```
# @title Import Required Libraries
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 LabelEncoder
from sklearn.metrics import accuracy_score, classification_report,
confusion_matrix
from sklearn.linear_model import LogisticRegression

# @title Load the Dataset
df = pd.read_csv("Titanic-Dataset.csv")
df.head()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 891,\n  \"fields\": [\
n    {\n      \"column\": \"PassengerId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 257,\n        \"min\": 1,\n
\"max\": 891,\n        \"num_unique_values\": 891,\n
\"samples\": [\n          710,\n          440,\n          841\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Survived\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 1,\n
\"num_unique_values\": 2,\n        \"samples\": [\n          1,\n
0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 0,\n        \"min\": 1,\n        \"max\": 3,\n
\"num_unique_values\": 3,\n        \"samples\": [\n          3,\n
1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Name\",\n      \"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 891,\n        \"samples\": [\n
\"Moubarek, Master. Halim Gonios (\\\"William George\\\")\",\n
\"Kvillner, Mr. Johan Henrik Johannesson\"\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Sex\",\n      \"properties\": {\n
\"dtype\": \"category\",\n        \"num_unique_values\": 2,\n
\"samples\": [\n          \"female\",\n          \"male\"\n        ],\
n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 14.526497332334044,\
n        \"min\": 0.42,\n        \"max\": 80.0,\n
\"num_unique_values\": 88,\n        \"samples\": [\n          0.75,\n
22.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"SibSp\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
```

\"std\": 1,\n   \"min\": 0,\n   \"max\": 8,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   1,\n
0\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Parch\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 0,\n   \"min\": 0,\n   \"max\": 6,\n
\"num_unique_values\": 7,\n   \"samples\": [\n   0,\n
1\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Ticket\",\n   \"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 681,\n   \"samples\": [\n
\"11774\",\n   \"248740\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Fare\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 49.693428597180905,\n
\"min\": 0.0,\n   \"max\": 512.3292,\n
\"num_unique_values\": 248,\n   \"samples\": [\n
11.2417,\n   51.8625\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Cabin\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 147,\n
\"samples\": [\n   \"D45\",\n   \"B49\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Embarked\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
3,\n   \"samples\": [\n   \"S\",\n   \"C\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   }\n]\n}","type":"dataframe","variable_name":"df"}

```
# @title Understand the Dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
```

```
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

```
df.describe()
```


```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 8,\n  \"fields\": [\n
{\n      \"column\": \"PassengerId\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 320.8159711429856,\n
\"min\": 1.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          891.0,\n          446.0,\n
668.5\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Survived\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 314.8713661874558,\n        \"min\":
0.0,\n        \"max\": 891.0,\n        \"num_unique_values\": 5,\n
\"samples\": [\n          0.3838383838383838,\n          1.0,\n
0.4865924542648585\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Pclass\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 314.2523437079693,\n        \"min\": 0.8360712409770513,\n
\"max\": 891.0,\n        \"num_unique_values\": 6,\n
\"samples\": [\n          891.0,\n          2.308641975308642,\n
3.0\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Age\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 242.9056731818781,\n        \"min\": 0.42,\n        \"max\":
714.0,\n        \"num_unique_values\": 8,\n        \"samples\": [\n
29.69911764705882,\n          28.0,\n          714.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"SibSp\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 314.4908277465442,\n
\"min\": 0.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
6,\n        \"samples\": [\n          891.0,\n
0.5230078563411896,\n          8.0\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Parch\",\n      \"properties\": {\
n        \"dtype\": \"number\",\n        \"std\": 314.65971717879,\n
\"min\": 0.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
5,\n        \"samples\": [\n          0.38159371492704824,\n
6.0,\n          0.8060572211299559\n        ],\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Fare\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 330.6256632228577,\n
\"min\": 0.0,\n        \"max\": 891.0,\n        \"num_unique_values\":
8,\n        \"samples\": [\n          32.204207968574636,\n
14.4542,\n          891.0\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe"}
```


