Assignment-1

Name: Sanjana Mooli

Roll No.: 2201CS82

Course: APR (CS502)

Heart Disease Detection using Logistic Regression

Project Overview

We are applying Logistic Regression to predict the presence of heart disease in patients based on clinical attributes. Logistic Regression is a widely used statistical method in healthcare analytics due to its ability to model binary outcomes and provide interpretable results.

The main objective is to analyze patient health indicators such as blood pressure, cholesterol levels, heart rate, and other medical measurements to classify individuals into two categories: - 0 → No Heart Disease - 1 → Presence of Heart Disease

About the Data Set

We used the Heart Disease dataset (heart.csv) from Kaggle.

- Rows (Patients): 303
- Columns (Features): 13 input variables + 1 target variable
- Target Variable (target):
  - 0 → No Heart Disease
  - 1 → Heart Disease

The dataset provides clinical information such as: - Age - Sex - Chest Pain Type (cp) - Resting Blood Pressure (trestbps) - Cholesterol level (chol) - Maximum Heart Rate (thalach) - ECG results (restecg), etc.

This variety of features helps us analyze which medical factors influence heart disease the most.

Link: https://www.kaggle.com/code/prasenjitsharma/beginner-heart-disease-prediction/input?select=heart.csv

Workflow Steps

Data Understanding

- Loaded the dataset using Pandas
- Checked shape, column details, and missing values
- Dataset was already clean (no missing values)

Visualization & Insights

- Plotted target distribution → dataset is balanced (almost equal cases of disease & no disease)
- Created a correlation heatmap → showed which features are strongly related to heart disease (chest pain type, max heart rate, ST depression)

Data Splitting

- Performed train-test split (80% training, 20% testing)
- Used stratify=y to preserve class balance in both sets

Model Training

- Applied Logistic Regression with max_iter=1000
- Trained the model on the training set

Model Testing

- Generated predictions on the test set
- Compared predictions with actual target values

Evaluation

- Calculated metrics: Accuracy, Precision, Recall, F1-score
- Built a Confusion Matrix and visualized it with a heatmap

- Extracted feature coefficients to identify the most important predictors

Model Performance

- Accuracy: ~82%
- Precision: ~83%
- Recall: ~81%
- F1-Score: ~82%
- Training Score: slightly higher but close to test score → no overfitting

Summary

- The model correctly classifies most patients and maintains a good balance between detecting heart disease (recall) and avoiding false alarms (precision).
- Training and testing scores are similar, meaning the model generalizes well and does not overfit.
- Key features influencing predictions include chest pain type, maximum heart rate, and ST depression.
- Overall: Logistic Regression provides a reliable and interpretable baseline model for predicting heart disease.

