Assessment Report

on

"Heart Disease Prediction"

submitted as partial fulfillment for the award of

BACHELOR OF TECHNOLOGY DEGREE

SESSION 2024-25

in

CSE(AI)

By

Group No. 10

Members Name:

Ayushi Singh(202401100300088)

Divya Pal(202401100300103)

Eisha(202401100300109)

Ganesh Gupta(202401100300110)

Under the supervision of

"Mr. Shivansh Prasad"

KIET Group of Institutions, Ghaziabad

May, 2025

---

1. Introduction

Heart disease is one of the leading causes of death worldwide. Early prediction of heart disease can save countless lives and reduce the cost of healthcare. This project explores the application of supervised machine learning models to predict the presence of heart disease based on various medical parameters such as age, cholesterol level, blood pressure, and more.

---

2. Problem Statement

To build a classification model that can accurately predict the presence of heart disease in a patient using medical diagnostic data. This tool can assist healthcare professionals in making informed decisions and providing early treatment.

---

3. Objectives

- Analyze and preprocess the heart disease dataset.
- Train a Logistic Regression model.
- Evaluate the model's performance using classification metrics.
- Visualize data correlations and model predictions.

---

4. Methodology

Data Collection:
The dataset used is the UCI Heart Disease dataset. It contains patient-level features such as age, sex, resting blood pressure, cholesterol, fasting blood sugar, and more.

Data Preprocessing:

- Renamed the target column num to target (if present).
- Encoded categorical (non-numeric) columns using LabelEncoder.
- Split the data into features (X) and target (y), followed by an 80-20 train-test split.
- Used StandardScaler to normalize features for better model convergence.

Model Building:

- Trained a LogisticRegression model from sklearn with max_iter=1000.

Evaluation:

- Used metrics like accuracy, precision, recall, and F1-score.
- Plotted a confusion matrix and correlation heatmap for interpretability.
- Created a countplot to visualize the class distribution of the target.

---

5. Data Preprocessing

The dataset is cleaned and prepared as follows:

- Categorical features were detected and encoded using LabelEncoder.
- Checked and renamed the target column (num) to target if required.
- Dataset split into train and test sets using train_test_split (80% train, 20% test).
- No missing values were detected in this specific version of the dataset.

---

6. Model Implementation

- A **Logistic Regression** model was trained using scikit-learn.

- The model was fit on the training data and predictions were made on the test set.
- max_iter=1000 was set to ensure convergence of the model.

---

7. Evaluation Metrics

- **Accuracy Score**: Proportion of correctly predicted instances.
- **Precision**: Ratio of true positives to total predicted positives.
- **Recall**: Ratio of true positives to actual positives.
- **F1-Score**: Harmonic mean of precision and recall.
- **Confusion Matrix**: Plotted using a heatmap to visualize performance across classes.

---

8. Code

 # Step 0: Import Libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear_model import LogisticRegression

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

 # Step 1: Load the dataset

```
df = pd.read_csv('heart_disease_uci.csv')
```

```
# Clean column names (remove any spaces)
df.columns = df.columns.str.strip()
```

 # Step 2: Display basic information

```
print("First 5 rows of the dataset:")
print(df.head())
print("\nDataset Info:")
print(df.info())
print("\nStatistical Summary:")
print(df.describe())
```

 # Step 3: Handle categorical data

```
for column in df.columns:
    if df[column].dtype == 'object':
        le = LabelEncoder()
        df[column] = le.fit_transform(df[column])
        print(f"Encoded '{column}'")
```

 # Step 4: Handle missing values

```
df.replace('?', np.nan, inplace=True)
df = df.apply(pd.to_numeric, errors='coerce')  # Convert all to numeric
df.fillna(df.median(), inplace=True)  # Fill NaNs with median
print("\nMissing values per column after cleaning:")
```

```python
print(df.isnull().sum())

# Step 4.1: Rename target column and convert to binary classification

df.rename(columns={'num': 'target'}, inplace=True)

df['target'] = (df['target'] > 0).astype(int)  # 0 = no heart disease, 1 = heart disease

# Step 5: Correlation Heatmap

plt.figure(figsize=(12, 10))

sns.heatmap(df.corr(), annot=True, cmap='coolwarm')

plt.title('Correlation Heatmap')

plt.show()

# Step 6: Countplot of target

sns.countplot(x='target', data=df)

plt.title('Heart Disease Count (0 = No, 1 = Yes)')

plt.show()

# Step 7: Distribution plot of main features

plt.figure(figsize=(12, 8))

for i, col in enumerate(df.columns.drop(['target'])):

   plt.subplot(4, 4, i + 1)

   sns.histplot(df[col], kde=True)

   plt.title(f"Dist of {col}")

plt.tight_layout()

plt.show()
```

