MAHATMA EDUCATION SOCIETY'S

PILLAI COLLEGE OF ARTS, COMMERCE & SCIENCE
(Autonomous)

NEW PANVEL

PROJECT REPORT ON

"Heart Disease Prediction Using Machine Learning Techniques"

IN PARTIAL FULFILLMENT OF
BACHELOR OF COMPUTER SCIENCE

SEMESTER VI 2025-26

PROJECT GUIDE

Prof. Sanjana Bhangale

SUBMITTED BY: **VEDASHREE PRADIP NIMJE**

ROLL NO: **9072**

Heart Disease Prediction Using Machine Learning Techniques

INTRODUCTION

Cardiovascular diseases continue to be a major global health concern. The diagnosis of heart disease depends on multiple interrelated clinical factors such as blood pressure, cholesterol levels, heart rate, and chest pain characteristics. Analyzing these factors manually can be complex and time-consuming.

With the advancement of data science, machine learning techniques provide efficient tools to identify patterns within medical datasets and assist in predictive analysis. By training models on historical clinical data, it is possible to classify whether a patient is likely to have heart disease.

This project focuses on building a classification model that predicts the presence of heart disease using clinical and demographic features. The study aims to combine medical understanding with machine learning techniques to improve early diagnosis and risk assessment.

ABSTRACT

Heart disease is one of the leading causes of death globally, making early detection and risk assessment extremely important. This project applies machine learning techniques to analyze clinical and demographic attributes in order to predict the presence of heart disease.

The dataset contains medical parameters such as age, chest pain type, cholesterol level, resting blood pressure, maximum heart rate, and other diagnostic indicators. A structured machine learning pipeline was followed, including data preprocessing, feature engineering, exploratory data analysis, model building, and evaluation. Logistic Regression was used as the classification model, and performance was evaluated using accuracy, confusion matrix, precision, recall, and F1-score.

The results demonstrate that machine learning can effectively support medical professionals in identifying high-risk patients at an early stage.

OBJECTIVES

The primary objectives of this project are:

1. To identify and classify data types of all attributes in the dataset.
2. To understand the medical significance of important clinical features.
3. To preprocess the dataset by handling missing values, duplicates, and outliers.

4. To perform feature engineering by creating new meaningful attributes such as risk_score and age_group.
5. To conduct exploratory data analysis (EDA) to identify trends and relationships between variables.
6. To encode categorical and binary variables appropriately.
7. To split the dataset into training and testing sets.
8. To normalize or standardize continuous variables where necessary.
9. To build a Logistic Regression classification model.
10. To evaluate the model using appropriate classification metrics.
11. To derive insights that can assist cardiologists in early diagnosis.

METHODOLOGY

The project was carried out using a structured and systematic approach:

1. Data Understanding

The dataset was examined to determine attribute types (numerical, categorical, binary). The medical meaning of key attributes such as chest pain type, maximum heart rate, oldpeak, number of major vessels, and thalassemia was studied.

2. Data Cleaning

The dataset was checked for missing values, inconsistent entries, and duplicate records. Outliers in cholesterol, resting blood pressure, and maximum heart rate were identified using visualization techniques.

3. Feature Engineering

A new feature called risk_score was created by combining medically relevant attributes. Patients were also categorized into age groups (Young, Middle-aged, Senior) to enhance interpretability.

4. Exploratory Data Analysis (EDA)

Statistical summaries and visualizations were used to analyze distributions and relationships between attributes and heart disease occurrence. A correlation heatmap was generated to identify highly correlated features and influential predictors.

5. Data Preparation

Categorical variables (cp, restecg, slope, thal) were converted into numerical form using encoding techniques. Binary variables were appropriately represented. The dataset was split into

training (80%) and testing (20%) sets. Continuous features were standardized using StandardScaler to improve model performance.

6. Model Building and Evaluation

A Logistic Regression model was trained using the training dataset. Predictions were made on the testing dataset. The model was evaluated using accuracy score, confusion matrix, precision, recall, and F1-score to assess classification performance.