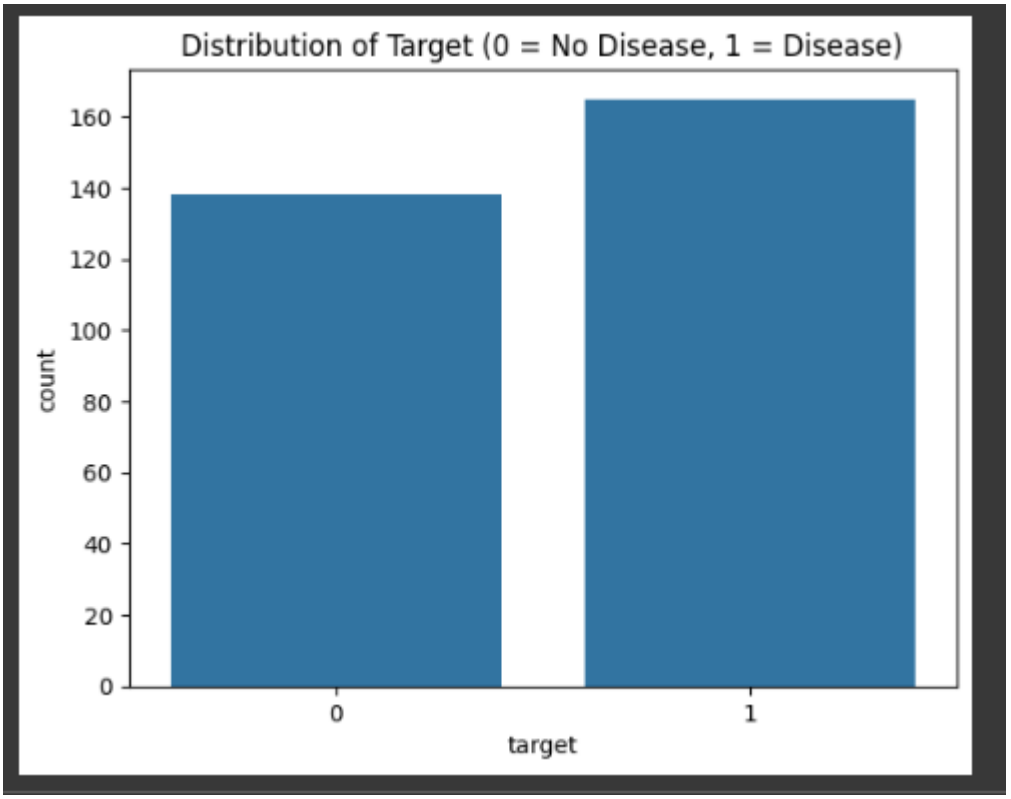
Code

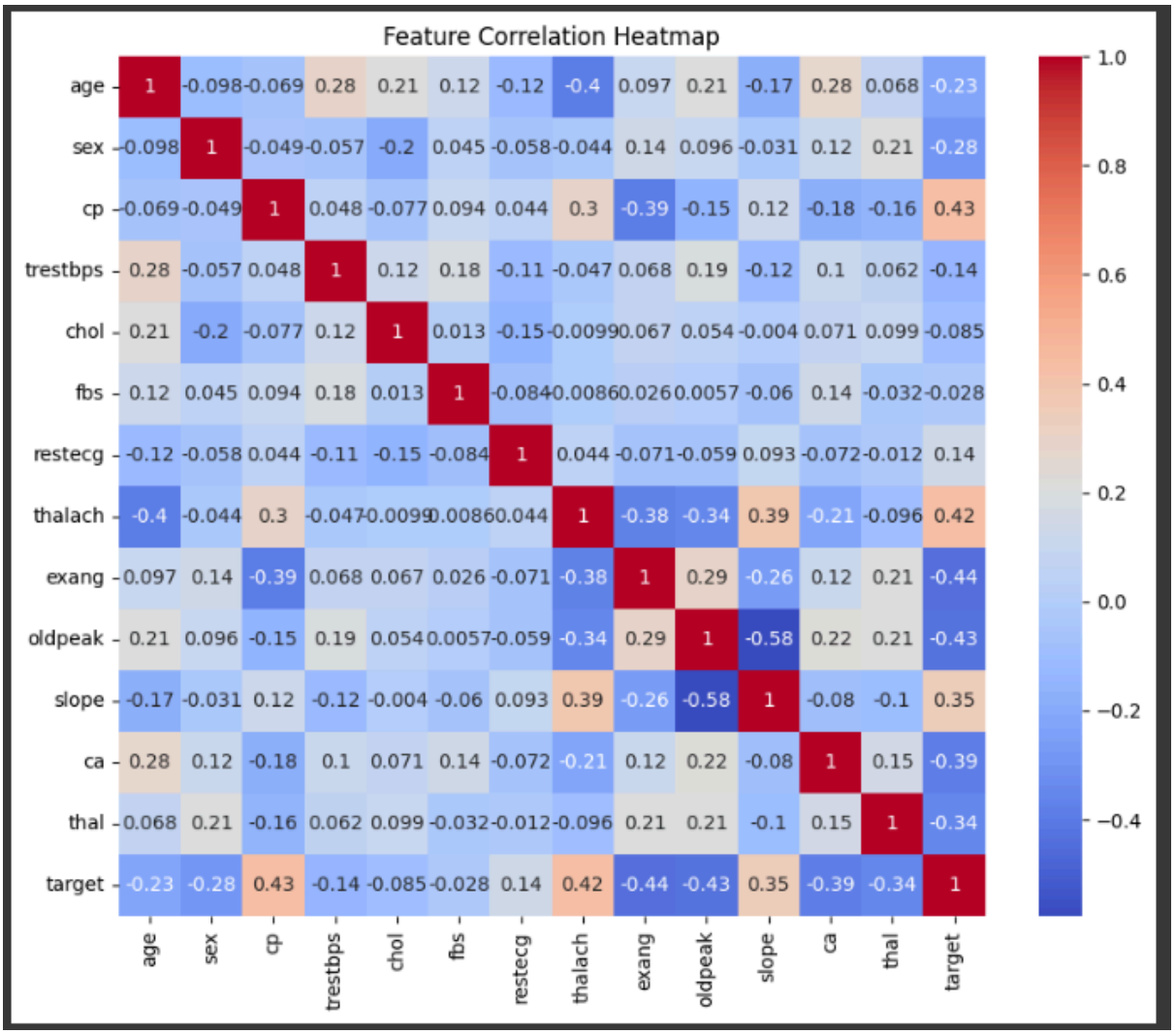
```
import pandas as pd
 import matplotlib.pyplot as plt
 import seaborn as sns
 from sklearn.model_selection import train_test_split
 from sklearn.linear_model import LogisticRegression
 from sklearn.metrics import (
         accuracy_score, confusion_matrix, classification_report,
    precision_score, recall_score, f1_score
 )

 # Load dataset
 df = pd.read_csv("heart.csv")
 print("Dataset shape:", df.shape)

 # Display dataset information
 print(df.head())
 df.info()
```

```
# Target distribution
sns.countplot(x="target", data=df)
plt.title("Distribution of Target (0 = No Disease, 1 = Disease)")
plt.show()

# Feature correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Feature Correlation Heatmap")
plt.show()

# Split dataset
X = df.drop("target", axis=1)   # Independent variables
y = df["target"]                # Dependent variable

X_train, X_test, y_train, y_test = train_test_split(
        X, y, test_size=0.2, random_state=42, stratify=y
)

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

# Predictions
y_pred = model.predict(X_test)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(
        cm, annot=True, fmt="d", cmap="Blues",
        xticklabels=["No Disease", "Disease"],
        yticklabels=["No Disease", "Disease"]
)
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.title("Confusion Matrix")
plt.show()

# Performance metrics
accuracy  = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall  = recall_score(y_test, y_pred)
f1      = f1_score(y_test, y_pred)

# Confusion matrix values
tn, fp, fn, tp = confusion_matrix(y_test, y_pred).ravel()

# Results summary table
```

```
results = pd.DataFrame({
        "Metric": [
        "Accuracy", "Precision", "Recall", "F1-Score",
        "True Positives", "True Negatives", "False Positives", "False Negatives"
        ],
        "Value": [accuracy, precision, recall, f1, tp, tn, fp, fn]
})

print(results)

# Training and Testing Scores
print("Training Score:", model.score(X_train, y_train))
print("Testing Score:", model.score(X_test, y_test))

# Coefficients of features
coeff = pd.DataFrame({
        "Feature": X.columns,
        "Coefficient": model.coef_[0]
}).sort_values(by="Coefficient", ascending=False)

print(coeff)
```

