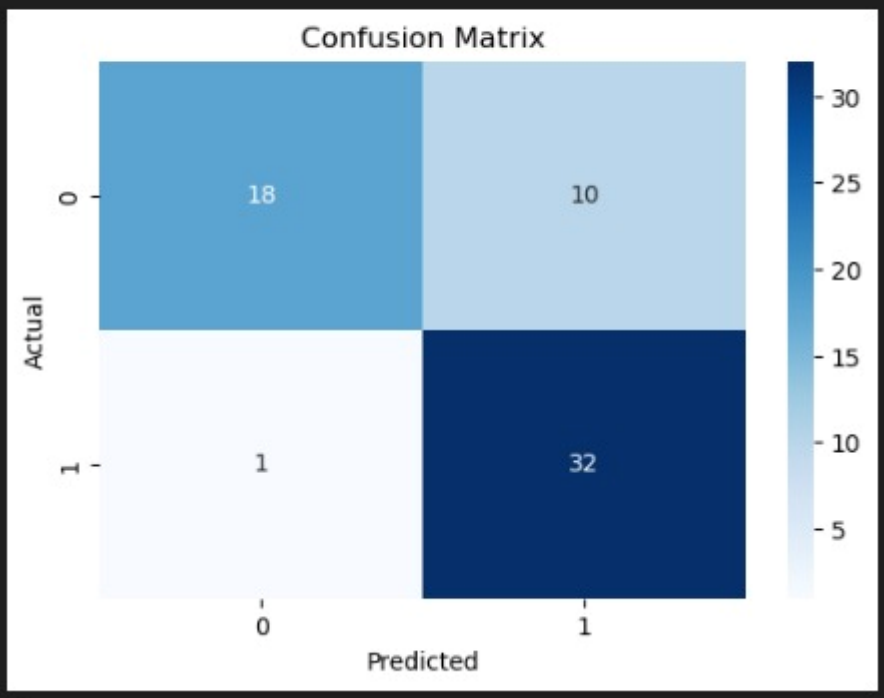
5.4 Results

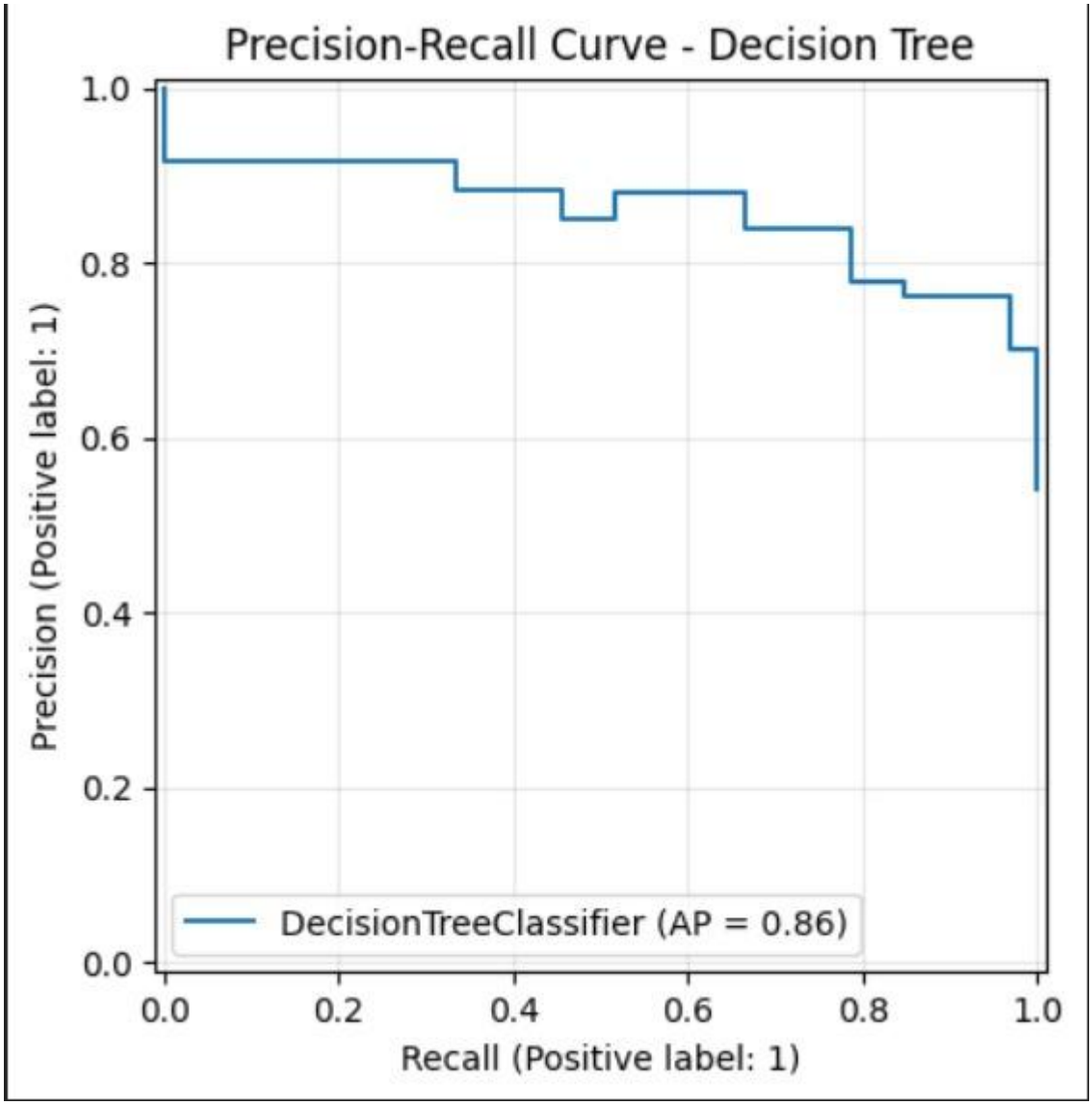


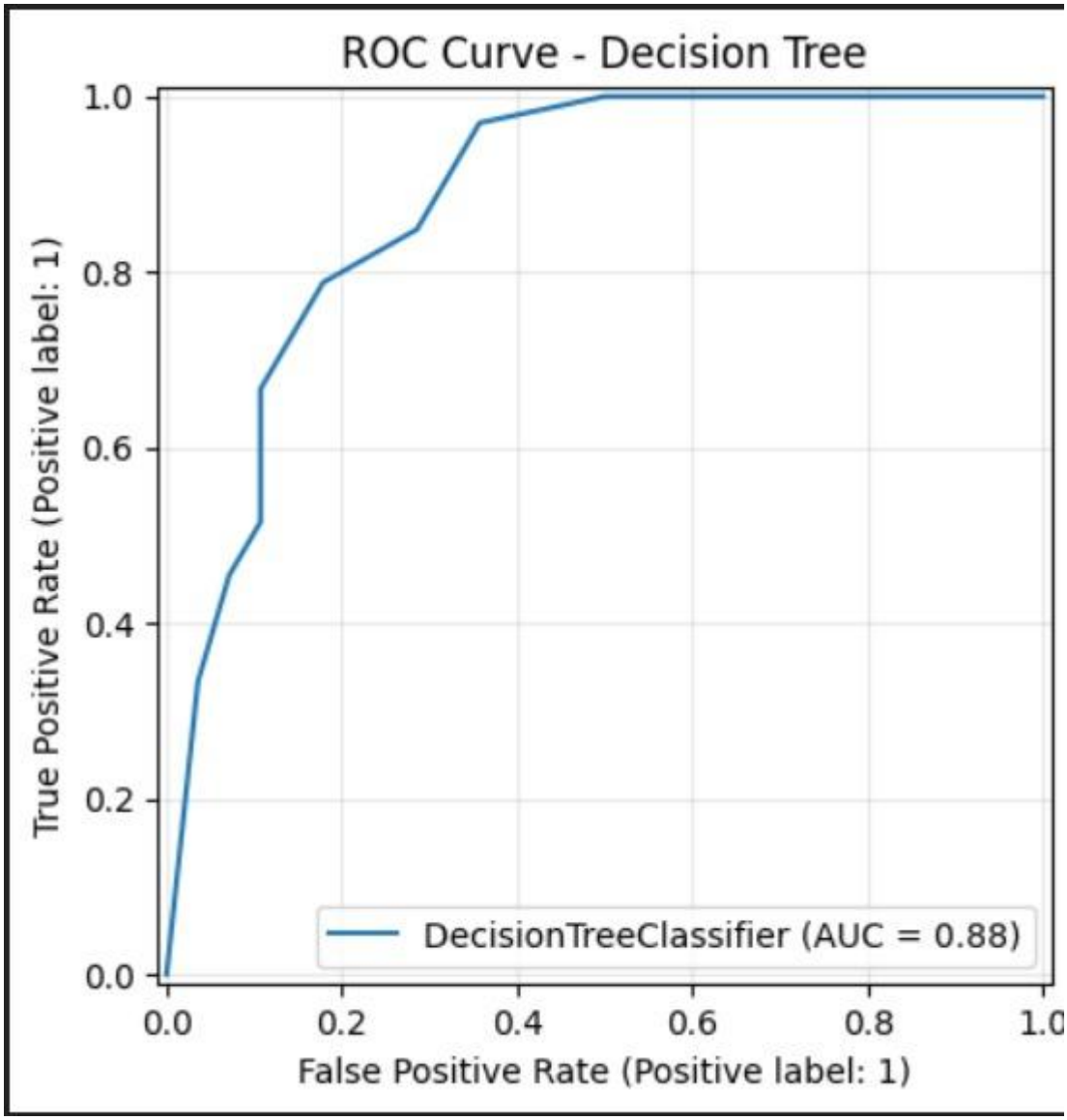

```
=== CROSS VALIDATION RESULTS ===
CV Scores: [0.8065, 0.7097, 0.8333, 0.8333, 0.8, 0.7, 0.7333, 0.8, 0.7, 0.8333]
Mean CV Accuracy: 0.7749
Std Deviation: 0.0545
```