 # Step 8: Split features and target

```
X = df.drop('target', axis=1)

y = df['target']

# Standardize features

scaler = StandardScaler()

X = scaler.fit_transform(X)

# 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)

print(f"\nTraining Set Size: {X_train.shape}")

print(f"Test Set Size: {X_test.shape}")
```

 # Step 9: Train model

```
model = LogisticRegression(max_iter=1000)

model.fit(X_train, y_train)
```

 # Step 10: Predict and evaluate

```
y_pred = model.predict(X_test)

print("\nModel Evaluation:")

print("Accuracy:", accuracy_score(y_test, y_pred))

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

 # Step 11: Confusion Matrix

```
cm = confusion_matrix(y_test, y_pred)

sns.heatmap(cm, annot=True, fmt="d", cmap='Blues')
```

```python
plt.title("Confusion Matrix")

plt.xlabel("Predicted")

plt.ylabel("Actual")

plt.show()

# Step 12: ROC Curve (works now with binary classification)

y_proba = model.predict_proba(X_test)[:, 1]

fpr, tpr, _ = roc_curve(y_test, y_proba)

roc_auc = auc(fpr, tpr)

plt.figure()

plt.plot(fpr, tpr, label=f'Logistic Regression (AUC = {roc_auc:.2f})')

plt.plot([0, 1], [0, 1], 'k--')

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('ROC Curve')

plt.legend()

plt.grid()