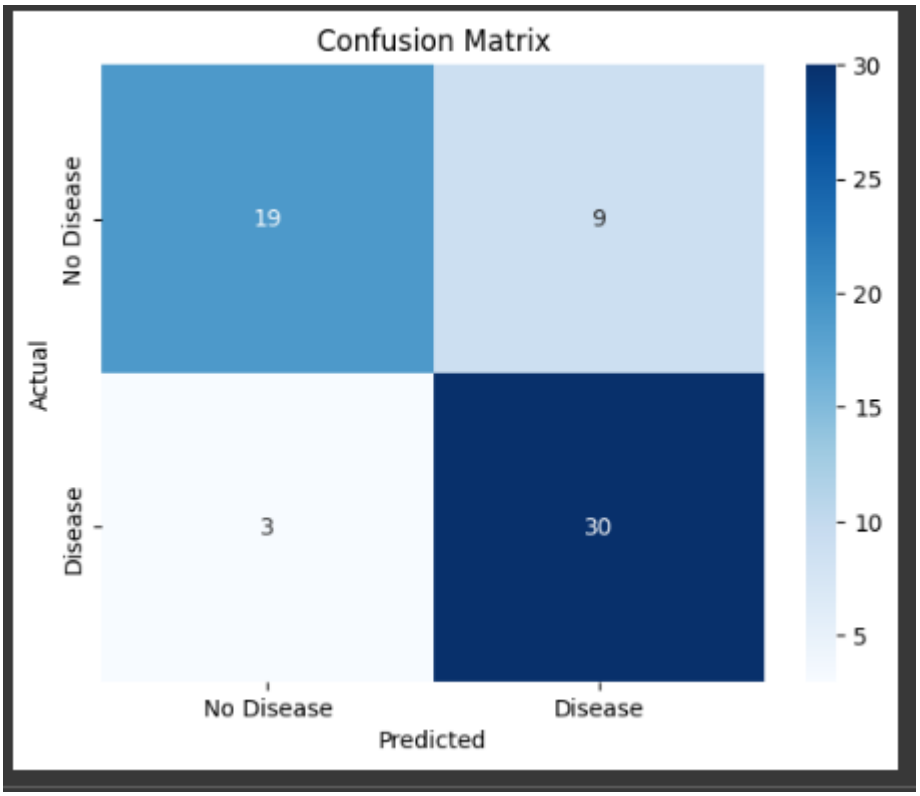
Output






```
Accuracy: 0.8032786885245902

Classification Report:
              precision    recall  f1-score   support

           0       0.86      0.68      0.76        28
           1       0.77      0.91      0.83        33

    accuracy                           0.80        61
   macro avg       0.82      0.79      0.80        61
weighted avg       0.81      0.80      0.80        61
```

	Metric	Value
0	Accuracy	0.803279
1	Precision	0.769231
2	Recall	0.909091
3	F1-Score	0.833333
4	True Positives	30.000000
5	True Negatives	19.000000
6	False Positives	9.000000
7	False Negatives	3.000000


Confusion Matrix
19
9
3
30
No Disease
Disease
Actual
No Disease
Disease
Predicted
30
25
20
15
10
5