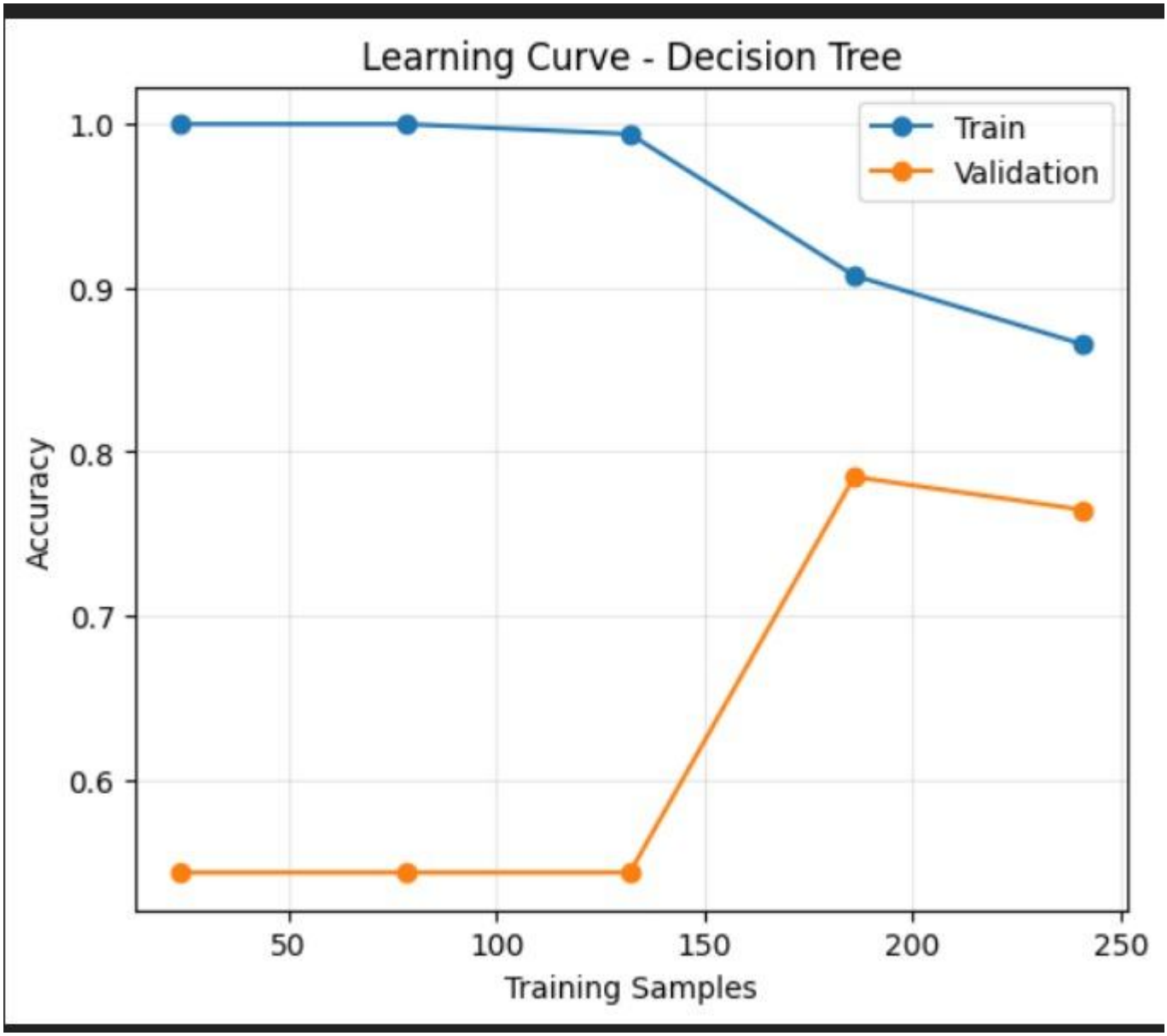


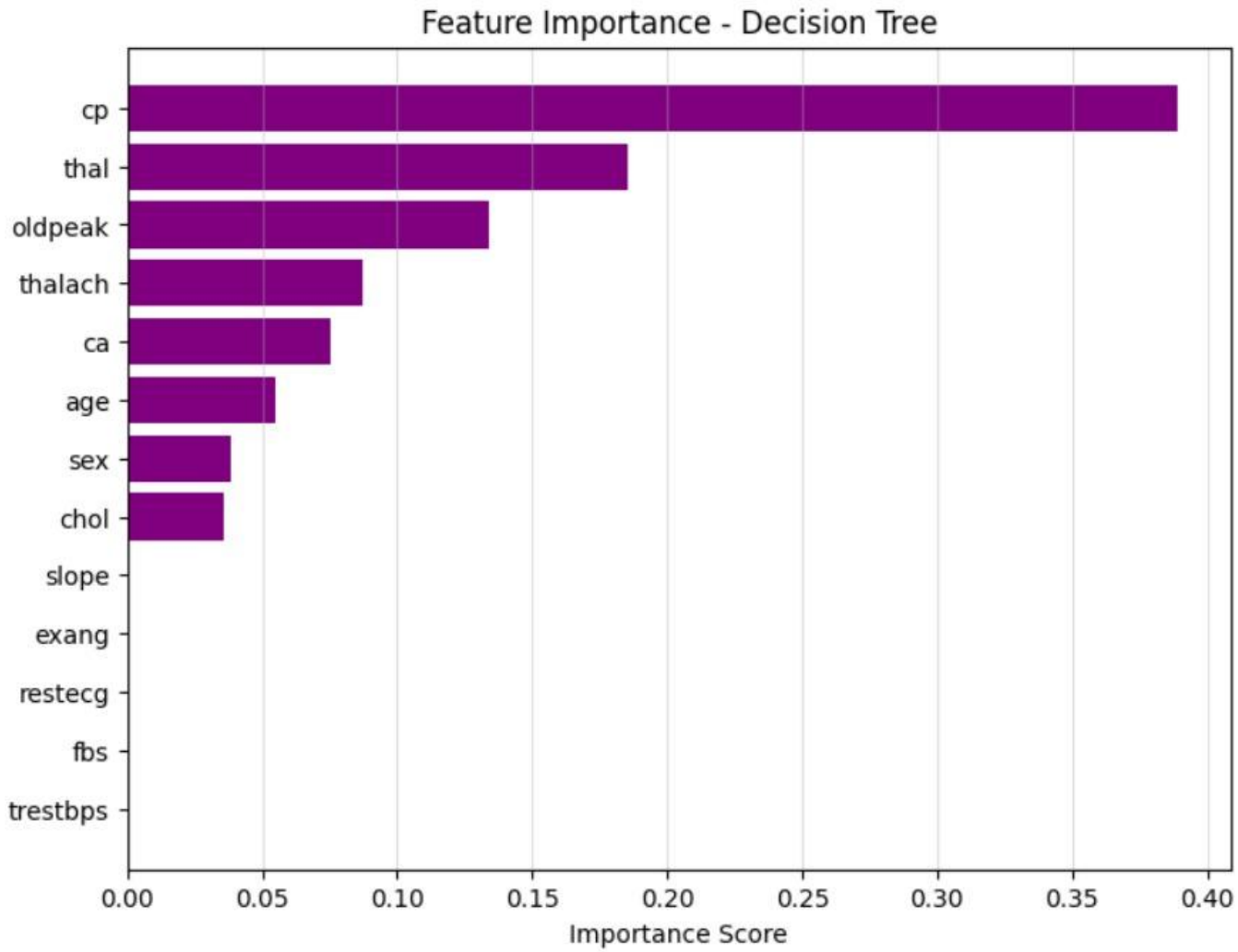
```
=== MODEL EVALUATION ===
Train Accuracy: 0.8506
Test Accuracy: 0.8197
```




Precision-Recall Curve - Decision Tree
Precision (Positive label: 1)
Recall (Positive label: 1)
DecisionTreeClassifier (AP = 0.86)
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve - Decision Tree
True Positive Rate (Positive label: 1)
False Positive Rate (Positive label: 1)
DecisionTreeClassifier (AUC = 0.88)
0.0
0.2
0.4
0.6
0.8
1.0


Learning Curve - Decision Tree
Train
Validation
1.0
0.9
0.8
0.7
0.6
Accuracy
50
100
150
200
250
Training Samples



6. Naive Bayes Classifier

6.1 Description of the Model

Naive Bayes is a probabilistic classification algorithm based on **Bayes' Theorem**, assuming that all features are conditionally independent given the target class.
Despite this "naive" assumption, it performs remarkably well on many real-world datasets due to its simplicity, speed, and ability to handle noisy data.

In this experiment, the **Gaussian Naive Bayes** variant was used, which models the likelihood of continuous features using a normal distribution. The model outputs class probabilities and assigns each sample to the class with the highest posterior probability.

6.2 Applied Tools / Techniques

The Naive Bayes experiment incorporated a complete preprocessing and modeling workflow, including:

Outlier Detection & Cleaning

- IQR Method to identify extreme values in numerical features.
- Isolation Forest (contamination = 0.05) to detect multivariate anomalies.
- Intersection of IQR and Isolation Forest outliers removed to ensure robust cleaning.

Data Preprocessing

- Winsorization of numerical features to reduce the impact of extreme values by clipping the top and bottom 1%.
- Categorical Fixing Function to standardize category values in categorical features.
- ColumnTransformer applying:
- PowerTransformer (Yeo-Johnson) to normalize numerical distributions.
- StandardScaler for feature scaling.
- Pipeline to combine preprocessing and model training into a single workflow.

Model Training

- Gaussian Naive Bayes classifier (GaussianNB) with:
  - priors = [0.4, 0.6]
  - var_smoothing = 1e-5

Train/Test Split

- 80/20 split with stratification to preserve class balance.

Model Evaluation

- Training and testing accuracy
- Confusion matrix (visualized using seaborn heatmap)
- Classification report (precision, recall, F1-score)
- ROC AUC Score for probability-based performance evaluation

Cross-Validation

- 5-fold cross-validation using KFold (shuffled with random_state=42)
- Outputs:
  - Fold accuracy scores
  - Mean accuracy
  - Standard deviation

6.3 Used Libraries

Data Handling & Processing

- pandas – dataset manipulation and cleaning
- numpy – numerical operations and winsorization

Visualization

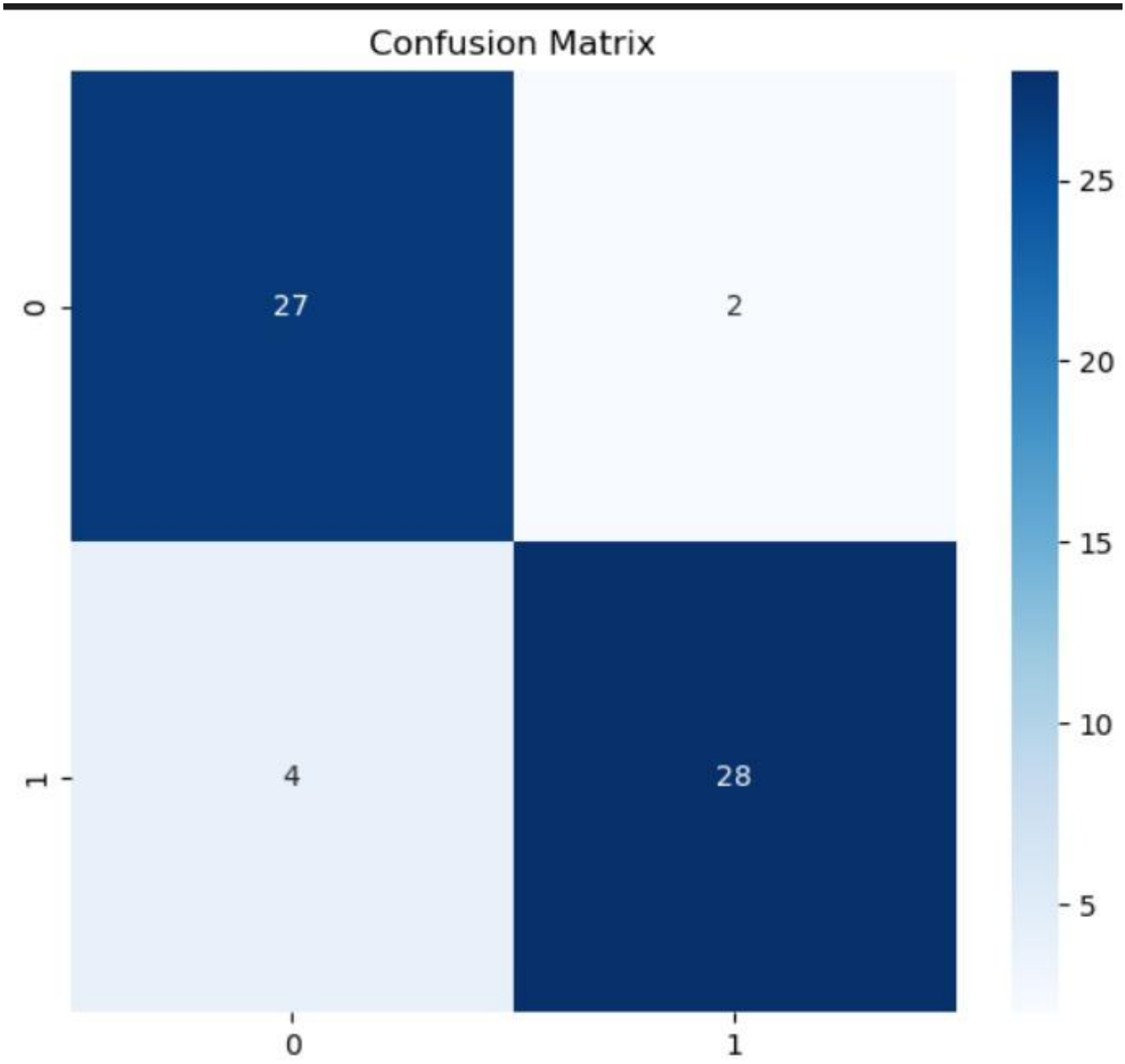
- matplotlib.pyplot – plots for evaluation
- seaborn – heatmap visualizations