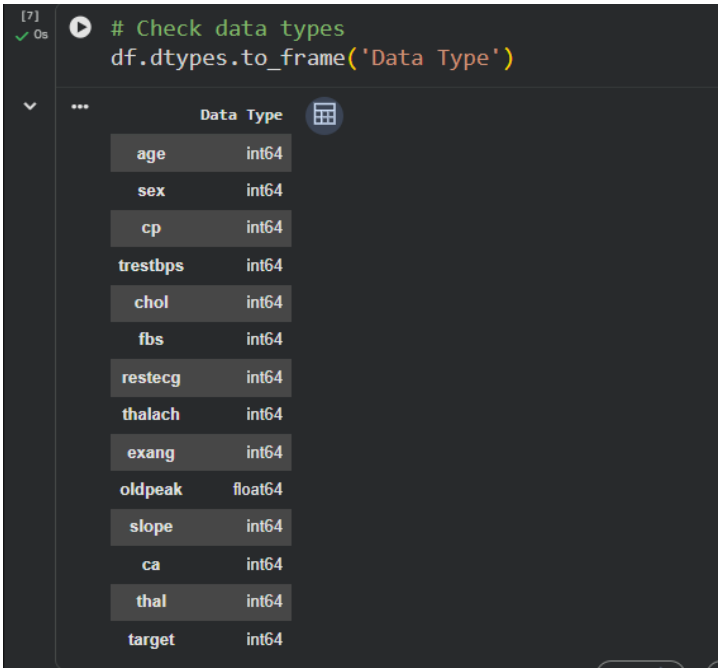
Load Dataset

```
import pandas as pd
import numpy as np

# Load dataset
df = pd.read_csv("heart.csv")

# View basic info
df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

Analysis:

1. Attribute Data Types



Analysis:

2. Medical Meaning of Specific Attributes

- **cp (Chest Pain Type):**

This attribute categorizes the type of chest pain experienced by the patient. In this dataset, the values typically represent:

  - **0:** Typical Angina (chest pain related to decreased blood flow to the heart).
  - **1:** Atypical Angina (chest pain not clearly related to the heart).
  - **2:** Non-anginal Pain (pain not caused by heart disease).
  - **3:** Asymptomatic (no pain).

- **thalach (Maximum Heart Rate Achieved):**

The highest heart rate (beats per minute) recorded during a stress test. Lower maximum heart rates relative to a person's age can sometimes indicate cardiovascular issues.

- **oldpeak (ST Depression):**

This refers to the "ST depression induced by exercise relative to rest." In an ECG (Electrocardiogram), the ST segment represents the interval between ventricular depolarization and repolarization. Significant depression in this segment during exercise is a common clinical indicator of myocardial ischemia (lack of blood flow to the heart muscle).

- **ca (Number of Major Vessels):**

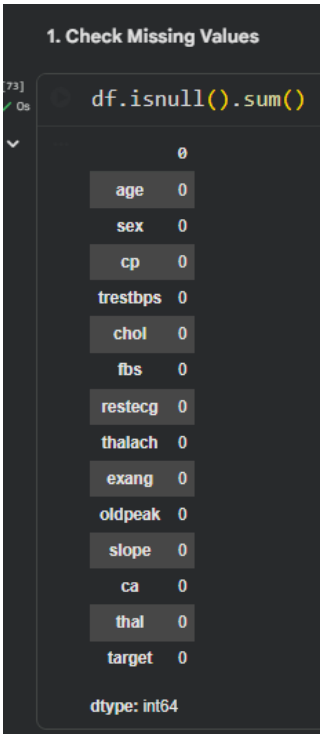
The number of major coronary vessels (0–3) that are visible or "colored" during a fluoroscopy procedure. Fluoroscopy uses a special dye to see how blood flows through the heart. A higher number of visible vessels generally indicates better heart health, whereas zero or few visible vessels might suggest blockages.

- **thal (Thalassemia / Thallium Stress Test):**

This is the result of a thallium stress test, which shows how well blood flows to the heart muscle.

  - **Normal:** Blood flow is healthy.
  - **Fixed Defect:** There is a part of the heart with no blood flow (often indicating permanent damage).
  - **Reversible Defect:** Blood flow is reduced during exercise but returns to normal at rest (indicating potential blockage/ischemia).

3. Check for missing or inconsistent values in the dataset



2. Check for Zero Values in Important Medical Columns

+ Code + Text

```
(df[['chol', 'trestbps', 'thalach']] == 0).sum()
```

```
            0
chol        0
trestbps    0
thalach     0
dtype: int64
```

```
print("cp unique values:", df['cp'].unique())
print("restecg unique values:", df['restecg'].unique())
print("slope unique values:", df['slope'].unique())
print("thal unique values:", df['thal'].unique())
print("ca unique values:", df['ca'].unique())
```

```
cp unique values: [0 1 2 3]
restecg unique values: [1 0 2]
slope unique values: [2 0 1]
thal unique values: [3 2 1 0]
ca unique values: [2 0 1 3 4]
```

Analysis:

4. Detect duplicate records and explain how they should be handled.

```
print("Original Shape:", df.shape)

df = df.drop_duplicates()

print("New Shape After Removing Duplicates:", df.shape)
```

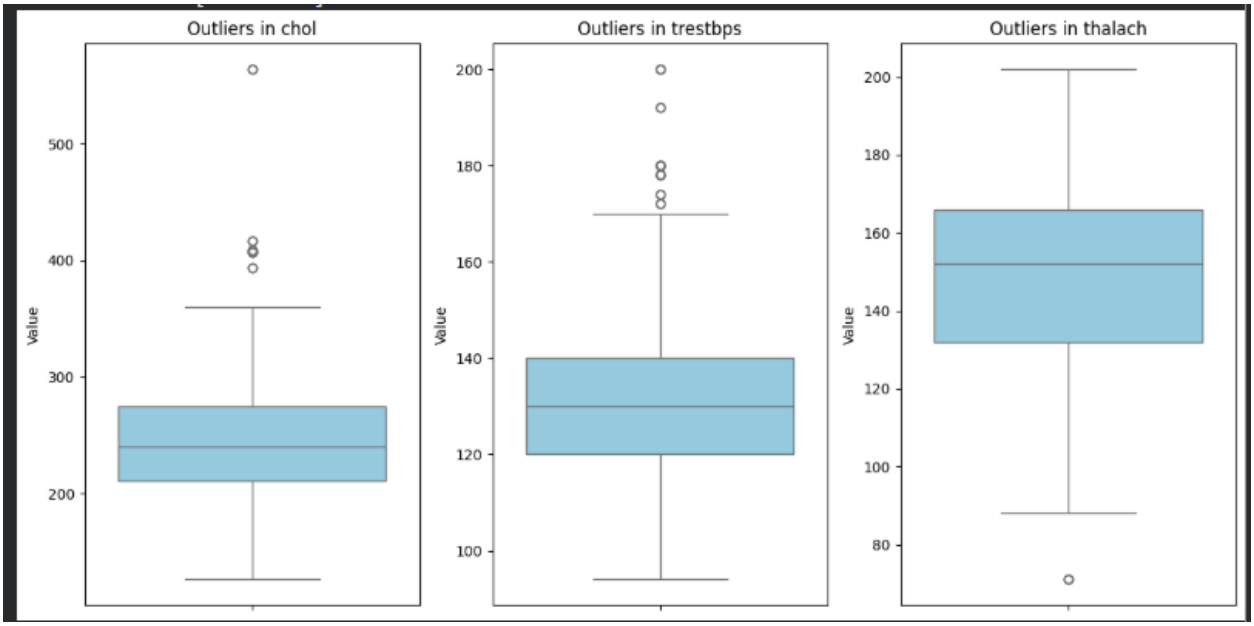
```
Original Shape: (1025, 20)
New Shape After Removing Duplicates: (302, 20)
```

Analysis:

5. Verify whether chol, trestbps, and thalach contain outliers.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns


# Define the columns to verify
columns_to_check = ['chol', 'trestbps', 'thalach']

print("--- Outlier Verification Results (IQR Method) ---")

for col in columns_to_check:
    # Calculate Q1 (25th percentile) and Q3 (75th percentile)
    Q1 = df[col].quantile(0.25)
    Q3 = df[col].quantile(0.75)

    # Calculate the Interquartile Range (IQR)
    IQR = Q3 - Q1

    # Define bounds for outliers
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = df[(df[col] < lower_bound) | (df[col] > upper_bound)]

    # Print results for the report
    print(f"\nAttribute: {col}")
    print(f"  Q1: {Q1}, Q3: {Q3}, IQR: {IQR}")
    print(f"  Lower Bound: {lower_bound}")
    print(f"  Upper Bound: {upper_bound}")
    print(f"  Number of Outliers: {len(outliers)}")
    if len(outliers) > 0:
        print(f"  Outlier Values: {outliers[col].values}")

# 3. Visual Verification using Boxplots
plt.figure(figsize=(12, 6))

# Creating a subplot for each column
for i, col in enumerate(columns_to_check):
    plt.subplot(1, 3, i+1)
    sns.boxplot(y=df[col], color='skyblue')
    plt.title(f'Outliers in {col}')
    plt.ylabel('Value')

plt.tight_layout()
plt.show()
```

```
--- Outlier Verification Results (IQR Method) ---

Attribute: chol
  Q1: 211.0, Q3: 275.0, IQR: 64.0
  Lower Bound: 115.0
  Upper Bound: 371.0
  Number of Outliers: 16
  Outlier Values: [417 564 409 564 394 407 564 407 394 394 409 417 407 407 417 409]

Attribute: trestbps
  Q1: 120.0, Q3: 140.0, IQR: 20.0
  Lower Bound: 90.0
  Upper Bound: 170.0
  Number of Outliers: 30
  Outlier Values: [180 178 174 180 192 200 178 192 180 200 192 172 180 174 178 180 200 180
 178 180 178 174 180 200 172 180 178 178 172 180]

Attribute: thalach
  Q1: 132.0, Q3: 166.0, IQR: 34.0
  Lower Bound: 81.0
  Upper Bound: 217.0
  Number of Outliers: 4
  Outlier Values: [71 71 71 71]
```