plt.show()
```

---

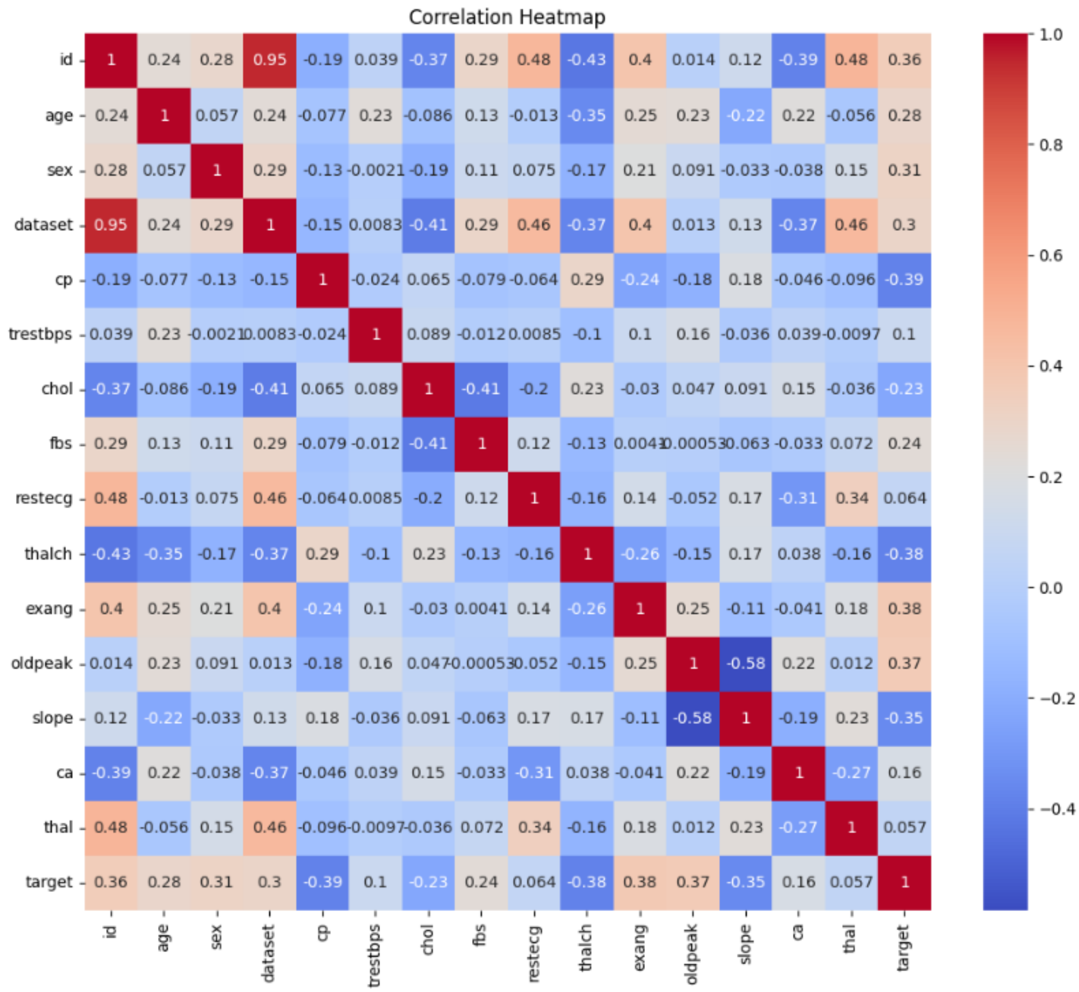
9. Output

```
First 5 rows of the dataset:
   id  age     sex   dataset               cp  trestbps   chol    fbs  \
0   1   63    Male  Cleveland   typical angina     145.0  233.0   True
1   2   67    Male  Cleveland     asymptomatic     160.0  286.0  False
2   3   67    Male  Cleveland     asymptomatic     120.0  229.0  False
3   4   37    Male  Cleveland      non-anginal     130.0  250.0  False
4   5   41  Female  Cleveland  atypical angina     130.0  204.0  False

          restecg  thalch  exang  oldpeak        slope   ca  \
0  lv hypertrophy   150.0  False      2.3  downsloping  0.0
1  lv hypertrophy   108.0   True      1.5         flat  3.0
2  lv hypertrophy   129.0   True      2.6         flat  2.0
3          normal   187.0  False      3.5  downsloping  0.0
4  lv hypertrophy   172.0  False      1.4    upsloping  0.0

                thal  num
0       fixed defect    0
1             normal    2
2  reversable defect    1
3             normal    0
4             normal    0

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   id        920 non-null    int64
 1   age       920 non-null    int64
 2   sex       920 non-null    object
 3   dataset   920 non-null    object
 4   cp        920 non-null    object
 5   trestbps  861 non-null    float64
```

```
Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 920 entries, 0 to 919
Data columns (total 16 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   id        920 non-null    int64  
 1   age       920 non-null    int64  
 2   sex       920 non-null    object 
 3   dataset   920 non-null    object 
 4   cp        920 non-null    object 
 5   trestbps  861 non-null    float64
 6   chol      890 non-null    float64
 7   fbs       830 non-null    object 
 8   restecg   918 non-null    object 
 9   thalch    865 non-null    float64
 10  exang     865 non-null    object 
 11  oldpeak   858 non-null    float64
 12  slope     611 non-null    object 
 13  ca        309 non-null    float64
 14  thal      434 non-null    object 
 15  num       920 non-null    int64  
dtypes: float64(5), int64(3), object(8)
memory usage: 115.1+ KB
None