Machine Learning (scikit-learn)

- model_selection
  - train_test_split – split data into training and testing sets
  - cross_val_score – compute cross-validation accuracy
  - KFold – custom 5-fold CV configuration
- preprocessing
  - PowerTransformer – normalize numerical features
  - StandardScaler – scale numerical data
- compose
  - ColumnTransformer – apply transformations to specific feature groups
- pipeline
  - Pipeline – build preprocessing + model training workflow
- ensemble
  - IsolationForest – detect anomalous samples
- metrics
  - accuracy_score, classification_report, confusion_matrix, roc_auc_score
- naive_bayes
  - GaussianNB – Naive Bayes classification model

6.4 Results

```
============================
 Model: Naive Bayes
============================

► CV Accuracy (5 folds): 0.8375
► Fold scores : [0.90163934 0.83606557 0.88333333 0.75       0.81666667]
► Train Accuracy : 0.8216
► Test Accuracy  : 0.9016
► ROC AUC        : 0.9138

Classification Report:
              precision    recall  f1-score   support

           0       0.87      0.93      0.90        29
           1       0.93      0.88      0.90        32

    accuracy                           0.90        61
   macro avg       0.90      0.90      0.90        61
weighted avg       0.90      0.90      0.90        61
```


Confusion Matrix
27
2
4
28
0
1
0
1
25
20
15
10
5

```
Outliers detected by IQR: 19
Outliers detected by IsolationForest: 16
Common Outliers removed: 10

============================
 Model: Gaussian Naive Bayes
============================

----------------------------
      MODEL PERFORMANCE
----------------------------
CV Accuracy: 0.8422
Train Acc:  0.8369
Test Acc:   0.8305
ROC AUC:    0.9114

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.77      0.80        26
           1       0.83      0.88      0.85        33

    accuracy                           0.83        59
   macro avg       0.83      0.82      0.83        59
weighted avg       0.83      0.83      0.83        59
```

```
Outliers detected by IQR: 19
Outliers detected by IsolationForest: 16
Common Outliers removed: 10

============================
 Model: Gaussian Naive Bayes
============================

----------------------------
      MODEL PERFORMANCE
----------------------------
CV Accuracy: 0.8387
Train Acc:  0.8498
► Fold scores : [0.84745763 0.91525424 0.77586207 0.82758621 0.82758621]
Test Acc:   0.8475
ROC AUC:    0.9200
```

7. K-Nearest Neighbors (KNN) Classifier

7.1 Description of the Model

The K-Nearest Neighbors (KNN) algorithm is a distance-based supervised learning method that classifies new observations by examining the classes of the *K* closest data points in the feature space.
Unlike other models, KNN does not build an internal model during training; instead, it stores the dataset and performs classification during prediction by computing distances.

For this experiment, the model used K = 5 neighbors, which is a commonly effective default. The algorithm predicts the class of a sample based on the majority vote among the nearest neighbors.

KNN is sensitive to the scale of features and the distribution of the dataset, but it is intuitive and performs well when decision boundaries are irregular.

7.2 Applied Tools / Techniques

The KNN experiment used the following machine learning components:

Data Preparation

- Removal of duplicate rows.
- Train/Test Split (80/20) with stratification to preserve class proportions.

Feature Scaling

- StandardScaler applied to all features to standardize numeric values before distance-based modeling.

Hyperparameter Tuning

- Evaluation of K = 1 to 20 using test accuracy scores.
- Visualization of K value vs accuracy to identify performance trends.
- Automatic selection of best K (highest accuracy).

Model Training

- Final model built using:
  - KNeighborsClassifier(n_neighbors = best_k)

Model Evaluation

- Training accuracy
- Testing accuracy
- Precision
- Recall

- F1-score
- Confusion matrix (heatmap)
- Classification report (precision, recall, F1-score, support)

Cross-Validation

Performed using 5-fold Stratified K-Fold, computing:

- Mean Accuracy (CV)
- Mean F1 Score (CV)
- Mean ROC-AUC Score (CV)

Visualization

- Line plot: K value vs Test Accuracy
- Confusion Matrix heatmap

7.3 Used Libraries

The following Python libraries were used in the KNN analysis:

Data Handling

- pandas – dataset manipulation
- numpy – numeric operations

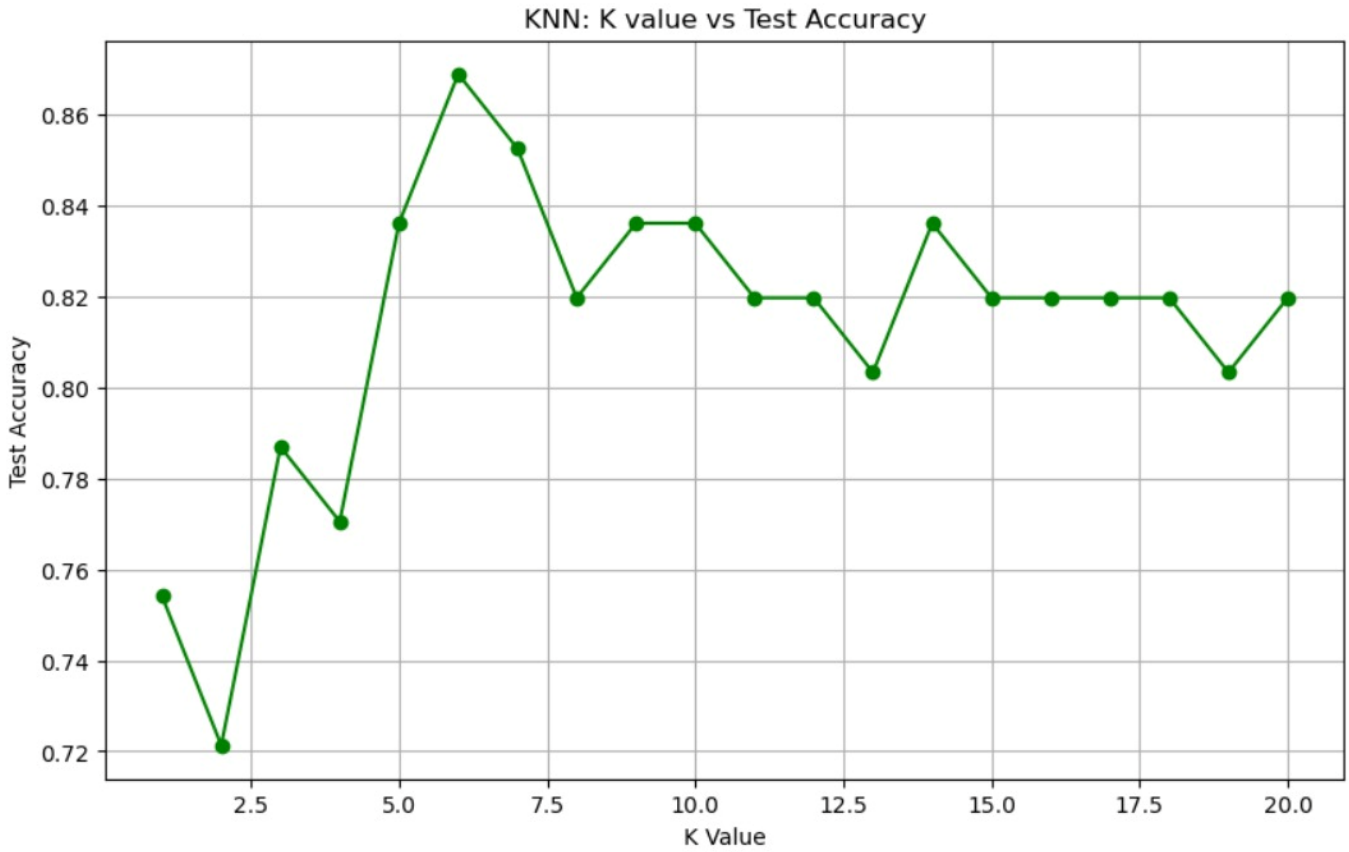
Visualization

- matplotlib.pyplot – plotting accuracy curves and confusion matrix
- seaborn – heatmap visualization