Analysis:

6. Create a new feature called risk_score using relevant attributes.

```
df['risk_score'] = (
    df['chol'] +
    df['trestbps'] +
    df['oldpeak'] +
    df['ca']
)

# Display the first few rows to verify the new column
print("--- Data with New Feature 'risk_score' ---")
df[['chol', 'trestbps', 'oldpeak', 'ca', 'risk_score', 'target']].head()
```

```
--- Data with New Feature 'risk_score' ---
```

	chol	trestbps	oldpeak	ca	risk_score	target
0	-0.659332	-0.377636	-0.060888	1.209221	0.111365	0
1	-0.833861	0.479107	1.727137	-0.731971	0.640412	0
2	-1.396233	0.764688	1.301417	-0.731971	-0.062099	0
3	-0.833861	0.936037	-0.912329	0.238625	-0.571529	0
4	0.930822	0.364875	0.705408	2.179817	4.180922	0

Analysis:

7. Categorize patients into age groups (Young, Middle-aged, Senior).

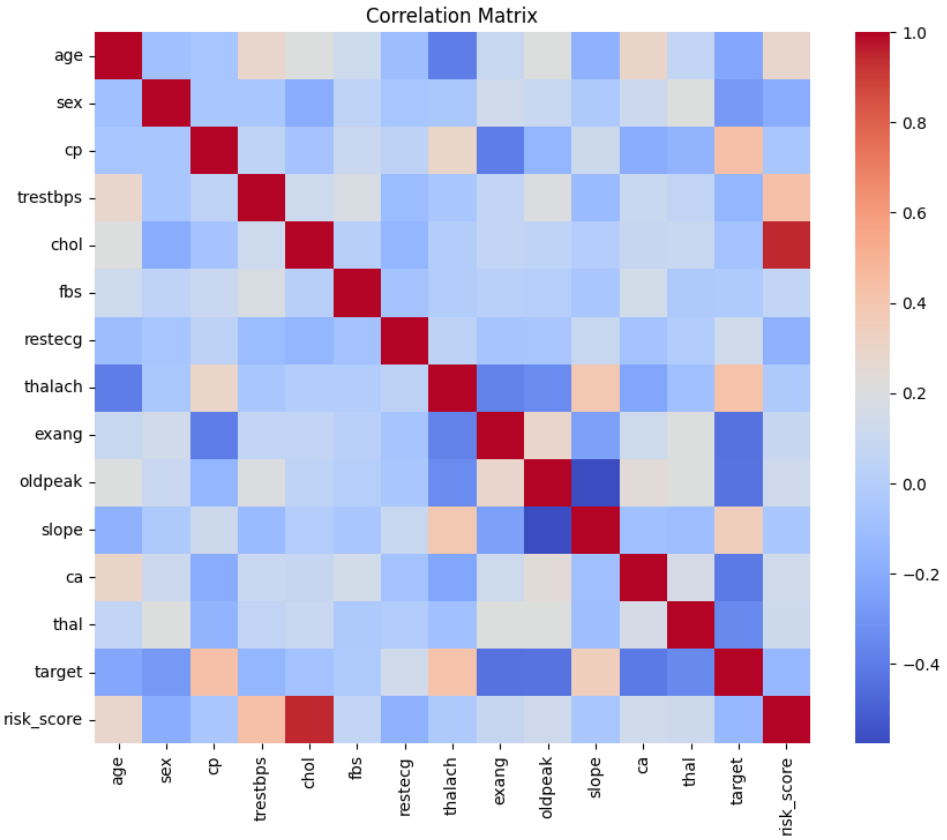
```
# Create age groups
def categorize_age(age):
    if age < 40:
        return "Young"
    elif 40 <= age <= 60:
        return "Middle-aged"
    else:
        return "Senior"

df['age_group'] = df['age'].apply(categorize_age)

# Check result
df[['age', 'age_group']].head()
```

	age	age_group
0	-0.268437	Young
1	-0.158157	Young
2	1.716595	Young
3	0.724079	Young
4	0.834359	Young

