[2]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
# Step 1: Load the dataset
# Using the Titanic dataset as an example
data = pd.read_csv('https://raw.githubusercontent.com/datasciencedojo/datasets/master/titanic.csv')
 # Display the first few rows of the dataset
print("Dataset preview:")
print(data.head())
```

```
Dataset preview:
   PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                Name     Sex   Age  SibSp  \
0                            Braund, Mr. Owen Harris    male  22.0      1
1  Cumings, Mrs. John Bradley (Florence Briggs Th...  female  38.0      1
2                             Heikkinen, Miss. Laina  female  26.0      0
3       Futrelle, Mrs. Jacques Heath (Lily May Peel)  female  35.0      1
4                           Allen, Mr. William Henry    male  35.0      0

   Parch            Ticket     Fare Cabin Embarked
0      0         A/5 21171   7.2500   NaN        S
1      0          PC 17599  71.2833   C85        C
2      0  STON/O2. 3101282   7.9250   NaN        S
3      0            113803  53.1000  C123        S
4      0            373450   8.0500   NaN        S
```

```python
# Step 2: Preprocess the data
# Selecting relevant features and target
selected_features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare']
data = data[selected_features + ['Survived']]
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})  # Encode categorical variable
data = data.dropna()  # Remove rows with missing values
X = data[selected_features]  # Features
y = data['Survived']         # Target
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```python
# Step 3: Initialize and train the Logistic Regression model
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

[12]:
```
 # Step 4: Make predictions
y_pred = model.predict(X_test)
 # Display predictions for the first 10 test samples
print("\nSample predictions:")
print(f"Predicted: {y_pred[:10]}")
print(f"Actual:    {y_test.values[:10]}")
```

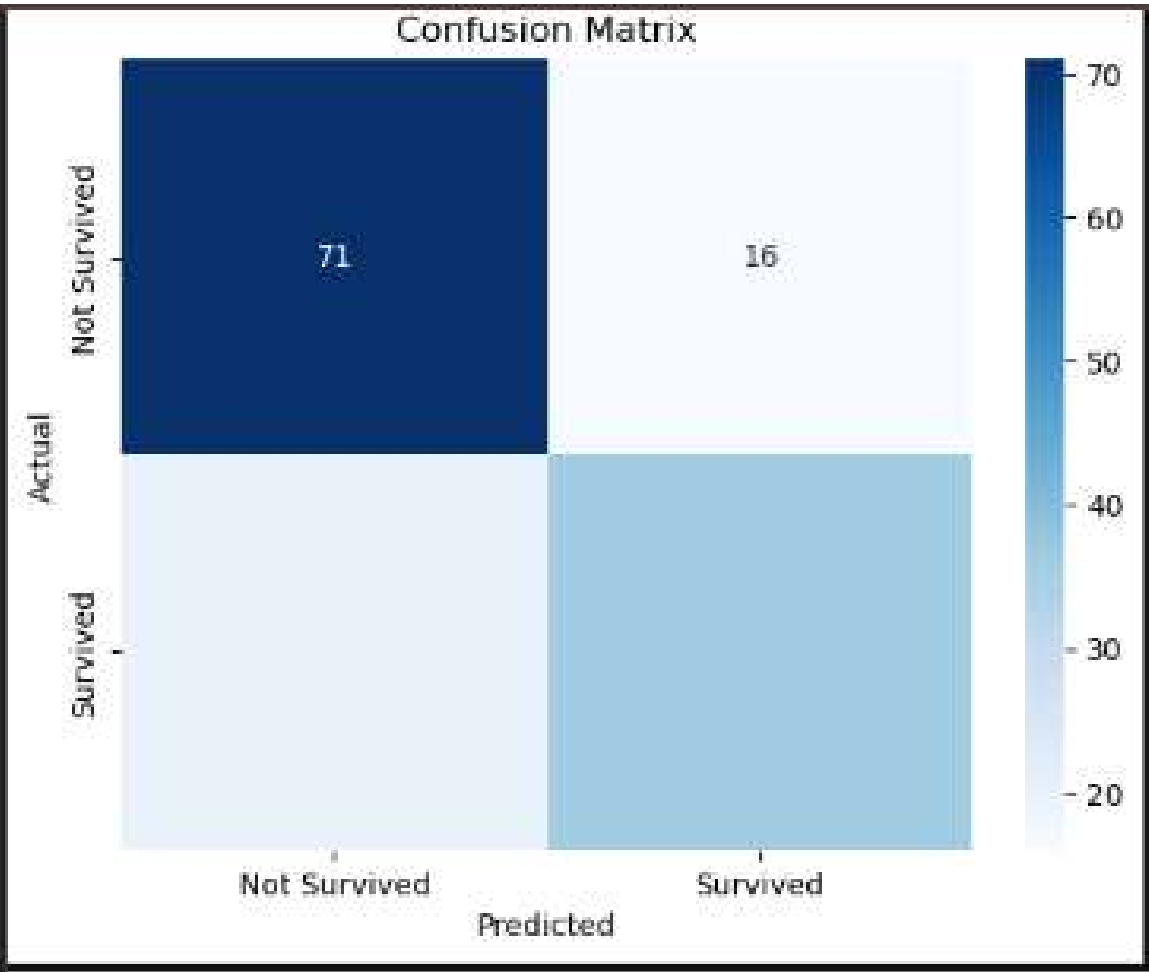
```
Sample predictions:
Predicted: [0 0 1 1 0 0 0 0 1 1]
Actual:    [0 1 1 1 0 1 1 1 0 0]
```