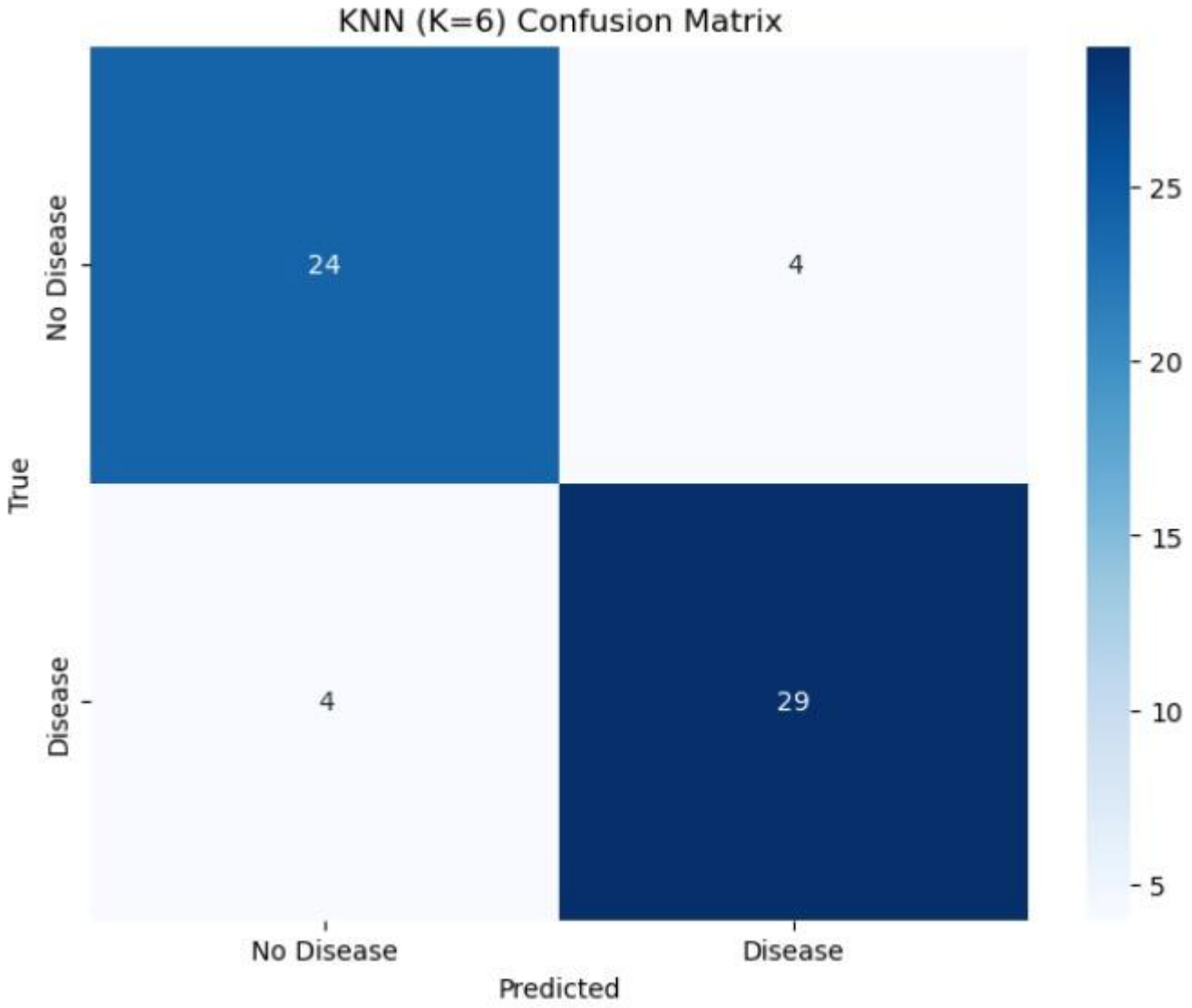
Machine Learning (scikit-learn)

- train_test_split – creating training and test sets
- StratifiedKFold – cross-validation with preserved class distribution
- cross_val_score – computing CV metrics
- StandardScaler – feature scaling
- KNeighborsClassifier – KNN model
- Evaluation utilities:
  - accuracy_score
  - precision_score
  - recall_score
  - f1_score
  - confusion_matrix
  - classification_report

7.4 Results




```
K-Nearest Neighbors Performance Metrics:

Accuracy Score:  0.8689
Precision Score: 0.8788
Recall Score:    0.8788
F1 Score:        0.8788

Train Accuracy: 0.8755
Test Accuracy : 0.8689
Accuracy (CV): 0.8040
F1 (CV):       0.8252
ROC-AUC (CV):  0.8691
```



8. Logistic Regression

8.1 Description of the Model

Logistic Regression is a widely used supervised learning algorithm designed for binary classification problems.
Instead of predicting continuous values, it estimates the probability that a given input belongs to a particular class using the sigmoid (logistic) function.

The model assumes a linear relationship between the features and the log-odds of the target variable.
Despite its simplicity, Logistic Regression performs exceptionally well when the data is linearly separable and is often used as a strong baseline model.

In this experiment, the model was trained using the default *regularized* implementation provided by scikit-learn to ensure numerical stability and prevent overfitting.

8.2 Applied Tools / Techniques

The Logistic Regression experiment used the following machine learning components:

Data Preparation

- Removal and correction of categorical outliers using a custom fix_cat() function
- Treatment of numeric outliers using IQR clipping, applied selectively based on outlier masks
- Evaluation of multiple outlier detection strategies:
  - IQR mask
  - Isolation Forest
  - Local Outlier Factor (LOF)
  - DBSCAN
  - Combined outlier mask (intersection of all methods)
- Creation of multiple datasets to compare model performance under different outlier-handling techniques

Feature Preprocessing (Pipeline)

- StandardScaler applied to numerical features
- OneHotEncoder applied to categorical features
- Combined using a ColumnTransformer
- Integrated into a full Pipeline that performs preprocessing + modeling in one object

Model Training

- Logistic Regression classifier configured with:
  - max_iter = 1000
  - solver = "liblinear" (stable for small datasets and binary classification)

Train/Test Split

- 80/20 split
- Stratified sampling to maintain class balance
- random_state = 42 for reproducibility

Model Evaluation

For each preprocessing strategy, the following metrics were calculated:

- Training Accuracy
- Testing Accuracy
- Classification Report (precision, recall, F1-score)
- ROC-AUC (Cross-Validation)
- Accuracy (Cross-Validation)
- F1 (Cross-Validation)

Cross-Validation

- 5-fold Stratified K-Fold CV using:
  - Accuracy
  - F1-score
  - ROC-AUC score
- Results compared across all outlier handling methods to identify the best performer

8.3 Used Libraries

Data Handling

- pandas – dataset manipulation
- numpy – numeric operations

Preprocessing

- StandardScaler – numeric scaling
- OneHotEncoder – categorical encoding
- ColumnTransformer – apply transformations to specific column groups
- Pipeline – build multi-step preprocessing + modeling pipeline

Machine Learning

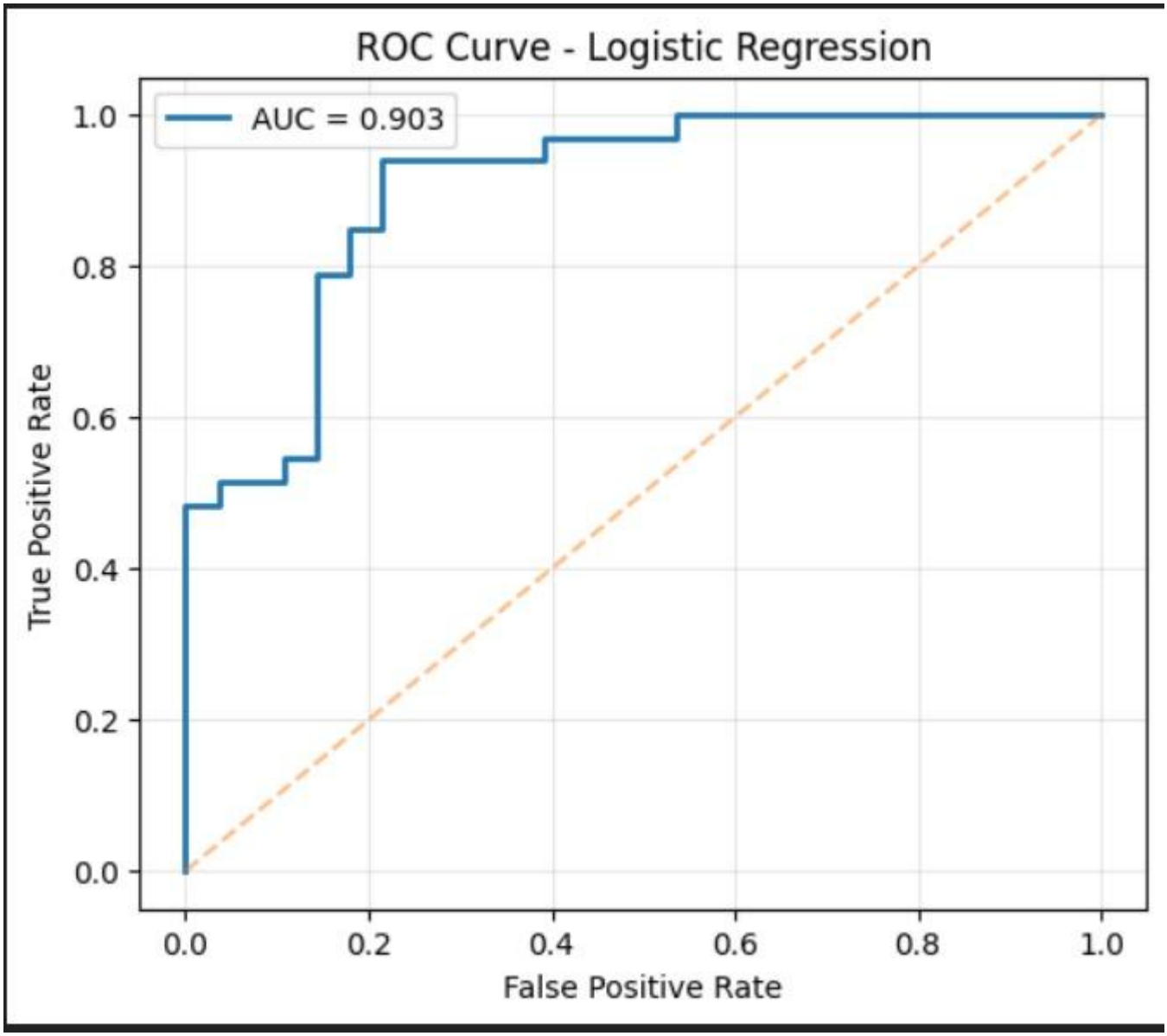
- LogisticRegression – main classification model
- train_test_split – dataset splitting
- StratifiedKFold – stratified cross-validation
- cross_val_score – CV scoring