Analysis:

8. Identify highly correlated features using a correlation matrix.

```python
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np

# 1. Select only numerical columns
corr_matrix = df.corr(numeric_only=True)

# 2. Plot correlation heatmap
plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix, cmap='coolwarm')
plt.title("Correlation Matrix")
plt.show()

# 3. Identify highly correlated feature pairs (threshold = 0.7)
print("Highly Correlated Feature Pairs (|correlation| > 0.7):\n")

found = False

for i in range(len(corr_matrix.columns)):
    for j in range(i):
        corr_value = corr_matrix.iloc[i, j]
        if abs(corr_value) > 0.7:
            print(f"{corr_matrix.columns[i]} and {corr_matrix.columns[j]} : {corr_value:.2f}")
            found = True

if not found:
    print("No highly correlated feature pairs found.")
```



Analysis:

9. Convert categorical variables (cp, restecg, slope, thal) into suitable numerical representations.

```
import pandas as pd

categorical_cols = ['cp', 'restecg', 'slope', 'thal']

df = pd.get_dummies(df, columns=categorical_cols, drop_first=True)

df.head()
```

	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	ca	target	cp_1	cp_2	cp_3	restecg_1	restecg_2	slope_1	slo
0	52	1	125	212	0	168	0	1.0	2	0	False	False	False	True	False	False	
1	53	1	140	203	1	155	1	3.1	0	0	False	False	False	False	False	False	
2	70	1	145	174	0	125	1	2.6	0	0	False	False	False	True	False	False	
3	61	1	148	203	0	161	0	0.0	1	0	False	False	False	True	False	False	
4	62	0	138	294	1	106	0	1.9	3	0	False	False	False	True	False	True	

Analysis:

10. Encode the binary variables (sex, fbs, exang, target) appropriately.

```
binary_cols = ['sex', 'fbs', 'exang', 'target']

for col in binary_cols:
    print(f"{col} unique values:", df[col].unique())
```

```
sex unique values: [1 0]
fbs unique values: [0 1]
exang unique values: [0 1]
target unique values: [0 1]
```

Analysis:

11. Split the dataset into training and testing sets.

```
from sklearn.model_selection import train_test_split

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

# Split the data
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

# Print shapes
print("Training set shape:", X_train.shape)
print("Testing set shape:", X_test.shape)
```

```
Training set shape: (241, 21)
Testing set shape: (61, 21)
```

Analysis:

12. Normalize or standardize continuous features and justify the method used.

```
from sklearn.preprocessing import StandardScaler

continuous_cols = ['age', 'trestbps', 'chol', 'thalach', 'oldpeak', 'ca']

scaler = StandardScaler()

df[continuous_cols] = scaler.fit_transform(df[continuous_cols])

df.head()
```

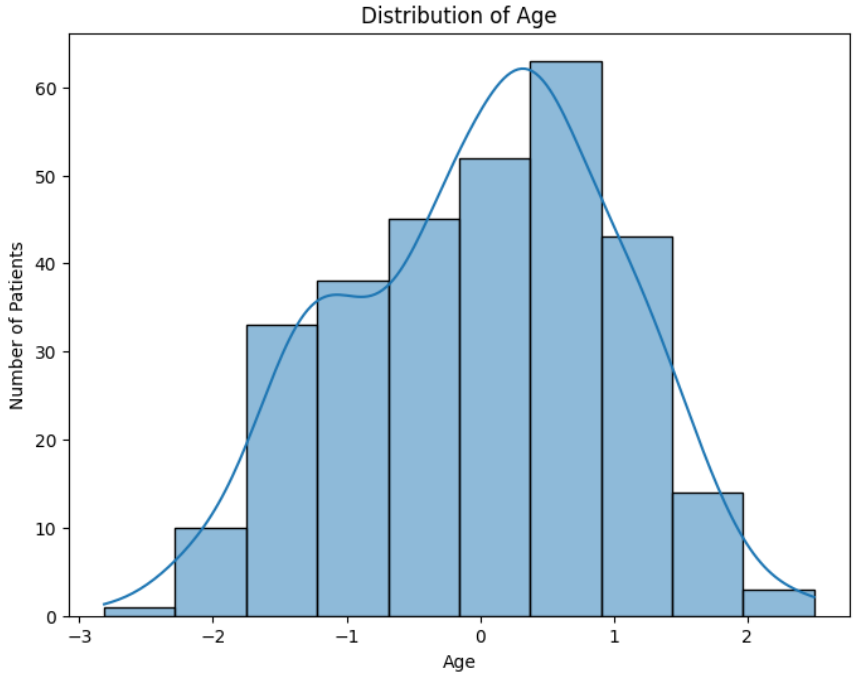
	age	sex	trestbps	chol	fbs	thalach	exang	oldpeak	ca	target	cp_1	cp_2	cp_3	restecg_1	restecg
0	-0.268437	1	-0.377636	-0.659332	0	0.821321	0	-0.060888	1.209221	0	False	False	False	True	Fa
1	-0.158157	1	0.479107	-0.833861	1	0.255968	1	1.727137	-0.731971	0	False	False	False	False	Fa
2	1.716595	1	0.764688	-1.396233	0	-1.048692	1	1.301417	-0.731971	0	False	False	False	True	Fa
3	0.724079	1	0.936037	-0.833861	0	0.516900	0	-0.912329	0.238625	0	False	False	False	True	Fa
4	0.834359	0	0.364875	0.930822	1	-1.874977	0	0.705408	2.179817	0	False	False	False	True	Fa

Analysis:

📊 B. Exploratory Data Analysis (EDA)