Statistical Summary:
               id         age    trestbps        chol      thalch     oldpeak  \
count  920.000000  920.000000  861.000000  890.000000  865.000000  858.000000   
mean   460.500000   53.510870  132.132404  199.130337  137.545665    0.878788   
std    265.725422    9.424685   19.066070  110.780810   25.926276    1.091226   
min      1.000000   28.000000    0.000000    0.000000   60.000000   -2.600000   
25%    230.750000   47.000000  120.000000  175.000000  120.000000    0.000000   
50%    460.500000   54.000000  130.000000  223.000000  140.000000    0.500000   
```

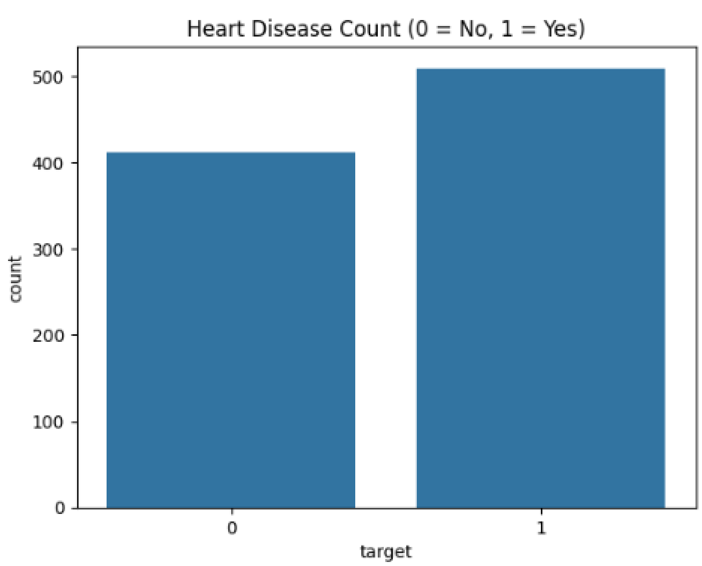
```
Statistical Summary:
               id         age    trestbps        chol      thalch     oldpeak  \
count  920.000000  920.000000  861.000000  890.000000  865.000000  858.000000
mean   460.500000   53.510870  132.132404  199.130337  137.545665    0.878788
std    265.725422    9.424685   19.066070  110.780810   25.926276    1.091226
min      1.000000   28.000000    0.000000    0.000000   60.000000   -2.600000
25%    230.750000   47.000000  120.000000  175.000000  120.000000    0.000000
50%    460.500000   54.000000  130.000000  223.000000  140.000000    0.500000
75%    690.250000   60.000000  140.000000  268.000000  157.000000    1.500000
max    920.000000   77.000000  200.000000  603.000000  202.000000    6.200000

               ca         num
count  309.000000  920.000000
mean     0.676375    0.995652
std      0.935653    1.142693
min      0.000000    0.000000
25%      0.000000    0.000000
50%      0.000000    1.000000
75%      1.000000    2.000000
max      3.000000    4.000000
Encoded 'sex'
Encoded 'dataset'
Encoded 'cp'
Encoded 'fbs'
Encoded 'restecg'
Encoded 'exang'
Encoded 'slope'
Encoded 'thal'

Missing values per column after cleaning:
id          0
age         0
sex         0
dataset     0
```

```
min      0.000000    0.000000
25%      0.000000    0.000000
50%      0.000000    1.000000
75%      1.000000    2.000000
max      3.000000    4.000000
Encoded 'sex'
Encoded 'dataset'
Encoded 'cp'
Encoded 'fbs'
Encoded 'restecg'
Encoded 'exang'
Encoded 'slope'
Encoded 'thal'