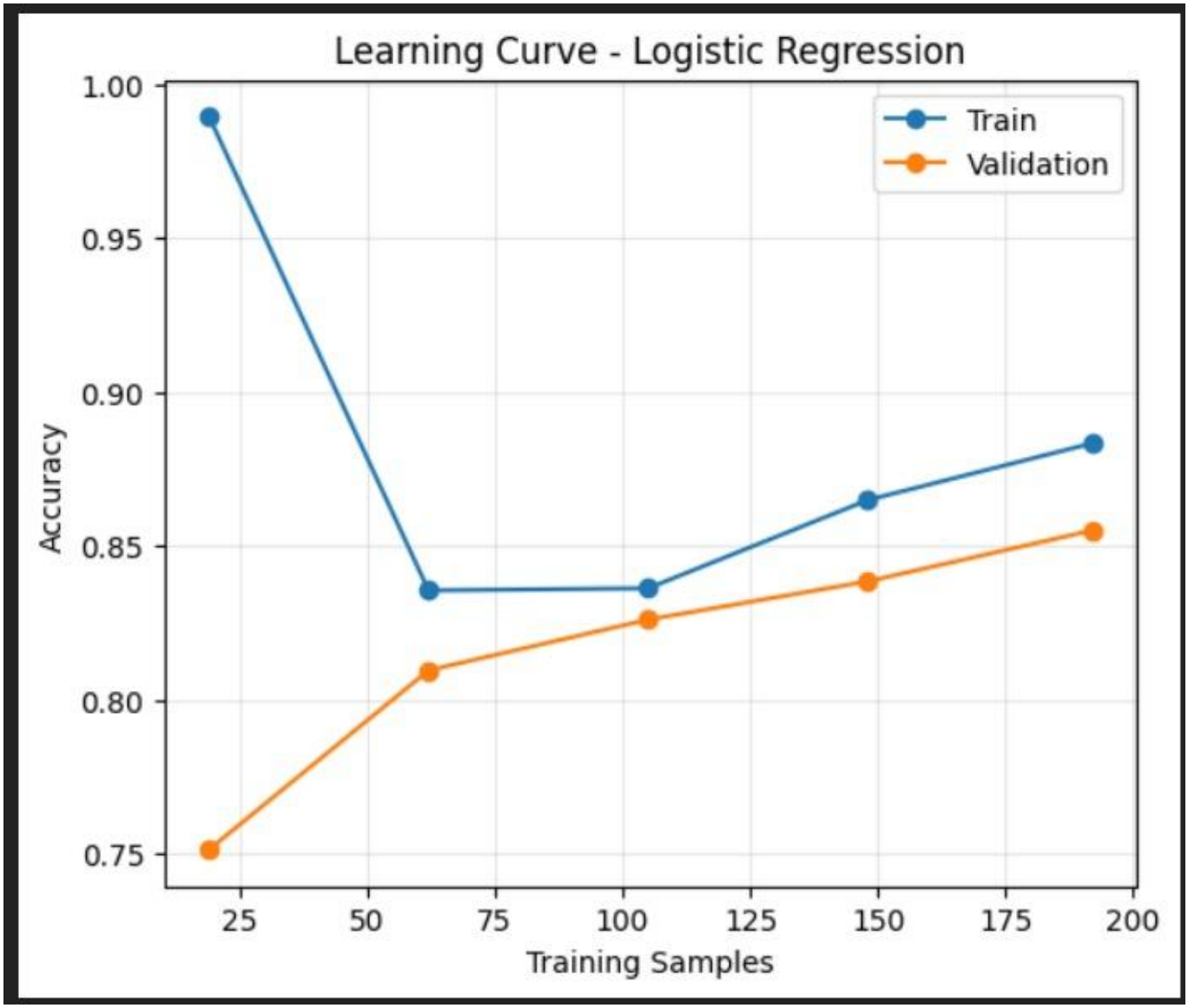
Evaluation Metrics

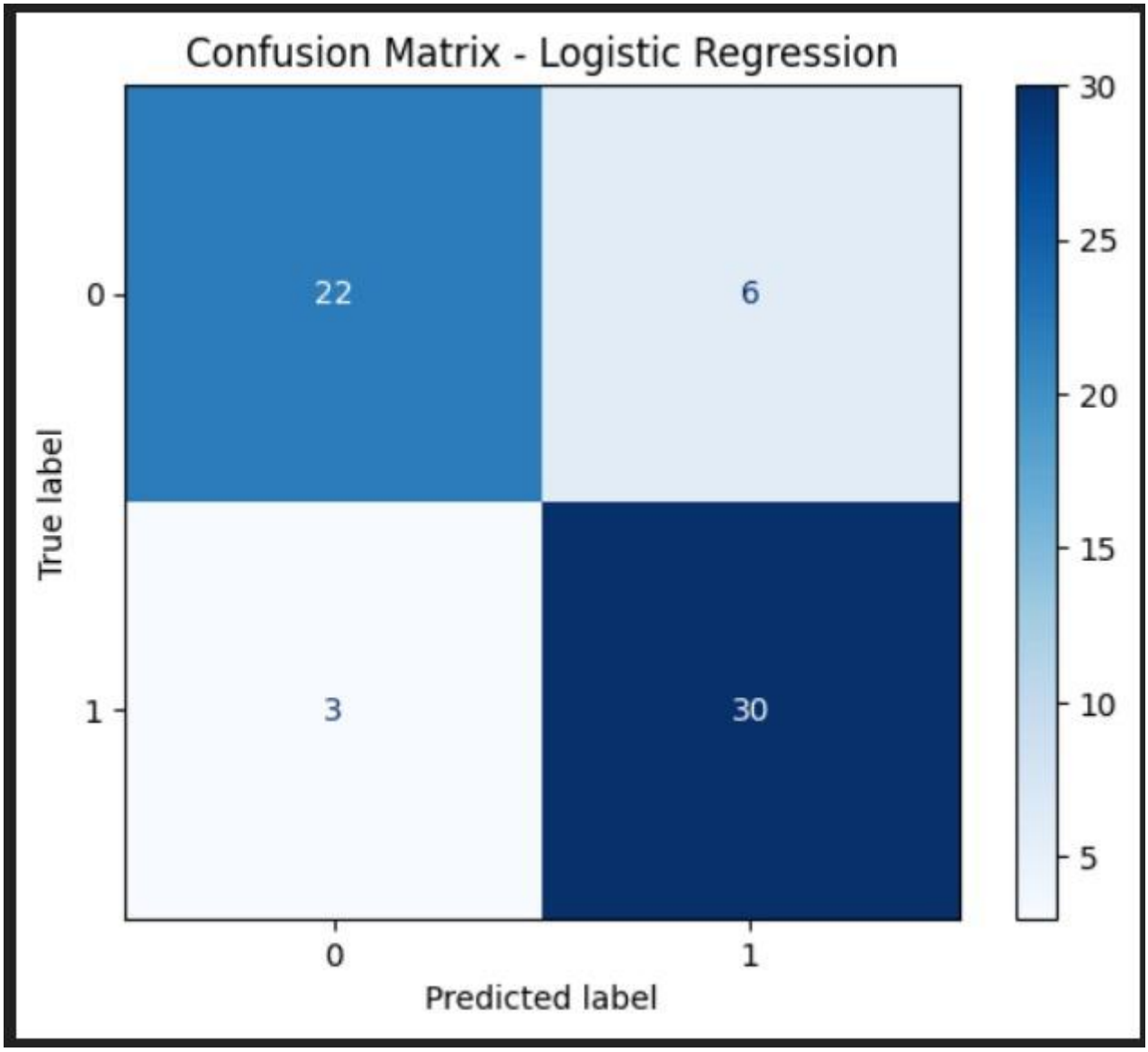
- Accuracy
- F1-score
- ROC-AUC score
- Classification report
- Confusion matrix visualization (when applied externally)

