EXP 6: Implement a logistic regression model on given dataset and check the accuracy for test dataset.

Logistic Regression is a supervised machine learning algorithm used for binary classification problems. Unlike Linear Regression, it predicts the probability of a target variable belonging to a specific class by applying a logistic (sigmoid) function to model outcomes between 0 and 1.

Step1:Load the Dataset

The Titanic dataset is used for this example. It contains data on passengers, including demographic information (e.g., age, gender) and travel details (e.g., ticket class, fare). The goal is to predict whether a passenger survived or not (Survived column).

The head() function is used to understand the structure and contents of the dataset.

Key Features: The dataset includes variables such as Pclass (ticket class), Sex (gender), and Age, which are vital for survival prediction.

```
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
```

Step2. Preprocessing

Data preprocessing prepares the dataset for analysis:

Feature Selection: Relevant features (Pclass, Sex, Age, etc.) are chosen based on their importance in predicting survival.

Encoding Categorical Variables: The Sex column is converted into numeric values (0 for male, 1 for female).

Handling Missing Data: Rows with missing values in any of the selected features are removed to ensure consistency.

The dataset is then split into training and testing subsets (80%-20% split), with the training data used to fit the model and the testing data reserved for evaluation.

```
# 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)
```

```
<ipython-input-11-20f97133c5e4>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
```

```
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})  # Encode categorical variable
```

Step3. Model Training

A Logistic Regression model is instantiated and trained using the LogisticRegression class from sklearn. The algorithm learns the relationship between the features (X_train) and the target (y_train).

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

```
▸  LogisticRegression
```

Step 4. Making Predictions

The model predicts outcomes (y_pred) for the test dataset (X_test). These predictions indicate whether a passenger is likely to survive (1) or not (0).

```
# 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]}")
```

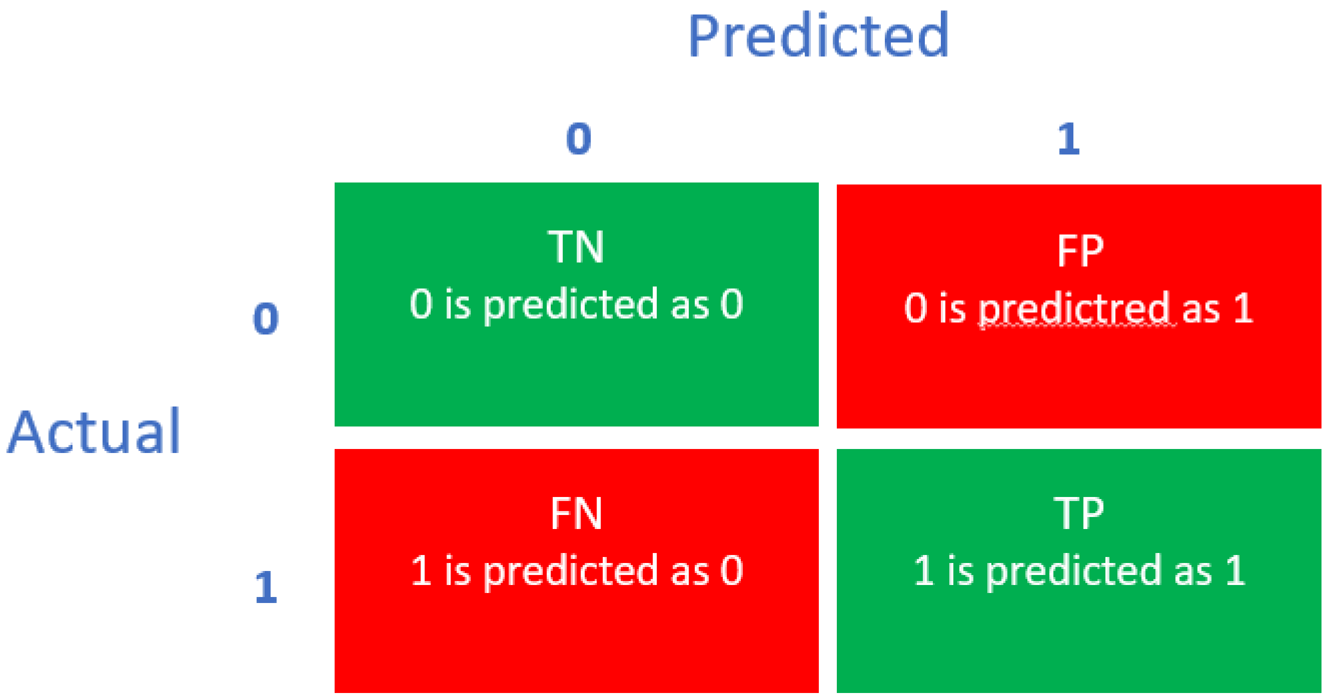
```
Sample predictions:
Predicted: [0 0 1 1 0 0 0 0 1 1]
Actual:    [0 1 1 1 0 1 1 1 0 0]
```

Step 5. Model Evaluation

Accuracy: Measures the percentage of correctly predicted labels in the test set.

Classification Report: Provides precision, recall, F1-score, and support for each class, offering insights into the model's performance.

