```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, accuracy_score, mean_squared_error

```

```
pd.set_option("display.max_columns", 200)
```

Heart DATASET

```
heart_df = pd.read_csv('https://raw.githubusercontent.com/Jatansahu/DEEP_LEARNING_ASSIGNMENTS/main/LAB
heart_df.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
heart_df.shape
```

```
(303, 14)
```

```
scaler = StandardScaler()
```

```
heart_df.info()
```

```
<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
```

```
heart_df.isnull().sum()
```

```
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
```

```
heart_df.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slop
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.00000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.39934
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.61622
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.00000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.00000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.00000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.00000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.00000

Correlation plot

```
heart_df.corr()
```

	age	sex	cp	trestbps	chol	fbs	restecg	tha
age	1.000000	-0.098447	-0.068653	0.279351	0.213678	0.121308	-0.116211	-0.39
sex	-0.098447	1.000000	-0.049353	-0.056769	-0.197912	0.045032	-0.058196	-0.04
cp	-0.068653	-0.049353	1.000000	0.047608	-0.076904	0.094444	0.044421	0.29
trestbps	0.279351	-0.056769	0.047608	1.000000	0.123174	0.177531	-0.114103	-0.04
chol	0.213678	-0.197912	-0.076904	0.123174	1.000000	0.013294	-0.151040	-0.00
fbs	0.121308	0.045032	0.094444	0.177531	0.013294	1.000000	-0.084189	-0.00
restecg	-0.116211	-0.058196	0.044421	-0.114103	-0.151040	-0.084189	1.000000	0.04
thalach	-0.398522	-0.044020	0.295762	-0.046698	-0.009940	-0.008567	0.044123	1.00
exang	0.096801	0.141664	-0.394280	0.067616	0.067023	0.025665	-0.070733	-0.37
oldpeak	0.210013	0.096093	-0.149230	0.193216	0.053952	0.005747	-0.058770	-0.34
slope	-0.168814	-0.030711	0.119717	-0.121475	-0.004038	-0.059894	0.093045	0.38
ca	0.276326	0.118261	-0.181053	0.101389	0.070511	0.137979	-0.072042	-0.21
thal	0.068001	0.210041	-0.161736	0.062210	0.098803	-0.032019	-0.011981	-0.09
target	-0.225439	-0.280937	0.433798	-0.144931	-0.085239	-0.028046	0.137230	0.42

```
X = heart_df.drop('target', axis=1)
y = heart_df['target']
```

Scaling the columns

```
X = scaler.fit_transform(X)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=5)
```

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

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

```
LogisticRegression()
```

```
y_pred = classifier.predict(X_test)
print(f"Accuracy Score on Test data: {accuracy_score(y_test, y_pred):.2f}")
print(f"Confusion Matrix:\n{confusion_matrix(y_test, y_pred)}")
```

```
Accuracy Score on Test data: 0.90
Confusion Matrix:
[[26  4]
 [ 2 29]]
```

```
class LogisticModel:
    def __init__(self):
        self.coef_ = None
        self.intercept_ = 0
        self.lr = 0.01
        self.max_iter = 1000
        self.e = 1e-5
        self.lambda_ = 0

    def sigmoid(self, z):
        return (1 / (1 + np.exp(-z)))

    def fit(self, X, y):
        self.coef_ = np.zeros(X.shape[1])
        W_prev = np.array([-1] * (X.shape[1]))

        # Iteration counter
        i = 0
        # Run the gradient descent algorithm for a maximum number of iterations or if the norm of the
        while i < self.max_iter and np.linalg.norm(self.coef_ - W_prev) > self.e:
            W_prev = self.coef_
            # Predict using current weights and bias
            y_cap = self.sigmoid(X @ self.coef_ + self.intercept_)

            # Calculate the gradients of the loss function with respect to the weights and bias term
            b_grad = np.sum(y_cap - y)
            W_grad = X.T @ (y_cap - y) + (self.lambda_ * self.coef_)

            # Update the weights and bias term using gradient descent
            self.coef_ = self.coef_ - self.lr * W_grad
            self.intercept_ = self.intercept_ - self.lr * b_grad

            i += 1

    def predict(self, X):
        y_pred = np.array(self.sigmoid(X @ self.coef_ +
                  self.intercept_) > 0.5, dtype=int)
        return y_pred
```

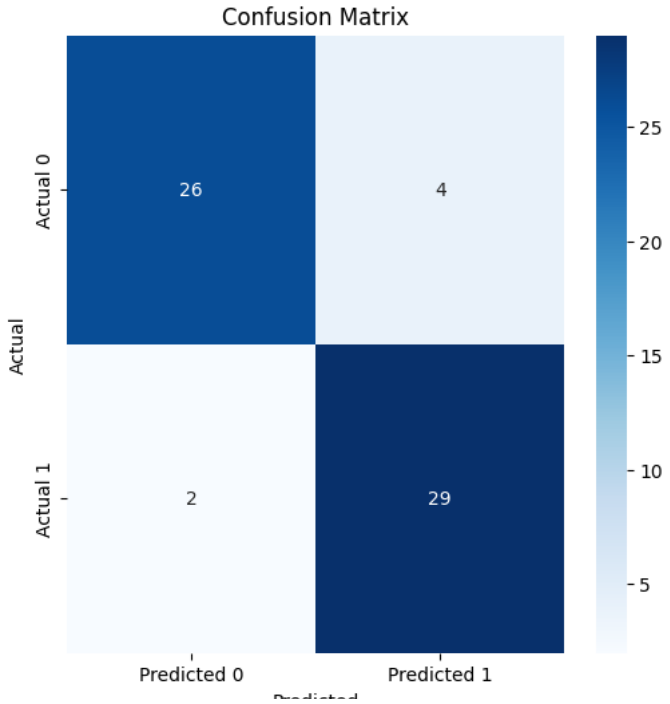
```
model = LogisticModel()
model.fit(X_train, y_train)
```

```
import seaborn as sns
y_pred = model.predict(X_test)
print(f"Accuracy Score on Test data: {accuracy_score(y_test, y_pred):.2f}")
cm = confusion_matrix(y_test, y_pred)

# Plot confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues",
            xticklabels=["Predicted 0", "Predicted 1"],
            yticklabels=["Actual 0", "Actual 1"])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()

```

```
Accuracy Score on Test data: 0.90
```




```
1
```

✓ 0s completed at 10:56 PM