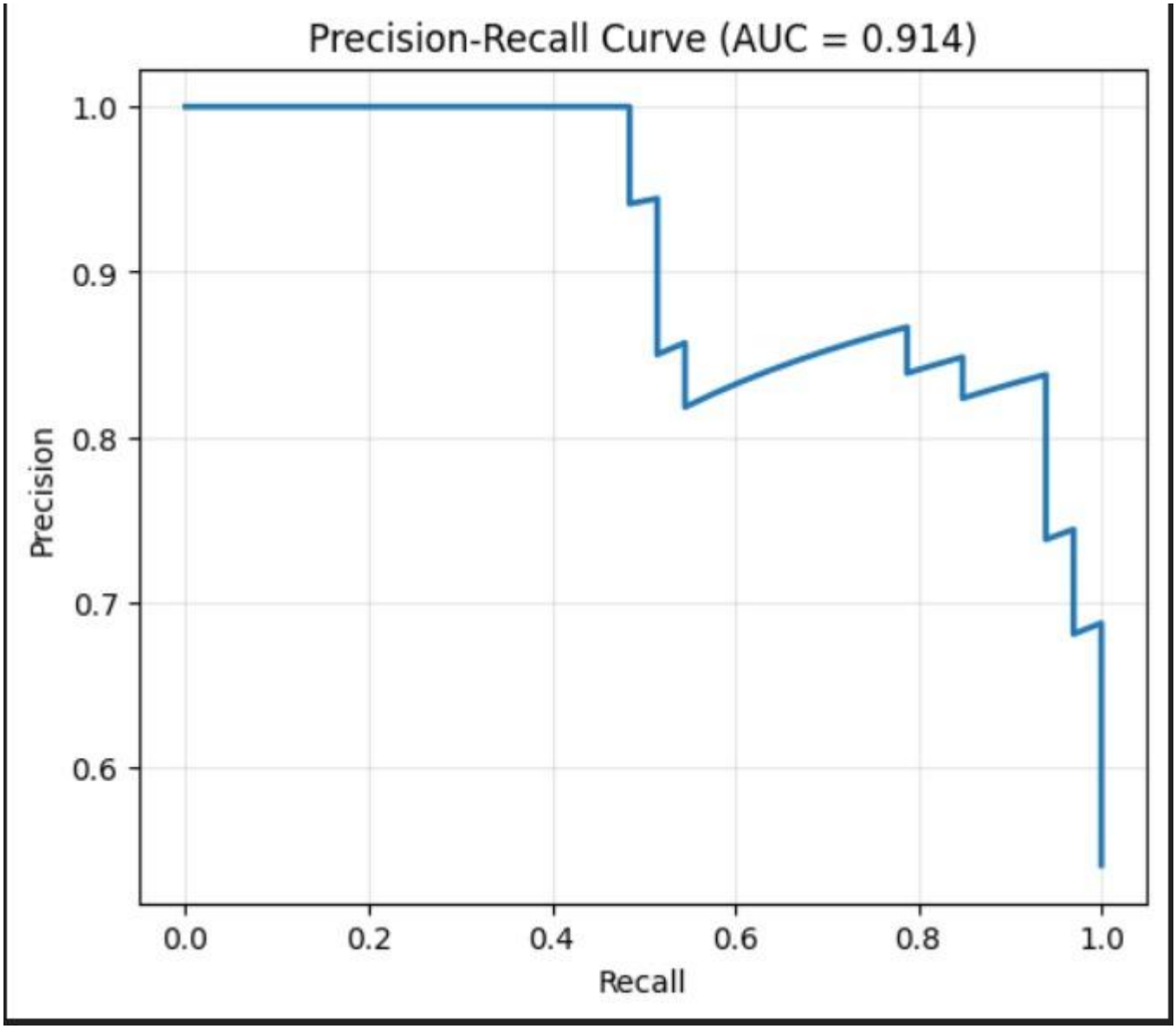
8.4 Results

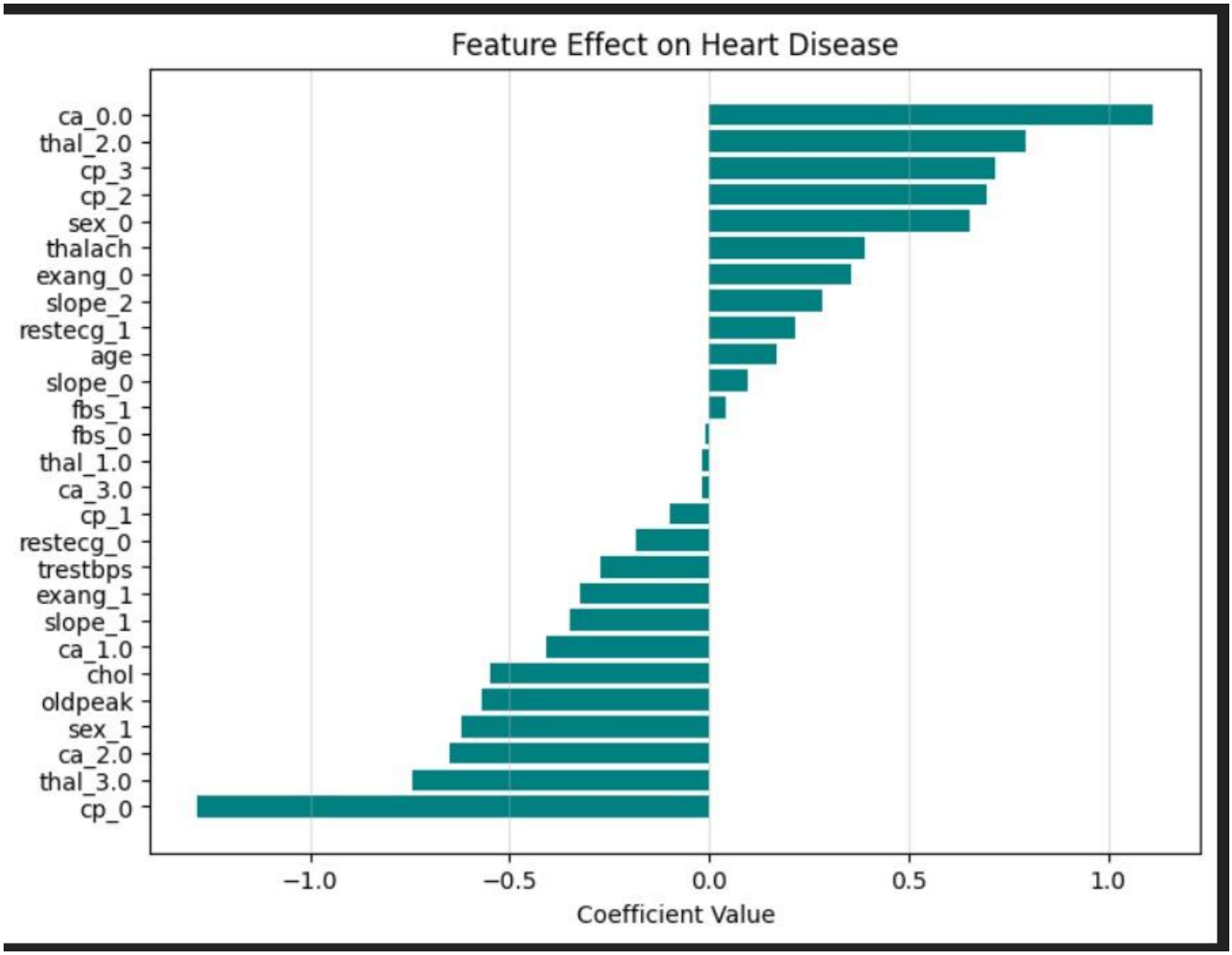
```
Numeric features: ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
Categorical features: ['sex', 'cp', 'fbs', 'restecg', 'exang', 'slope', 'ca', 'thal']

  Comparison:
                Train Accuracy  Test Accuracy  Accuracy (CV)  ROC-AUC (CV)  \
Baseline              0.871369       0.836066       0.860710      0.916457
IQR                   0.871369       0.852459       0.860765      0.916674
IsolationForest       0.871369       0.852459       0.864044      0.916017
LOF                   0.871369       0.852459       0.860765      0.916464
DBSCAN                0.871369       0.852459       0.864044      0.916017
Common_All            0.871369       0.852459       0.864044      0.916234

                 F1 (CV)
Baseline        0.878646
IQR             0.878025
IsolationForest 0.881289
LOF             0.878025
DBSCAN          0.881289
Common_All      0.881289

 Best method: IQR
Train Accuracy    0.871369
Test Accuracy     0.852459
Accuracy (CV)     0.860765
ROC-AUC (CV)      0.916674
F1 (CV)           0.878025
Name: IQR, dtype: float64
```


ROC Curve - Logistic Regression
AUC = 0.903
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0


Learning Curve - Logistic Regression
Train
Validation
Accuracy
1.00
0.95
0.90
0.85
0.80
0.75
25
50
75
100
125
150
175
200
Training Samples


Confusion Matrix - Logistic Regression
22
6
3
30
0
1
True label
0
1
Predicted label
30
25
20
15
10
5


Precision-Recall Curve (AUC = 0.914)
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.0
0.2
0.4
0.6
0.8
1.0



9. Support Vector Machine (SVM)

9.1 Description of the Model

Support Vector Machine (SVM) is a powerful supervised learning algorithm used for classification tasks.
SVM works by identifying the optimal hyperplane that separates classes with the maximum margin, making it highly effective for datasets with clear decision boundaries.

In this experiment, the model used the RBF (Radial Basis Function) kernel, which is well-suited for capturing non-linear relationships in the data.
SVM is also robust to high-dimensional feature spaces and often provides strong performance even without extensive parameter tuning.

9.2 Applied Tools / Techniques

The SVM experiment utilized the following components:

Outlier Handling

- Use of IQR flags, Isolation Forest, LOF, and DBSCAN outlier indicators
- Custom row-wise clipping for numerical features where:
  - the row was flagged by IQR and
  - the row had ≥ 3 outlier votes
- Selective clipping applied only to features that were IQR outliers for each specific row

Data Splitting

- Train/Validation/Test split: 70% / 10% / 20%
  - Achieved using two stratified train_test_split() operations
- Stratification ensures balanced class distribution across all sets

Feature Preprocessing Pipeline

Implemented using scikit-learn's Pipeline:

- RobustScaler()
  - Chosen for outlier-resistant scaling
- SVC(probability=True, random_state=42)
  - Probability enabled for ROC and threshold optimisation

Hyperparameter Tuning (Grid Search)

A comprehensive GridSearchCV optimization was performed over:

- Kernels: rbf, poly
- Regularization: C = [0.1, 1, 5, 10, 20, 50, 100]
- Gamma values: scale, auto, 0.1, 0.01, 0.001
- Polynomial degrees: 2, 3
- Class weighting: "balanced"

Using:

- 5-fold StratifiedKFold cross-validation
- Scoring metric: Accuracy

Threshold Optimization

Instead of using the default 0.5 decision threshold, the model:

- Computes ROC curve on the validation set
- Selects the threshold maximizing Youden's J statistic (TPR – FPR)

This improves recall and overall classification performance.

Model Evaluation

Metrics computed for train, validation, and test sets:

- Accuracy
- Precision

- Recall
- F1-score
- ROC-AUC

Visualizations include:

- ROC curve
- Precision–Recall curve
- Confusion Matrix (heatmap)

9.3 Used Libraries

Data Handling

- pandas – dataset preparation
- numpy – numerical operations

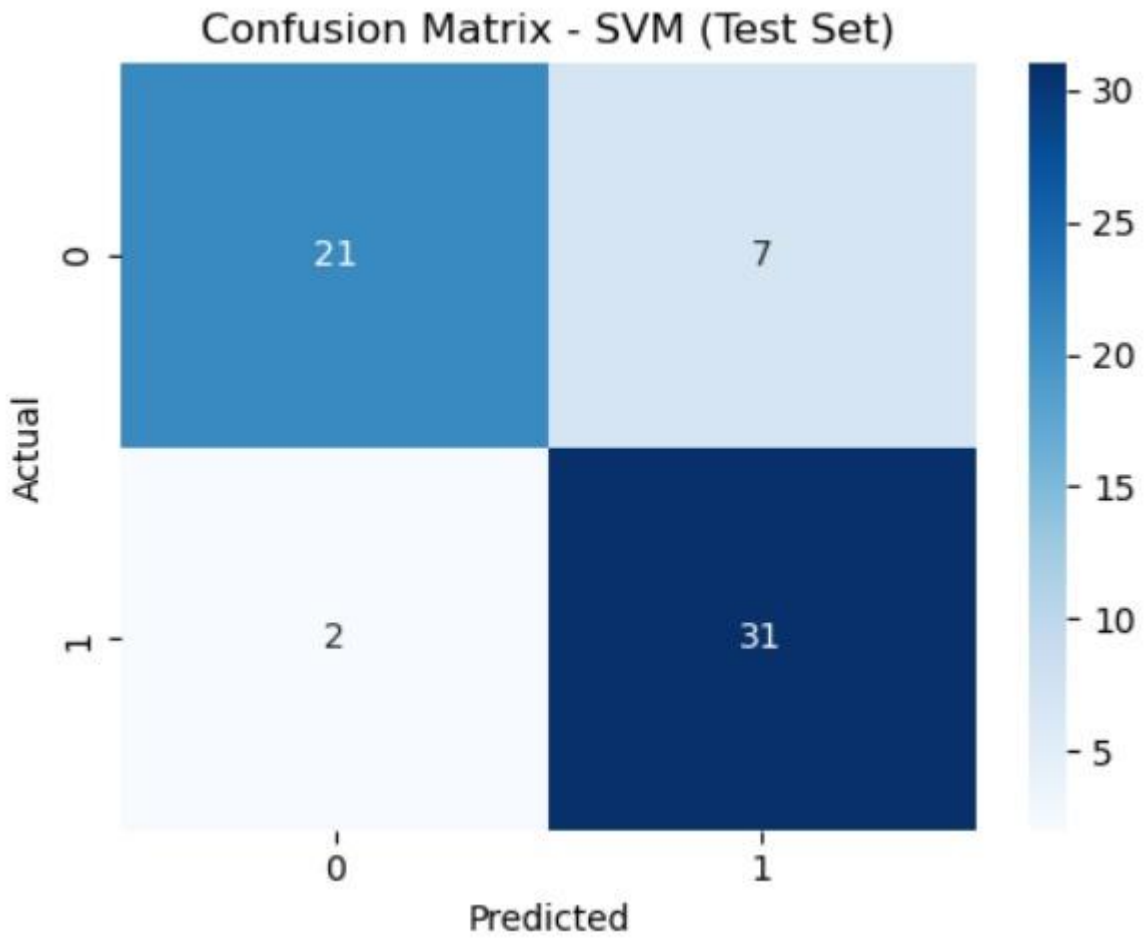
Machine Learning (scikit-learn)

- Model selection:
  - train_test_split
  - StratifiedKFold
  - GridSearchCV
- Preprocessing:
  - RobustScaler
  - ColumnTransformer (if used)
  - Pipeline
- Model:
  - SVC (Support Vector Classifier)
- Evaluation metrics:
  - accuracy_score, precision_score, recall_score, f1_score
  - confusion_matrix, classification_report
  - ROC & PR curve metrics: roc_curve, precision_recall_curve, auc