13. Analyze the distribution of age among patients.

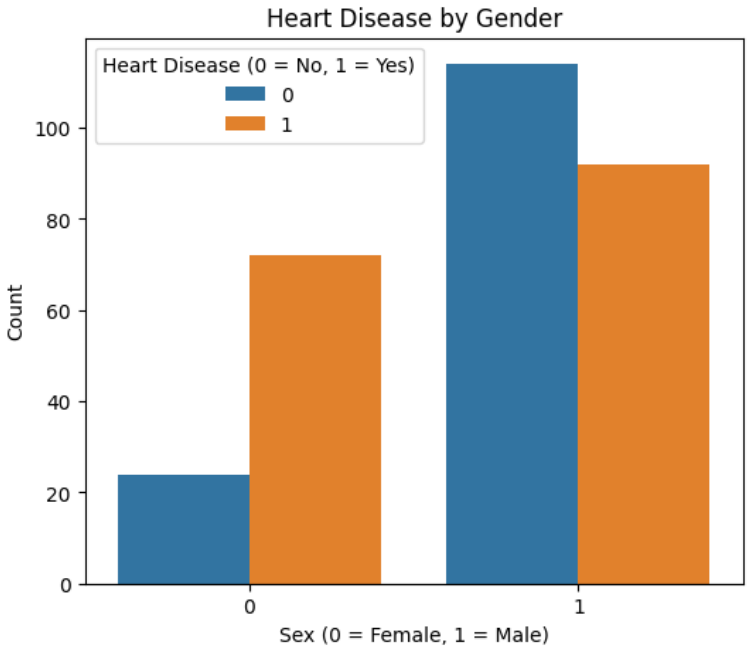
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(8,6))
sns.histplot(df['age'], bins=10, kde=True)

plt.title("Distribution of Age")
plt.xlabel("Age")
plt.ylabel("Number of Patients")
plt.show()
```



Analysis:

14. Compare heart disease occurrence across gender.

```
plt.figure(figsize=(6,5))
sns.countplot(x='sex', hue='target', data=df)

plt.title("Heart Disease by Gender")
plt.xlabel("Sex (0 = Female, 1 = Male)")
plt.ylabel("Count")
plt.legend(title="Heart Disease (0 = No, 1 = Yes)")
plt.show()
```



Analysis:

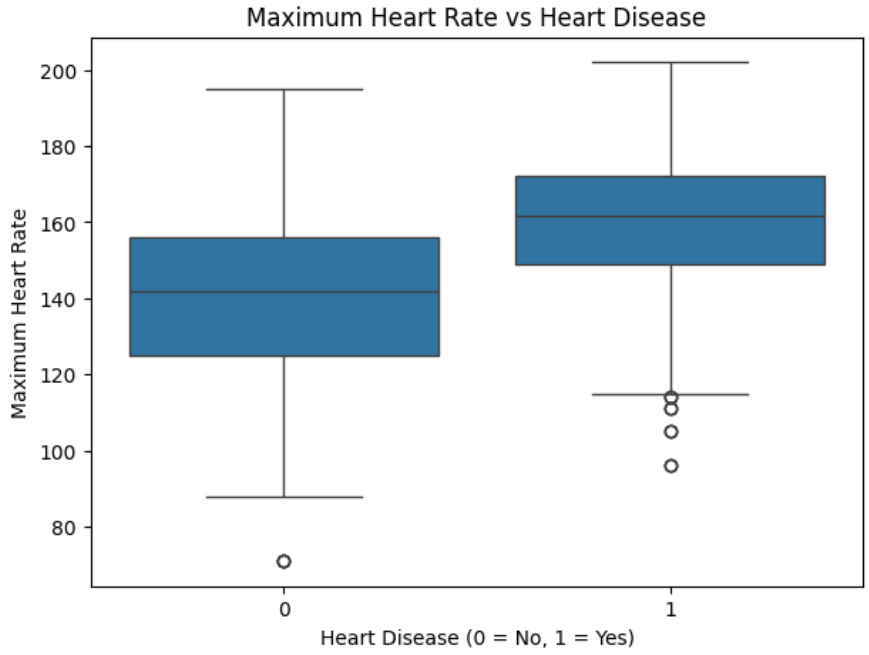
15. Study the relationship between chest pain type (cp) and target.

```
df = pd.read_csv("heart.csv")
plt.figure(figsize=(7,5))
sns.countplot(x='cp', hue='target', data=df)

plt.title("Chest Pain Type vs Heart Disease")
plt.xlabel("Chest Pain Type")
plt.ylabel("Count")
plt.legend(title="Heart Disease (0 = No, 1 = Yes)")
plt.show()
```

Analysis:

16. Does maximum heart rate (thalach) differ for patients with and without heart disease?

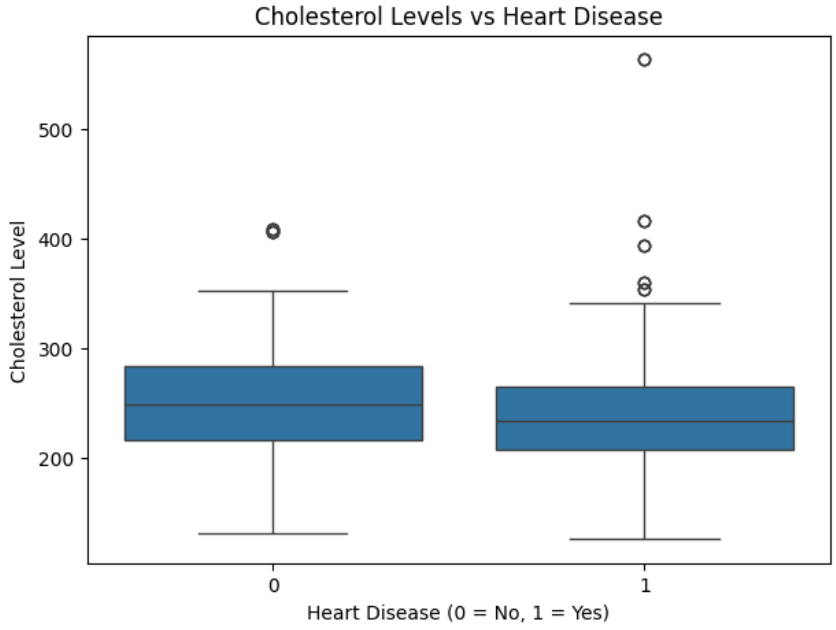
```
#Since thalach is continuous, we use a boxplot.
plt.figure(figsize=(7,5))
sns.boxplot(x='target', y='thalach', data=df)