Confusion Matrix: Visualizes the performance of the classification model, showing the counts of true positives, true negatives, false positives, and false negatives.



$$\text{specificity} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}}$$

$$\text{sensitivity} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}$$

$$\text{accuracy} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

```
# 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
```

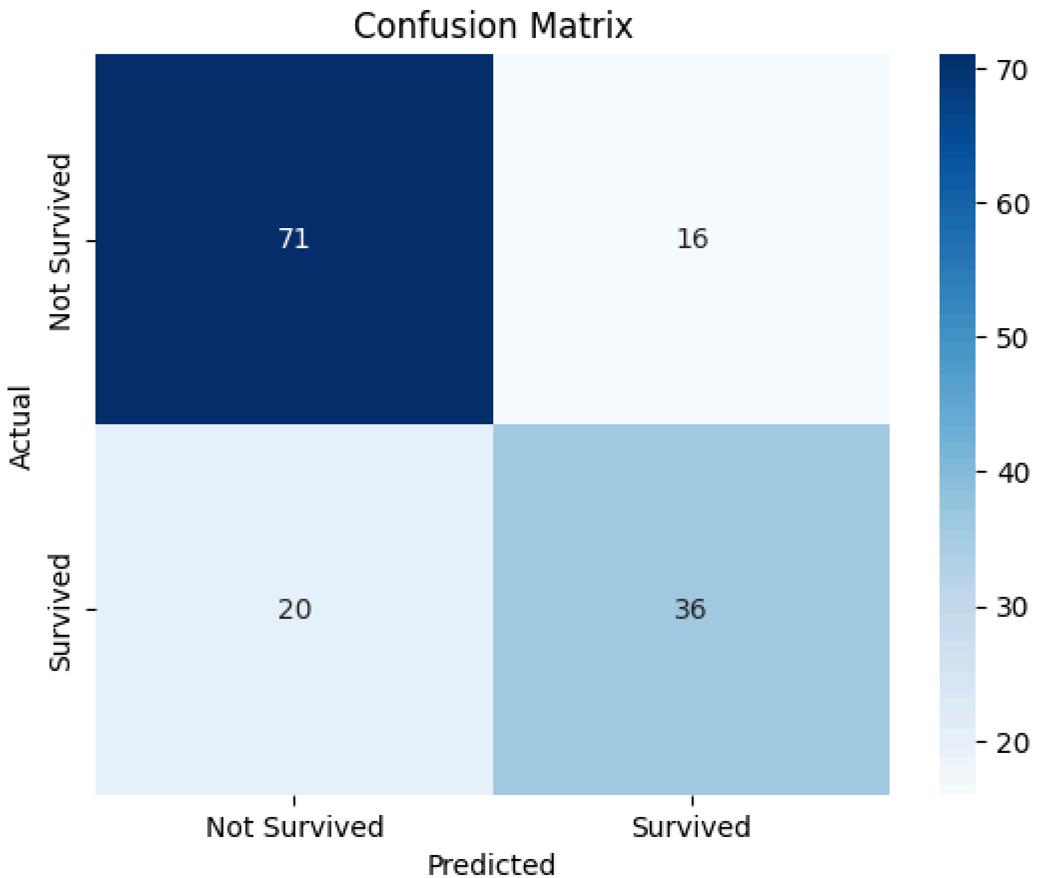
STEP 6. Visualizations

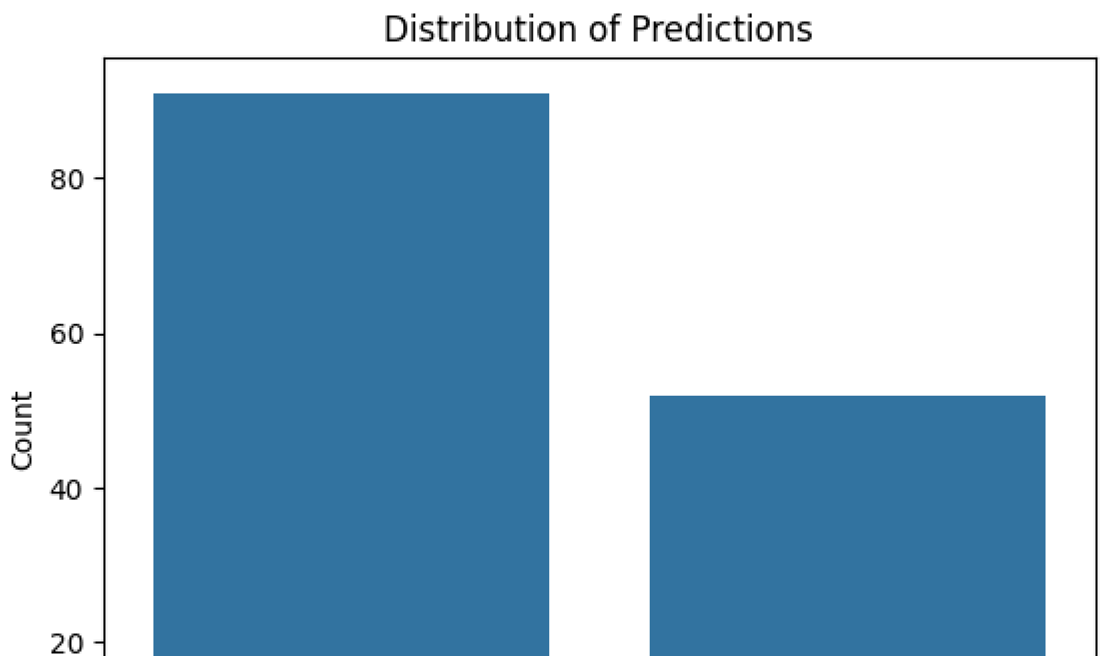
Confusion Matrix Heatmap: Displays the confusion matrix using a heatmap for an intuitive understanding of classification errors.

Prediction Distribution: A count plot of predicted classes illustrates the distribution of predictions (survived vs. not survived).

```
# 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
Actual
Not Survived
Survived
71
16
20
36
Not Survived
Survived
Predicted
70
60
50
40
30
20


Distribution of Predictions
Count
80
60
40
20


0
0
1
Predicted Class