Visualization

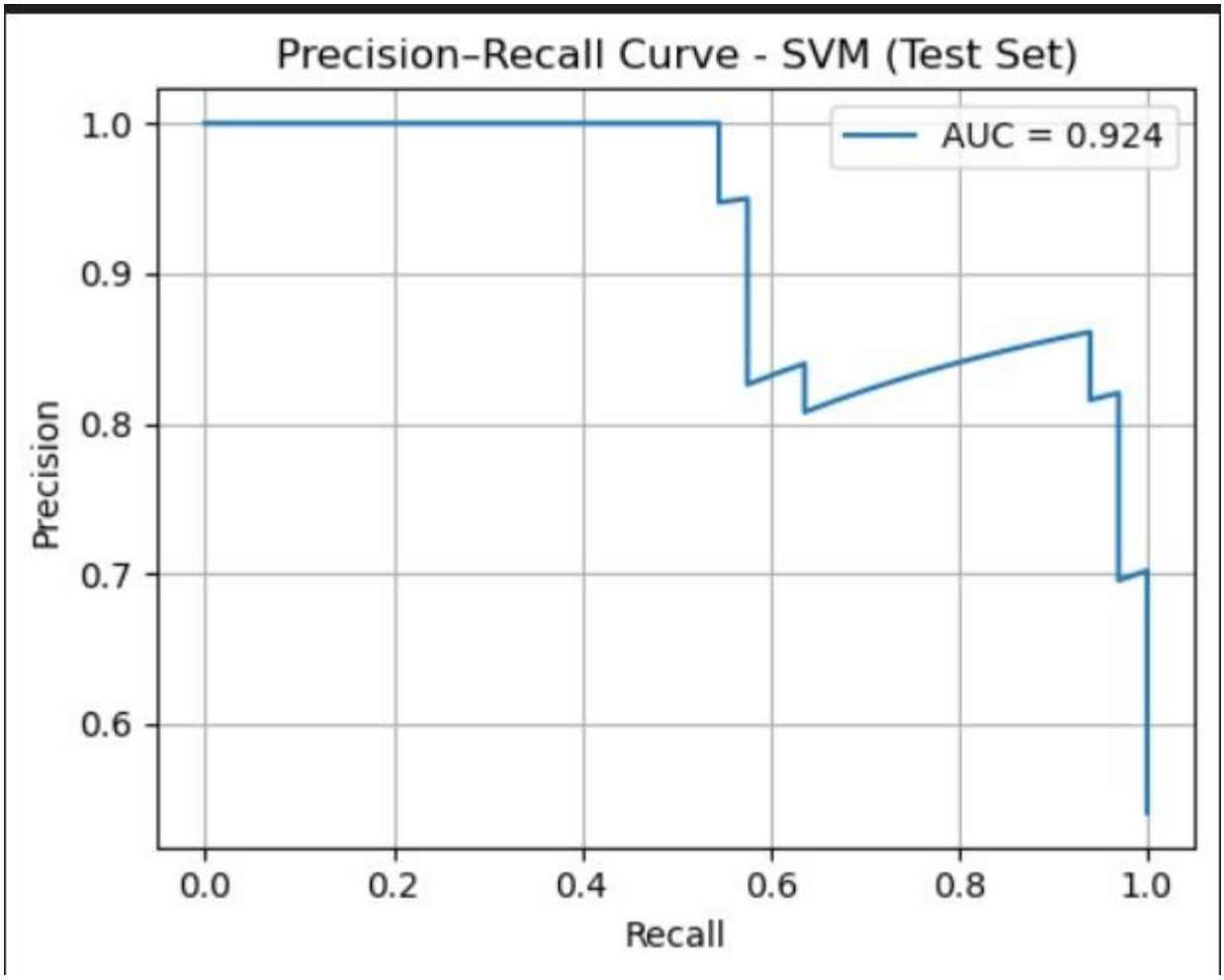
- matplotlib.pyplot – performance plots
- seaborn – confusion matrix heatmap

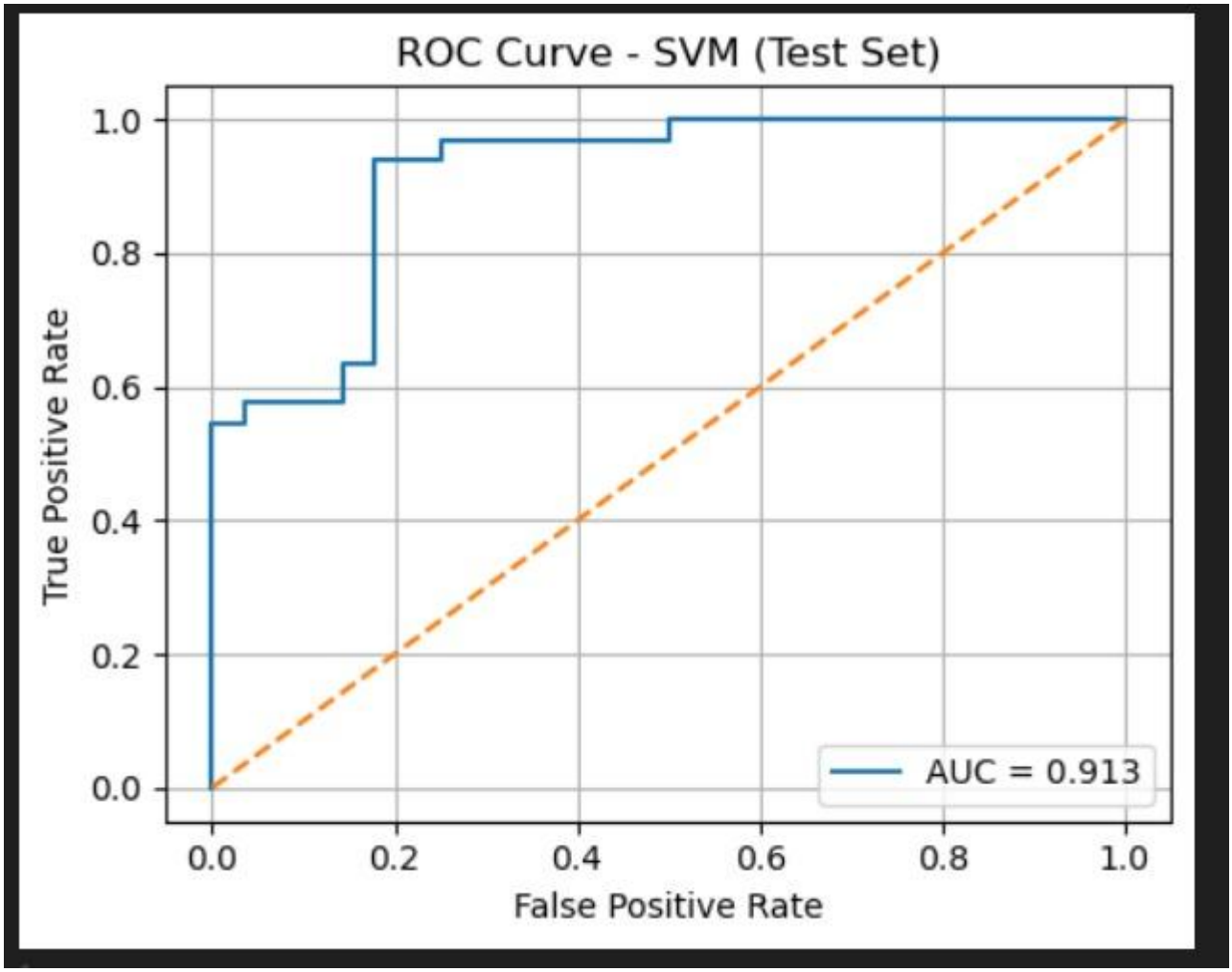
9.4 Results




```
...
    Classification Report (Test Set):

                  precision    recall  f1-score   support

             0.0       0.91      0.75      0.82        28
             1.0       0.82      0.94      0.87        33

        accuracy                           0.85        61
       macro avg       0.86      0.84      0.85        61
    weighted avg       0.86      0.85      0.85        61
```

```
Rows with iqr_flag == 1: 19
Rows with iqr_flag == 1 and >=3 votes: 10
Clipped columns (possible): ['age', 'trestbps', 'chol', 'thalach', 'oldpeak']
```






```
Train size: 210
Val size: 31
Test size: 61
Fitting 5 folds for each of 140 candidates, totalling 700 fits

Best CV accuracy: 0.8476190476190476
Best parameters: {'svm__C': 20, 'svm__class_weight': 'balanced', 'svm__degree': 2, 'svm__gamma': 0.001, 'svm__kernel': 'rbf'}

Best threshold: 0.471785060072921

TRAIN Metrics:
Accuracy : 0.8714285714285714
Precision: 0.8320610687022901
Recall   : 0.956140350877193
F1 Score : 0.889795918367347
AUC      : 0.9257127192982457

VALIDATION Metrics:
Accuracy : 0.8387096774193549
Precision: 0.8333333333333334
Recall   : 0.8823529411764706
F1 Score : 0.8571428571428571
AUC      : 0.9159663865546218

TEST Metrics:
Accuracy : 0.8524590163934426
Precision: 0.8157894736842105
Recall   : 0.9393939393939394
F1 Score : 0.8732394366197183
AUC      : 0.9134199134199135
```

10. Random Forest Classifier

10.1 Description of the Model

Random Forest is an ensemble learning algorithm that builds multiple decision trees and combines their predictions to produce a more accurate and stable result.
Each tree is trained on a random subset of the data and features, which reduces overfitting and improves generalization compared to a single decision tree.

In this experiment, the model was trained using 200 decision trees with a maximum depth of 10, balancing model complexity and performance.
Random Forest also provides feature importance scores, offering insight into which variables contribute most to the classification task.

10.2 Applied Tools / Techniques

The Random Forest experiment used the following machine learning components and preprocessing steps:

Outlier Flag Processing

- Conversion of outlier indicator columns (iqr, cat_outlier, iforest, lof) into boolean formats.
- Removal of helper columns not used as model inputs.

Feature & Target Preparation

- Features extracted by excluding:
  - target
  - Categorical outlier explanation
  - Outlier indicator columns
- Target defined as y = df_outliers["target"]

Train/Test Split

- 80/20 split with stratified sampling
- random_state = 42 for reproducibility

Cross-Validation (Before Training)

- 5-fold cross-validation performed on a baseline RandomForest model:
  - n_estimators=500
  - max_depth=10
  - min_samples_split=10
    - min_samples_leaf=4
    - max_features=0.6
- Metrics computed:
  - Individual CV accuracy scores
  - Mean accuracy

- Standard deviation

GridSearchCV (Hyperparameter Optimization)

Performed to find the optimal anti-overfitting parameters.
Grid search included:

- n_estimators = [300, 500]
- max_depth = [8, 10, 12]
- min_samples_split = [8, 10, 15]
- min_samples_leaf = [3, 4, 5]
- max_features = [0.5, 0.6, 0.7]
- bootstrap = [True]

5-fold Stratified CV used inside GridSearchCV.

Outputs:

- Best hyperparameters
- Best cross-validation accuracy score

Final Model Training

- Model re-trained on training data using the best parameters from grid search
- Predictions generated for both training and testing sets
- Evaluation metrics:
  - Training accuracy
  - Testing accuracy

Model Evaluation

Your code calculates:

- Training accuracy
- Testing accuracy

(Confusion Matrix and Classification Report were *not* generated in this code block.)