plt.title("Maximum Heart Rate vs Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Maximum Heart Rate")
plt.show()
```



Analysis:

17. Analyze how cholesterol levels impact heart disease.

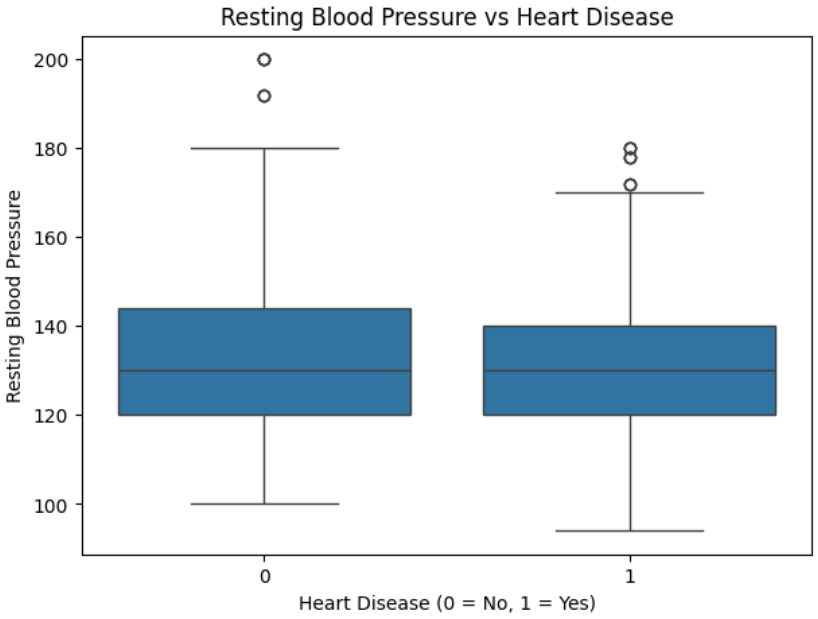
```
plt.figure(figsize=(7,5))
sns.boxplot(x='target', y='chol', data=df)

plt.title("Cholesterol Levels vs Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Cholesterol Level")
plt.show()
```



Analysis:

18. Compare resting blood pressure (trestbps) across target classes.

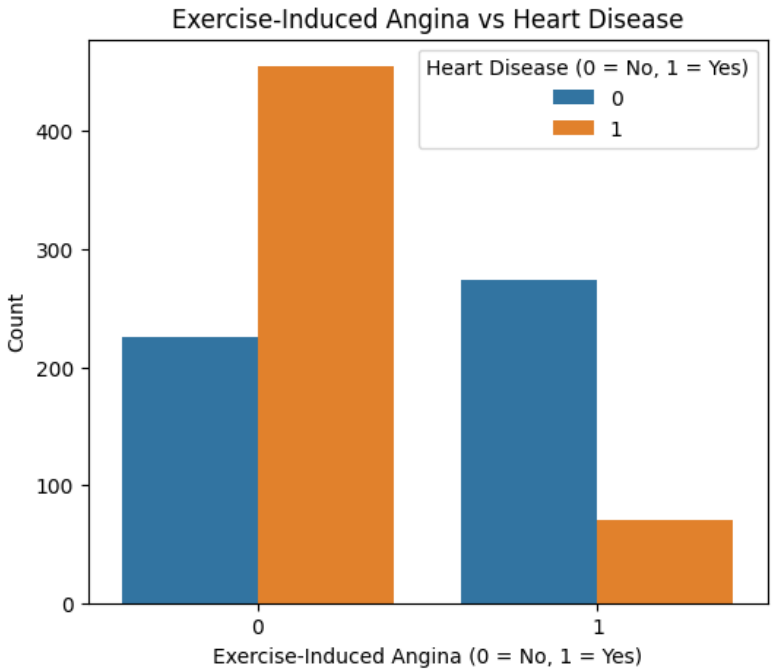
```
plt.figure(figsize=(7,5))
sns.boxplot(x='target', y='trestbps', data=df)

plt.title("Resting Blood Pressure vs Heart Disease")
plt.xlabel("Heart Disease (0 = No, 1 = Yes)")
plt.ylabel("Resting Blood Pressure")
plt.show()
```



Analysis:

19. Study the effect of exercise-induced angina (exang) on heart disease.

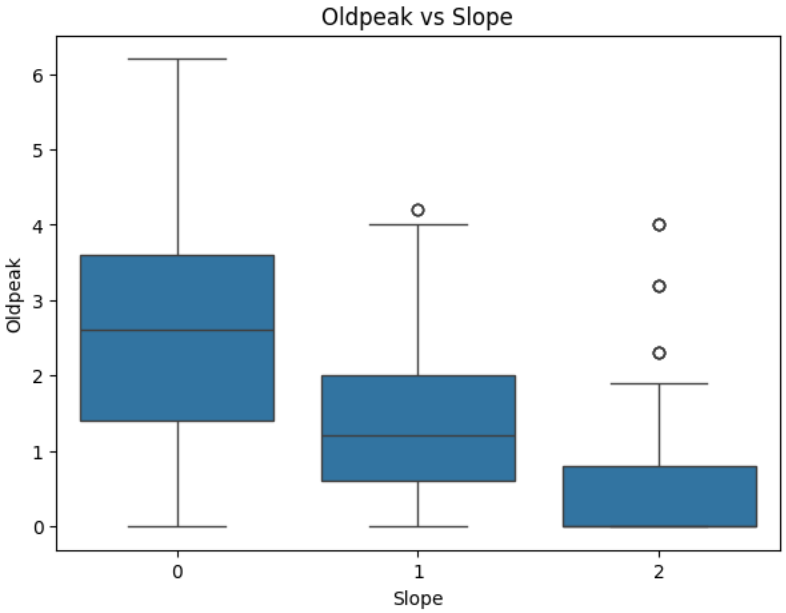
```
plt.figure(figsize=(6,5))
sns.countplot(x='exang', hue='target', data=df)