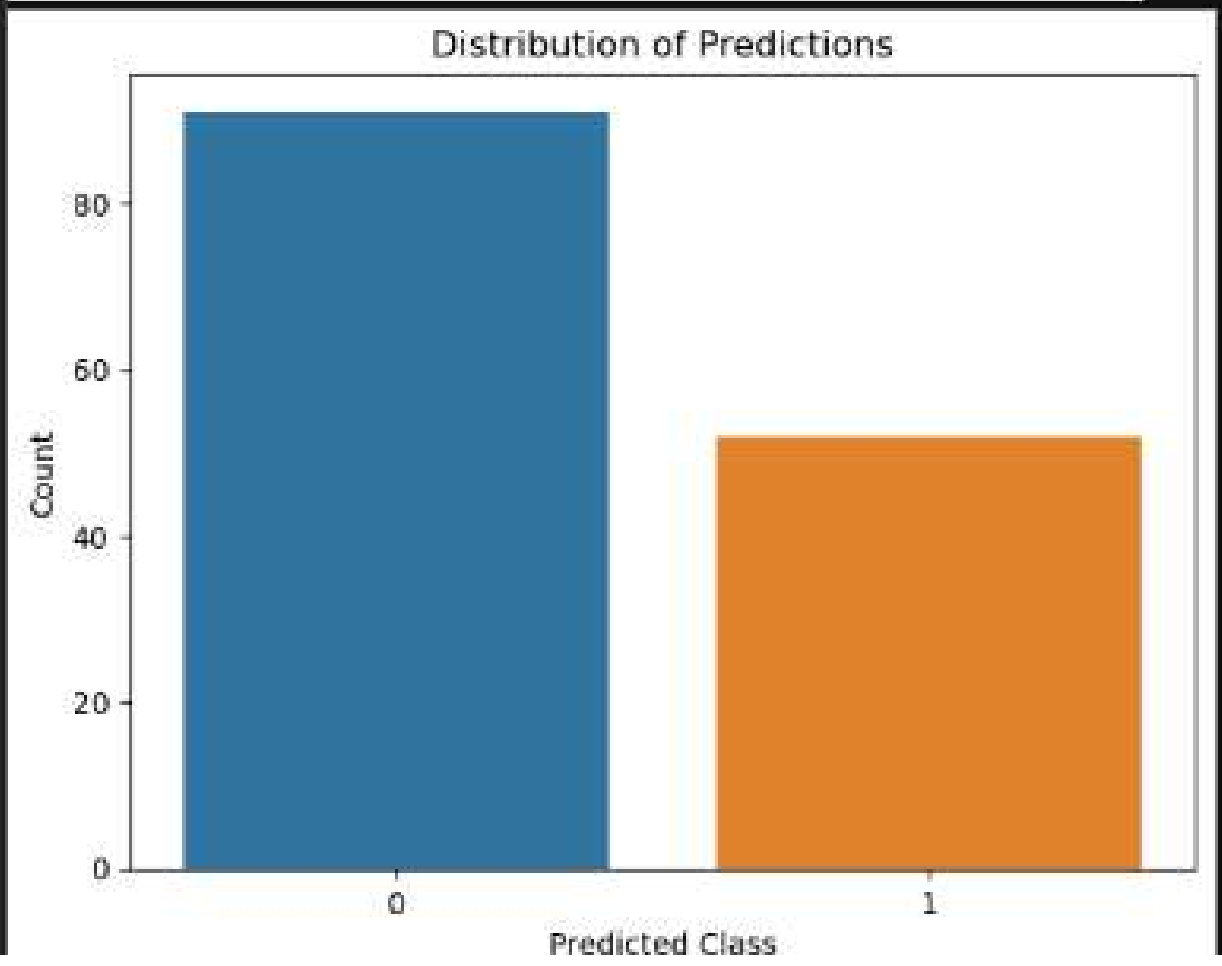
[14]:
```
 # Step 5: Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy on test data: {accuracy * 100:.2f}%")
 # Additional performance metrics
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy on test data: 74.83%

Classification Report:
              precision    recall  f1-score   support

           0       0.78      0.82      0.80        87
           1       0.69      0.64      0.67        56

    accuracy                           0.75       143
   macro avg       0.74      0.73      0.73       143
weighted avg       0.75      0.75      0.75       143
```

[20]:

```python
 # Visualizations
 # Confusion Matrix
conf_matrix = confusion_matrix(y_test, y_pred)
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues', xticklabels=['Not Survived', 'Survived'], yticklabels=['Not Survived', 'Survived'])
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
 # Distribution of predictions
sns.countplot(x=y_pred)
plt.title('Distribution of Predictions')
plt.xlabel('Predicted Class')
plt.ylabel('Count')
plt.show()
```


Confusion Matrix
Not Survived
Survived
Actual
71
16
70
60
50
40
30
20
Not Survived
Survived
Predicted


Distribution of Predictions
Count
80
60
40
20
0
0
1
Predicted Class