```
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 StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

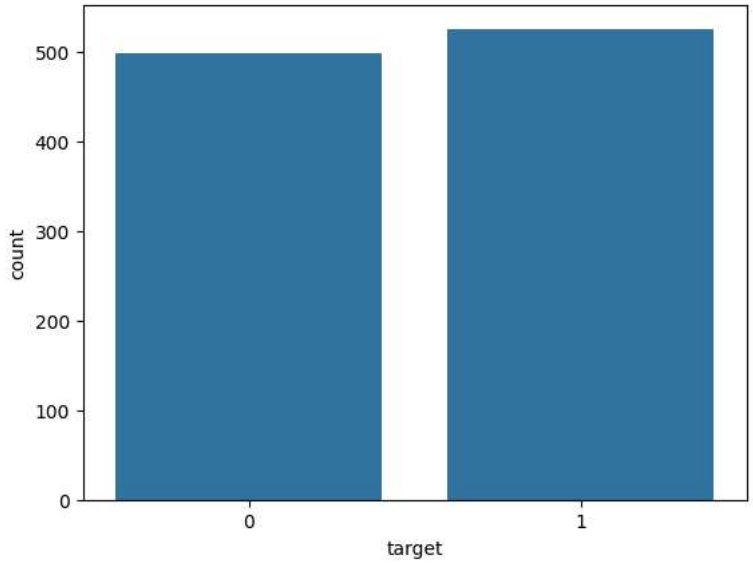
```
# Load dataset
df = pd.read_csv("/content/heart.csv")

# Preview data
print(df.head())
print(df.info())
```

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

   ca  thal  target
0   2     3       0
1   0     3       0
2   0     3       0
3   1     3       0
4   3     2       0
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
None
```

```
print(df.isnull().sum())   # Check missing values
print(df.describe())       # Statistical summary

# Count of target classes
sns.countplot(x="target", data=df)
plt.show()
```

```
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
```

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal | target |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| count | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.00000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 | 1025.000000 |
| mean | 54.434146 | 0.695610 | 0.942439 | 131.611707 | 246.00000 | 0.149268 | 0.529756 | 149.114146 | 0.336585 | 1.071512 | 1.385366 | 0.754146 | 2.323902 | 0.513171 |
| std | 9.072290 | 0.460373 | 1.029641 | 17.516718 | 51.59251 | 0.356527 | 0.527878 | 23.005724 | 0.472772 | 1.175053 | 0.617755 | 1.030798 | 0.620660 | 0.500070 |
| min | 29.000000 | 0.000000 | 0.000000 | 94.000000 | 126.00000 | 0.000000 | 0.000000 | 71.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| 25% | 48.000000 | 0.000000 | 0.000000 | 120.000000 | 211.00000 | 0.000000 | 0.000000 | 132.000000 | 0.000000 | 0.000000 | 1.000000 | 0.000000 | 2.000000 | 0.000000 |
| 50% | 56.000000 | 1.000000 | 1.000000 | 130.000000 | 240.00000 | 0.000000 | 1.000000 | 152.000000 | 0.000000 | 0.800000 | 1.000000 | 0.000000 | 2.000000 | 1.000000 |
| 75% | 61.000000 | 1.000000 | 2.000000 | 140.000000 | 275.00000 | 0.000000 | 1.000000 | 166.000000 | 1.000000 | 1.800000 | 2.000000 | 1.000000 | 3.000000 | 1.000000 |
| max | 77.000000 | 1.000000 | 3.000000 | 200.000000 | 564.00000 | 1.000000 | 2.000000 | 202.000000 | 1.000000 | 6.200000 | 2.000000 | 4.000000 | 3.000000 | 1.000000 |




```
X = df.drop("target", axis=1)   # Features
y = df["target"]                # Target
```

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

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
```

```
X_test = scaler.transform(X_test)
```

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

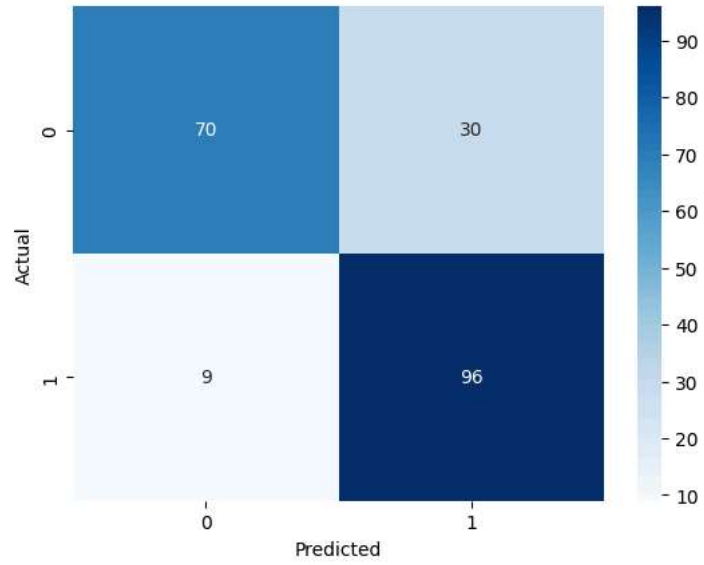
▾ LogisticRegression

```
LogisticRegression()
```

```
y_pred = model.predict(X_test)
```

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

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()

# Classification Report
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8097560975609757
```




```
              precision    recall  f1-score   support

           0       0.89      0.70      0.78       100
           1       0.76      0.91      0.83       105

    accuracy                           0.81       205
   macro avg       0.82      0.81      0.81       205
weighted avg       0.82      0.81      0.81       205
```

```
# Example input (values must match dataset features order)
new_data = np.array([[63,1,3,145,233,1,0,150,0,2.3,0,0,1]])

# Scale input
new_data_scaled = scaler.transform(new_data)

# Prediction
prediction = model.predict(new_data_scaled)
print("Prediction (1=Heart Disease, 0=No Disease):", prediction)
```

```
Prediction (1=Heart Disease, 0=No Disease): [1]
/usr/local/lib/python3.12/dist-packages/sklearn/utils/validation.py:2739: UserWarning: X does not have valid feature names, but Sta
  warnings.warn(
```