plt.title("Exercise-Induced Angina vs Heart Disease")
plt.xlabel("Exercise-Induced Angina (0 = No, 1 = Yes)")
plt.ylabel("Count")
plt.legend(title="Heart Disease (0 = No, 1 = Yes)")
plt.show()
```



Analysis:

20. Analyze the relationship between oldpeak and slope.

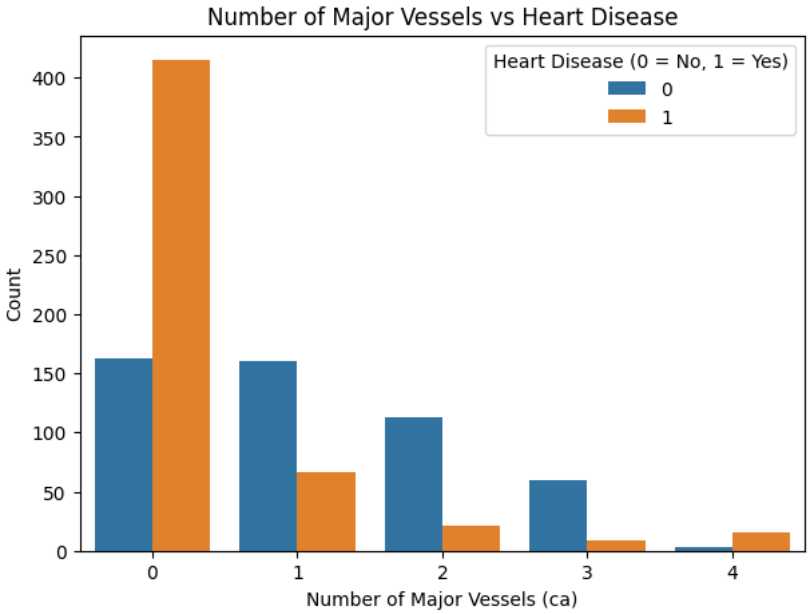
```
plt.figure(figsize=(7,5))
sns.boxplot(x='slope', y='oldpeak', data=df)

plt.title("Oldpeak vs Slope")
plt.xlabel("Slope")
plt.ylabel("Oldpeak")
plt.show()
```



Analysis:

21. Compare the number of major vessels (ca) with disease presence.

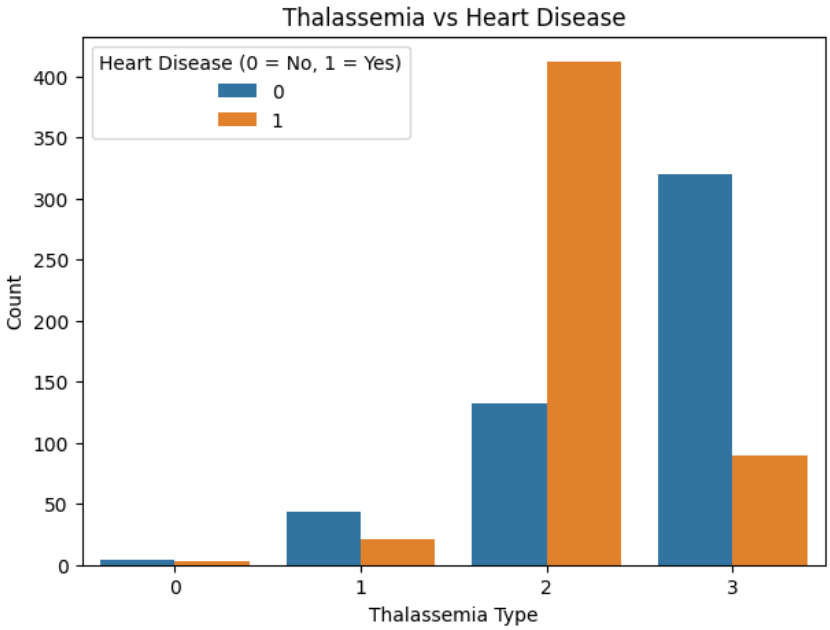
```
plt.figure(figsize=(7,5))
sns.countplot(x='ca', hue='target', data=df)