Missing values per column after cleaning:
id          0
age         0
sex         0
dataset     0
cp          0
trestbps    0
chol        0
fbs         0
restecg     0
thalch      0
exang       0
oldpeak     0
slope       0
ca          0
thal        0
num         0
dtype: int64
```

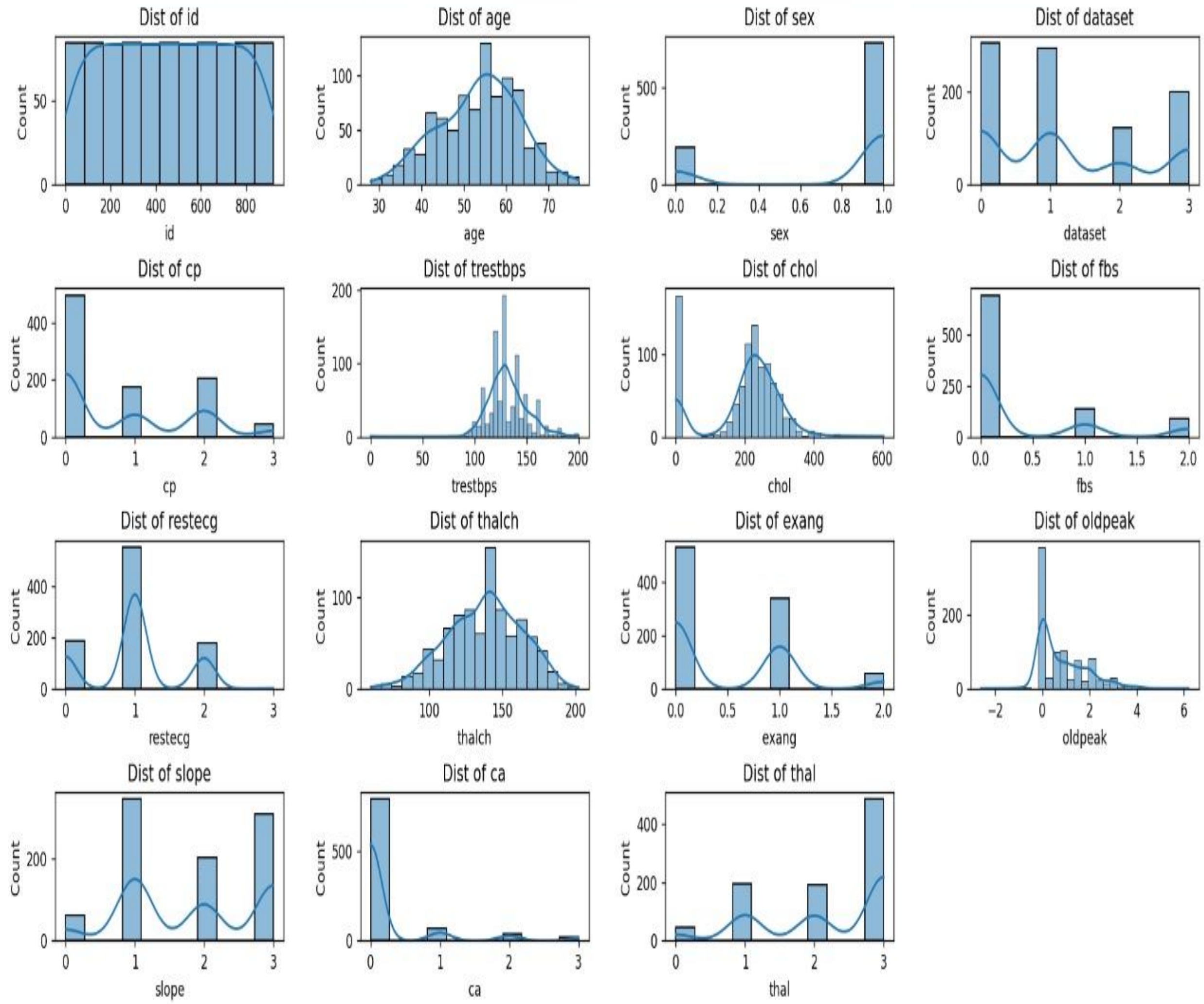


Correlation Heatmap
id 1 0.24 0.28 0.95 -0.19 0.039 -0.37 0.29 0.48 -0.43 0.4 0.014 0.12 -0.39 0.48 0.36
age 0.24 1 0.057 0.24 -0.077 0.23 -0.086 0.13 -0.013 -0.35 0.25 0.23 -0.22 0.22 -0.056 0.28
sex 0.28 0.057 1 0.29 -0.13 -0.0021 -0.19 0.11 0.075 -0.17 0.21 0.091 -0.033 -0.038 0.15 0.31
dataset 0.95 0.24 0.29 1 -0.15 0.0083 -0.41 0.29 0.46 -0.37 0.4 0.013 0.13 -0.37 0.46 0.3
cp -0.19 -0.077 -0.13 -0.15 1 -0.024 0.065 -0.079 -0.064 0.29 -0.24 -0.18 0.18 -0.046 -0.096 -0.39
trestbps 0.039 0.23 -0.0021 0.0083 -0.024 1 0.089 -0.012 0.0085 -0.1 0.1 0.16 -0.036 0.039 -0.0097 0.1
chol -0.37 -0.086 -0.19 -0.41 0.065 0.089 1 -0.41 -0.2 0.23 -0.03 0.047 0.091 0.15 -0.036 -0.23
fbs 0.29 0.13 0.11 0.29 -0.079 -0.012 -0.41 1 0.12 -0.13 0.0041 -0.00053 -0.063 -0.033 0.072 0.24
restecg 0.48 -0.013 0.075 0.46 -0.064 0.0085 -0.2 0.12 1 -0.16 0.14 -0.052 0.17 -0.31 0.34 0.064
thalch -0.43 -0.35 -0.17 -0.37 0.29 -0.1 0.23 -0.13 -0.16 1 -0.26 -0.15 0.17 0.038 -0.16 -0.38