10.3 Used Libraries

Data Handling

- pandas – data manipulation and preprocessing

Visualization

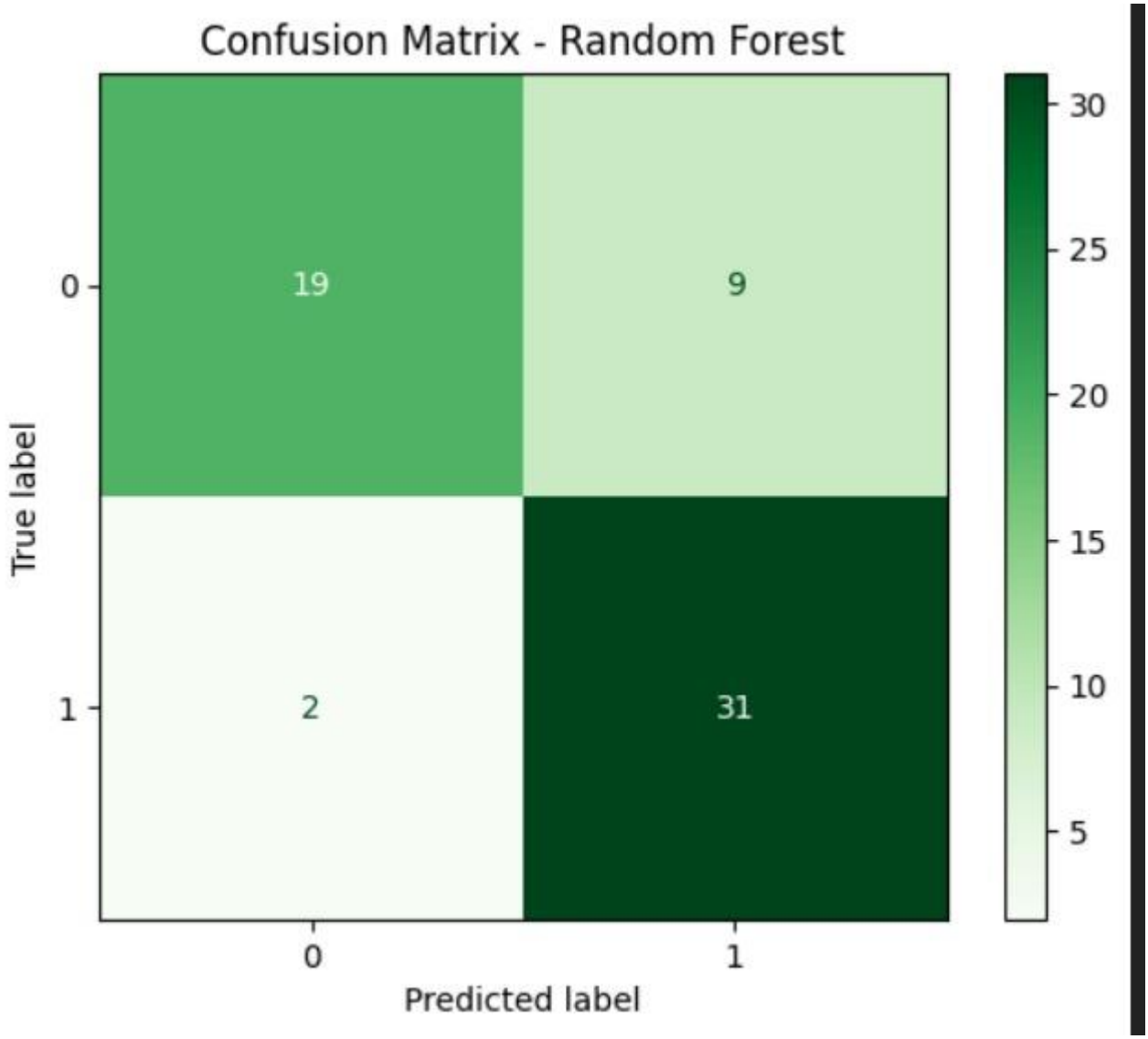
- matplotlib / seaborn
  (Used earlier in your notebook but not in the Random Forest block itself)

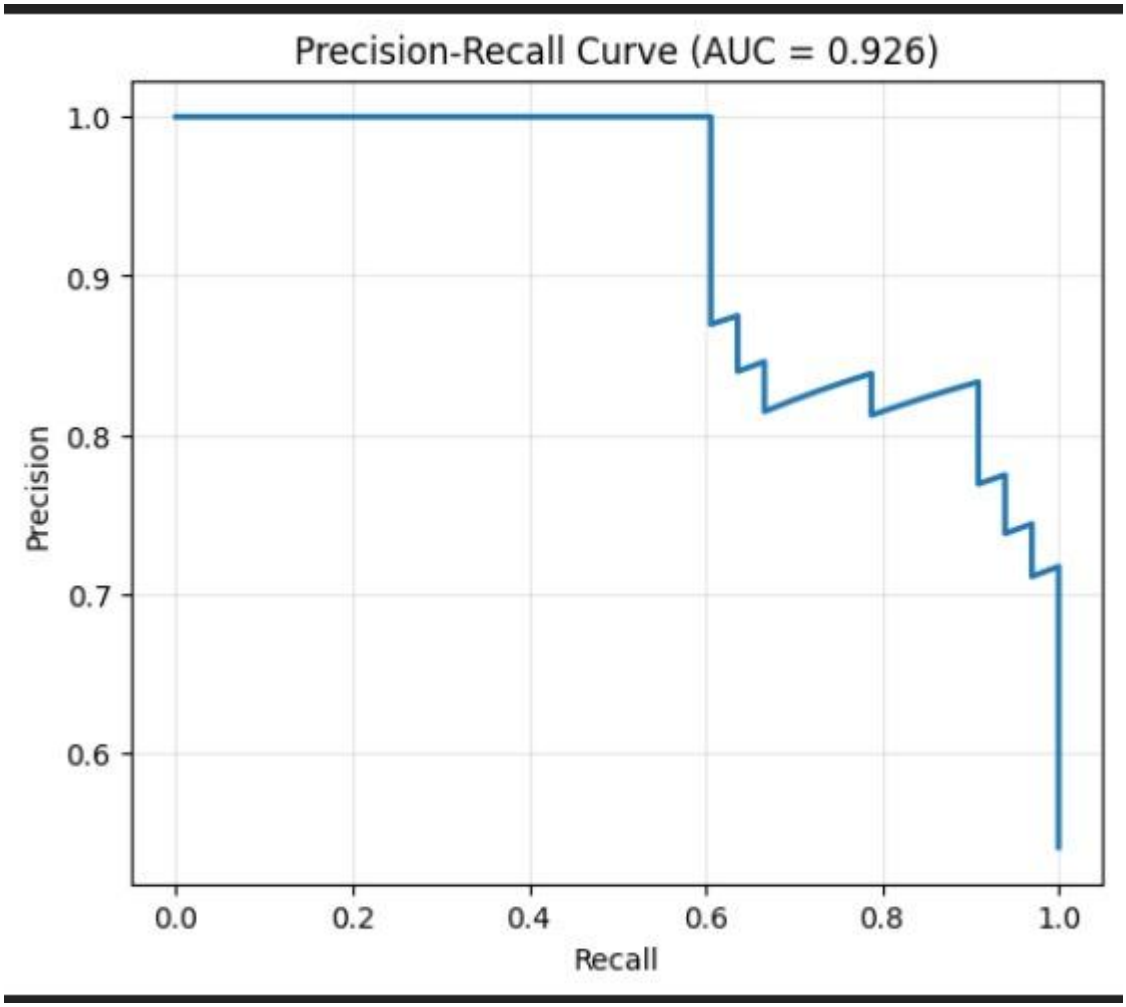
Machine Learning (scikit-learn)

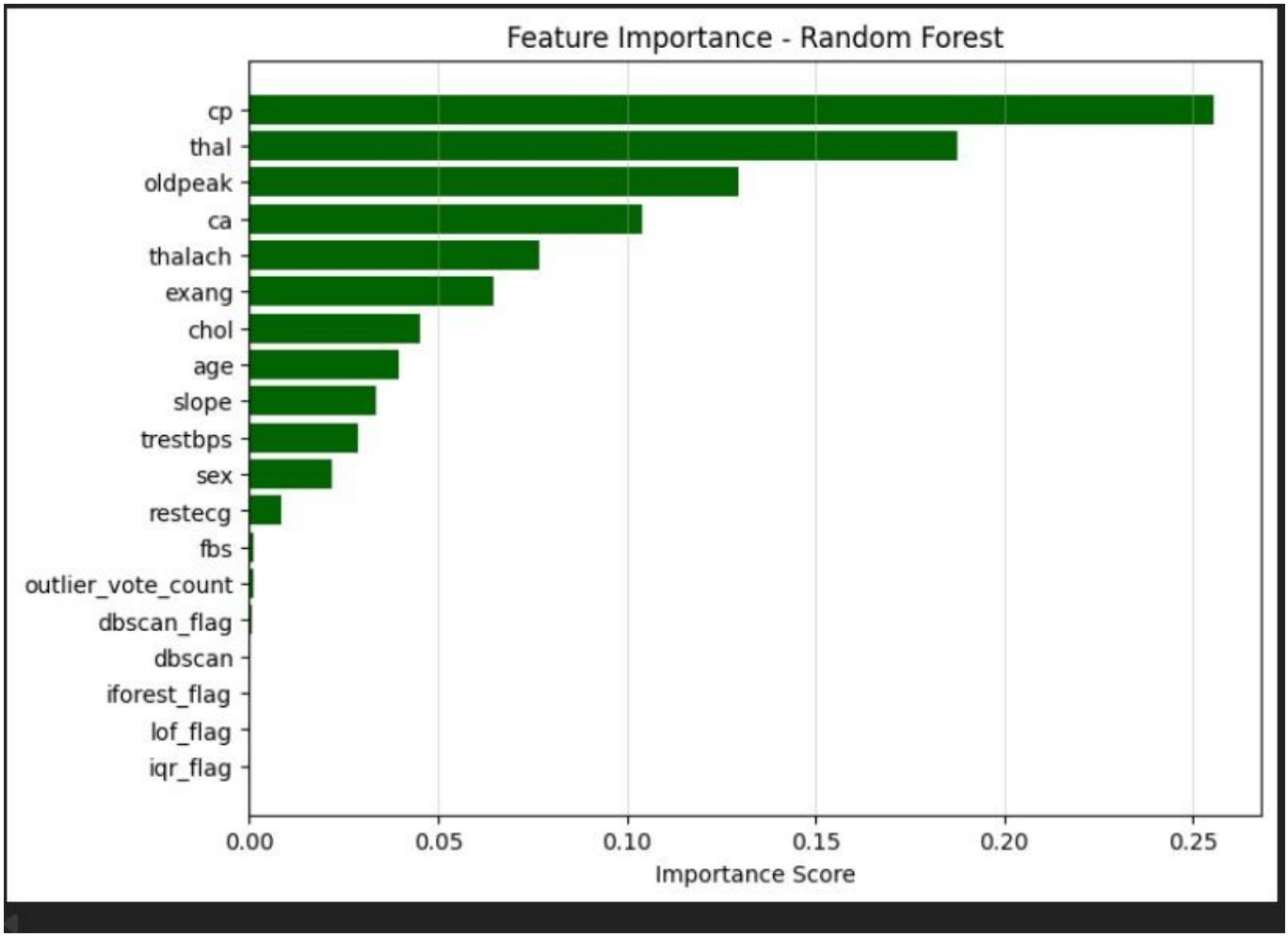
- train_test_split – stratified dataset splitting

- RandomForestClassifier – main ensemble model
- cross_val_score – baseline cross-validation
- GridSearchCV – hyperparameter tuning
- accuracy_score – evaluation metric

10.4 Results




Precision-Recall Curve (AUC = 0.926)
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.0
0.2
0.4
0.6
0.8
1.0


Feature Importance - Random Forest
cp
thal
oldpeak
ca
thalach
exang
chol
age
slope
trestbps
sex
restecg
fbs
outlier_vote_count
dbscan_flag
dbscan
iforest_flag
lof_flag
iqr_flag
0.00
0.05
0.10
0.15
0.20
0.25
Importance Score

📌 Classification Report:

	precision	recall	f1-score	support
0	0.90	0.68	0.78	28
1	0.78	0.94	0.85	33
accuracy			0.82	61
macro avg	0.84	0.81	0.81	61
weighted avg	0.83	0.82	0.82	61