plt.title("Number of Major Vessels vs Heart Disease")
plt.xlabel("Number of Major Vessels (ca)")
plt.ylabel("Count")
plt.legend(title="Heart Disease (0 = No, 1 = Yes)")
plt.show()
```



Analysis:

22. Evaluate how thalassemia (thal) affects heart disease risk.

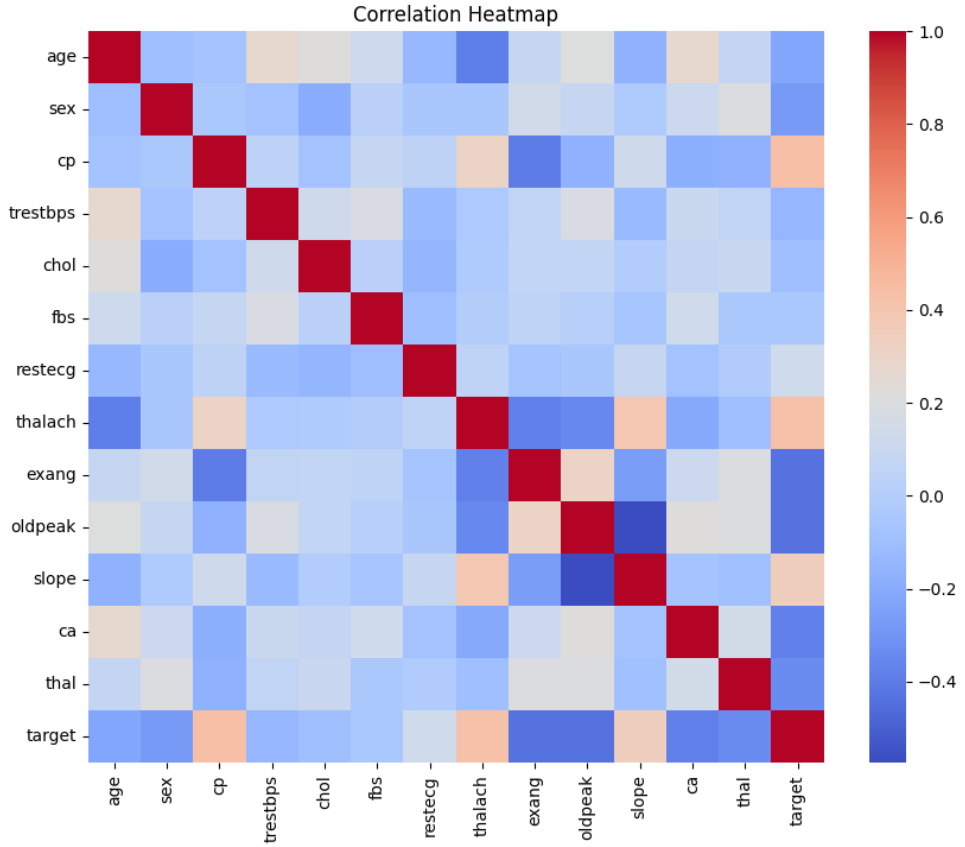
```
plt.figure(figsize=(7,5))
sns.countplot(x='thal', hue='target', data=df)

plt.title("Thalassemia vs Heart Disease")
plt.xlabel("Thalassemia Type")
plt.ylabel("Count")
plt.legend(title="Heart Disease (0 = No, 1 = Yes)")
plt.show()
```



Analysis:

23. Plot and interpret the correlation heatmap.

```
import seaborn as sns
import matplotlib.pyplot as plt

corr_matrix = df.corr(numeric_only=True)

plt.figure(figsize=(10,8))
sns.heatmap(corr_matrix, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```


Correlation Heatmap
age
sex
cp
trestbps
chol
fbs
restecg
thalach
exang
oldpeak
slope
ca
thal
target
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4

Analysis:

24. Identify the top 3 most influential features affecting the target.

```
# Get correlation with target
target_corr = corr_matrix['target'].drop('target')

# Sort by absolute correlation values
top_features = target_corr.abs().sort_values(ascending=False).head(3)

print("Top 3 Most Influential Features Affecting Target:\n")

for feature in top_features.index:
    print(f"{feature} : {target_corr[feature]:.2f}")
```

```
Top 3 Most Influential Features Affecting Target:

oldpeak : -0.44
exang : -0.44
cp : 0.43
```

Analysis:

25. Summarize key insights that could help a cardiologist in early diagnosis.

```
print("\nKey Insights for Early Diagnosis:\n")

for feature in top_features.index:
    if target_corr[feature] > 0:
        print(f"- Higher {feature} is associated with increased heart disease risk.")
    else:
        print(f"- Higher {feature} is associated with lower heart disease occurrence (negative correlation).")

print("\nThese features can assist cardiologists in early identification of high-risk patients.")
```

```
...
    Key Insights for Early Diagnosis:

    - Higher oldpeak is associated with lower heart disease occurrence (negative correlation).
    - Higher exang is associated with lower heart disease occurrence (negative correlation).
    - Higher cp is associated with increased heart disease risk.

    These features can assist cardiologists in early identification of high-risk patients.
```

Analysis:

Model Building

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Drop target only
X = df.drop('target', axis=1)
y = df['target']

# If age_group exists, remove it safely
if 'age_group' in X.columns:
    X = X.drop('age_group', axis=1)

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y,
    test_size=0.2,
    random_state=42
)

# Standardize features
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Train model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)

print("Model building completed successfully.")
```

```
Model building completed successfully.
```

Analysis:

Model Testing

```
# Make predictions on test data
y_pred = model.predict(X_test)

print("Model testing completed.")
```

```
Model testing completed.
```

Analysis:

Model Accuracy

```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Model Accuracy:", round(accuracy * 100, 2), "%")

# Confusion Matrix
print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# Classification Report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Model Accuracy: 79.51 %

Confusion Matrix:
[[73 29]
 [13 90]]

Classification Report:
              precision    recall  f1-score   support

           0       0.85      0.72      0.78       102
           1       0.76      0.87      0.81       103

    accuracy                           0.80       205
   macro avg       0.80      0.79      0.79       205
weighted avg       0.80      0.80      0.79       205
```

Analysis:

SWOT ANALYSIS

Strengths

- Structured and systematic machine learning workflow.
- Proper preprocessing and feature engineering.
- Use of interpretable classification model.
- Clear evaluation using appropriate metrics.

Weaknesses

- Limited dataset size may affect generalization.
- Logistic Regression assumes linear relationships between features and target.
- Model performance may vary across different populations.

Opportunities

- The model can be enhanced using advanced algorithms such as Random Forest or Gradient Boosting.
- Integration into clinical decision support systems.

- Application to larger and real-time healthcare datasets.

Threats

- Risk of overfitting if not properly validated.
- Variability in medical data across demographics.
- Ethical concerns regarding automated medical predictions.

CONCLUSION

This project successfully demonstrates the application of machine learning techniques in predicting heart disease using clinical data. Through systematic preprocessing, feature engineering, and exploratory analysis, important relationships between medical attributes and heart disease were identified.

The Logistic Regression model achieved satisfactory classification performance, indicating that machine learning can be an effective tool in supporting early diagnosis. Key features such as chest pain type, maximum heart rate, number of major vessels, and cholesterol levels were found to significantly influence disease prediction.

Overall, the project highlights the potential of data-driven approaches in improving healthcare decision-making and assisting cardiologists in early risk assessment.