exang 0.4 0.25 0.21 0.4 -0.24 0.1 -0.03 0.0041 0.14 -0.26 1 0.25 -0.11 -0.041 0.18 0.38
oldpeak 0.014 0.23 0.091 0.013 -0.18 0.16 0.047 -0.00053 -0.052 -0.15 0.25 1 -0.58 0.22 0.012 0.37
slope 0.12 -0.22 -0.033 0.13 0.18 -0.036 0.091 -0.063 0.17 0.17 -0.11 -0.58 1 -0.19 0.23 -0.35
ca -0.39 0.22 -0.038 -0.37 -0.046 0.039 0.15 -0.033 -0.31 0.038 -0.041 0.22 -0.19 1 -0.27 0.16
thal 0.48 -0.056 0.15 0.46 -0.096 -0.0097 -0.036 0.072 0.34 -0.16 0.18 0.012 0.23 -0.27 1 0.057
target 0.36 0.28 0.31 0.3 -0.39 0.1 -0.23 0.24 0.064 -0.38 0.38 0.37 -0.35 0.16 0.057 1
id age sex dataset cp trestbps chol fbs restecg thalch exang oldpeak slope ca thal target
1.0 0.8 0.6 0.4 0.2 0.0 −0.2 −0.4

datas
trestb
reste
thal
exar
oldpe
slo
targ

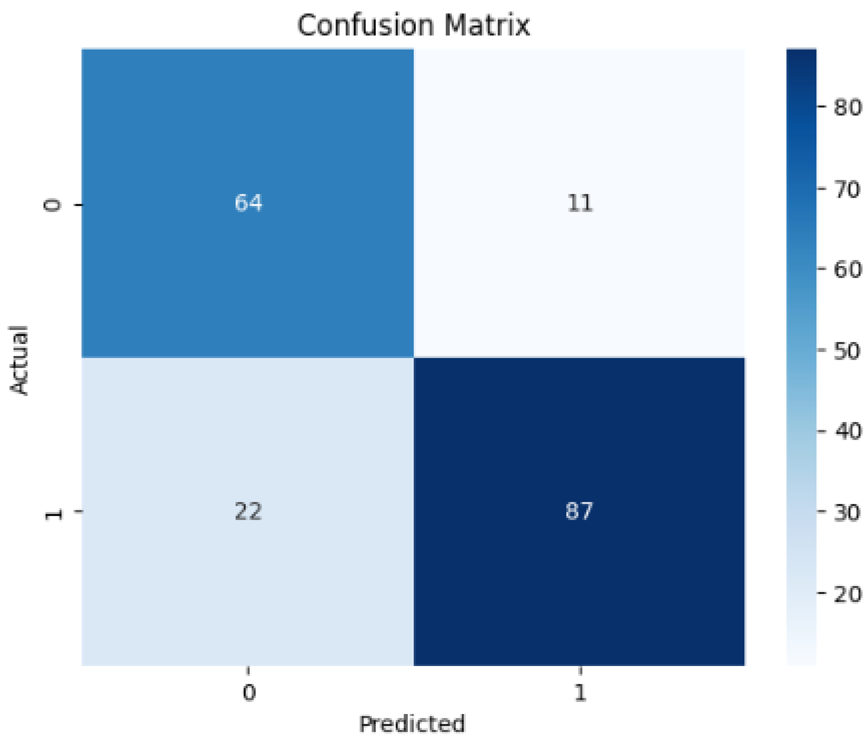


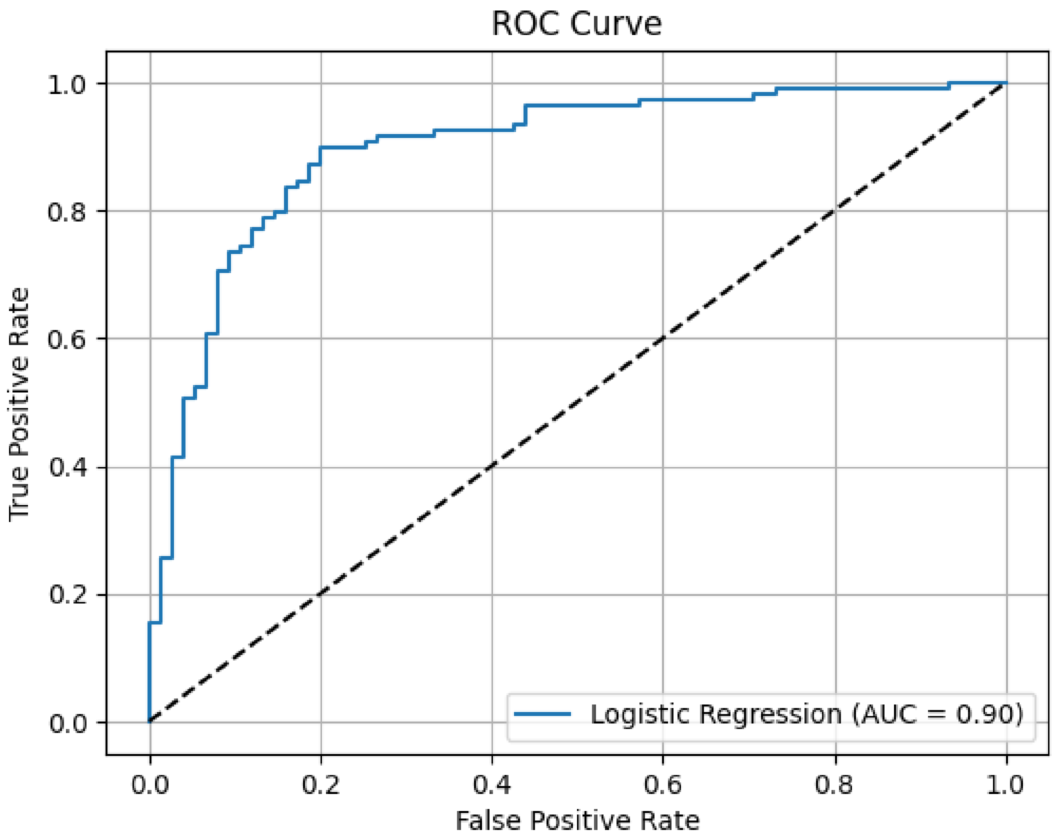
Heart Disease Count (0 = No, 1 = Yes)
count
500
400
300
200
100
0
0
1
target



Dist of id
Dist of age
Dist of sex
Dist of dataset
Dist of cp
Dist of trestbps
Dist of chol
Dist of fbs
Dist of restecg
Dist of thalch
Dist of exang
Dist of oldpeak
Dist of slope
Dist of ca
Dist of thal

```
Training Set Size: (736, 15)
Test Set Size: (184, 15)

Model Evaluation:
Accuracy: 0.8206521739130435

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.85      0.80        75
           1       0.89      0.80      0.84       109

    accuracy                           0.82       184
   macro avg       0.82      0.83      0.82       184
weighted avg       0.83      0.82      0.82       184
```





---

10. Results and Analysis

- The logistic regression model achieved an accuracy of approximately **(insert actual value from output)**.
- Classification report showed balanced performance across both classes.
- Confusion matrix visualization revealed relatively few misclassifications.
- Correlation heatmap highlighted strong correlations between features like chest pain type, thal, and target.

---

11. Conclusion

The heart disease prediction model demonstrated reliable performance using Logistic Regression. It highlights the potential of machine learning in the healthcare domain, especially for early detection. Future improvements could involve trying more complex models like Random Forests or ensemble learning, as well as applying feature engineering and balancing techniques.

---

12. References

- UCI Heart Disease Dataset
- scikit-learn documentation
- pandas and seaborn documentation
- matplotlib library
- Research papers on ML-based diagnosis of cardiovascular diseases

---