```
df.isnull().sum()
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
# @title Data Cleaning
df['Age'].fillna(df['Age'].mean(), inplace=True)
```

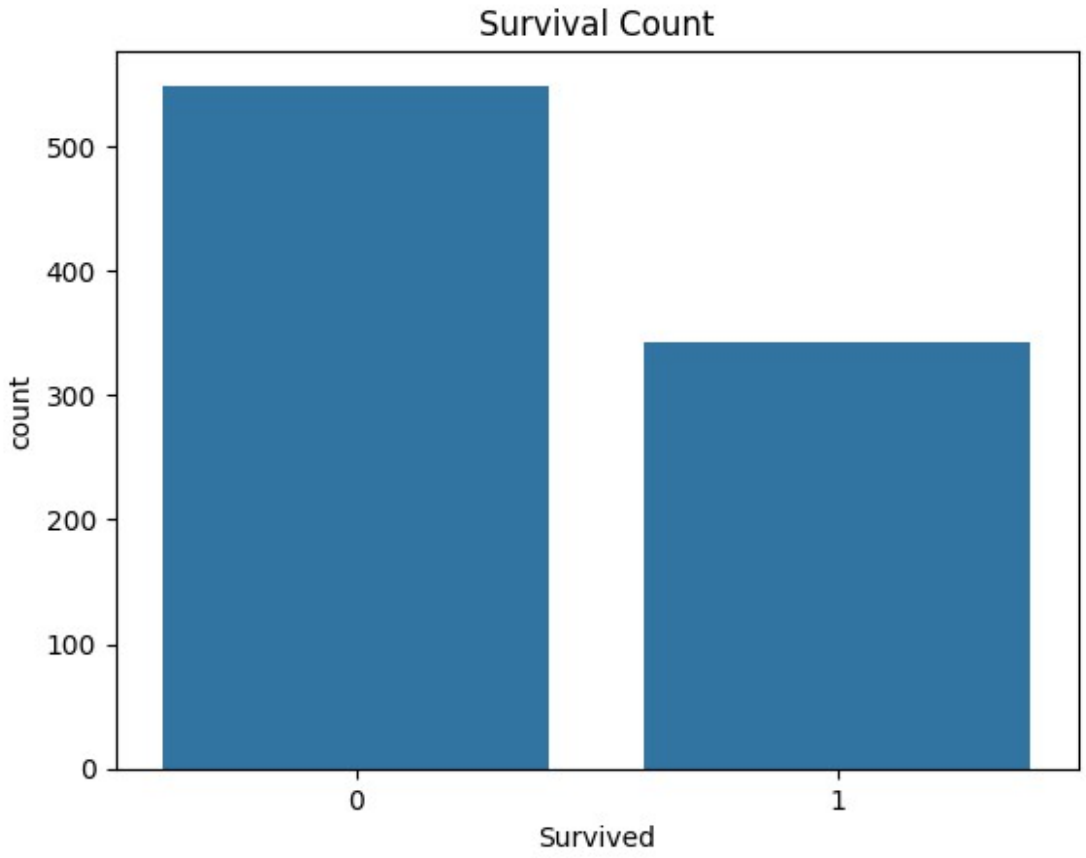
```
/tmp/ipython-input-1226013979.py:1: FutureWarning: A value is trying
to be set on a copy of a DataFrame or Series through chained
assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['Age'].fillna(df['Age'].mean(), inplace=True)
```

```
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

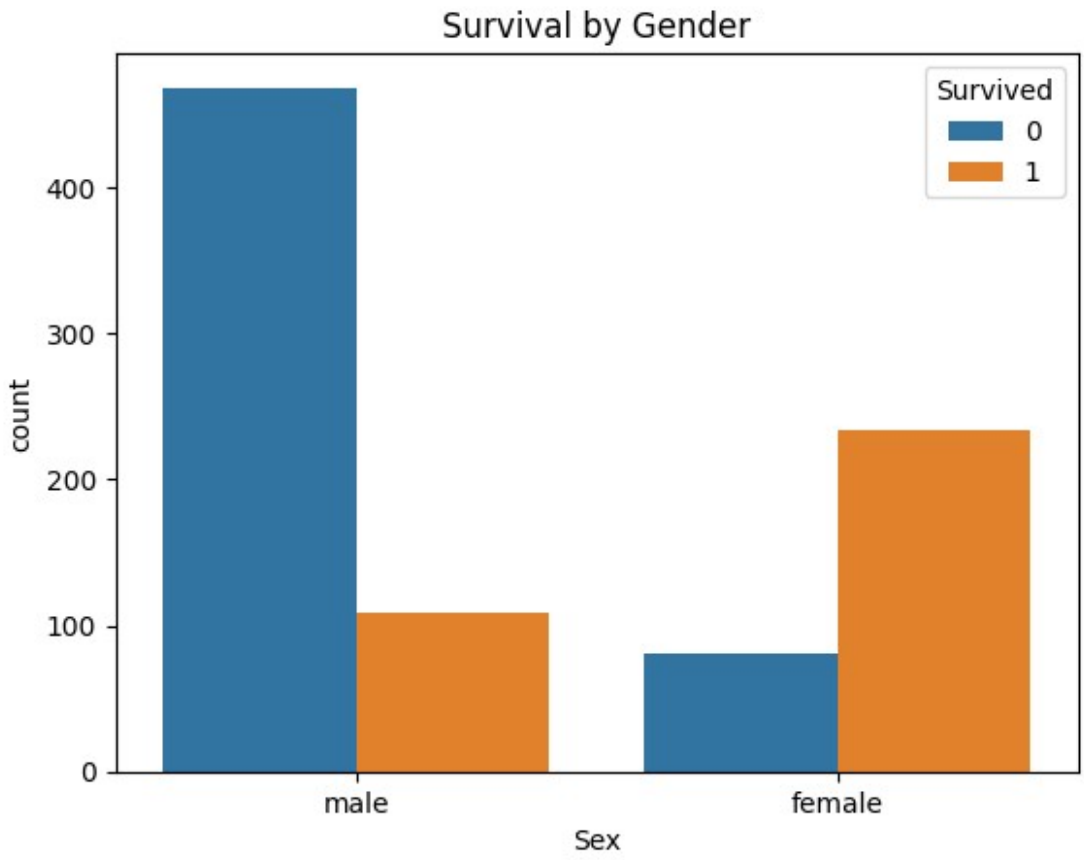
```
/tmp/ipython-input-411043493.py:1: FutureWarning: A value is trying to
be set on a copy of a DataFrame or Series through chained assignment
using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never
work because the intermediate object on which we are setting values
always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try
using 'df.method({col: value}, inplace=True)' or df[col] =
df[col].method(value) instead, to perform the operation inplace on the
original object.


  df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
```

```
df.drop(columns=['Cabin'], inplace=True)
```

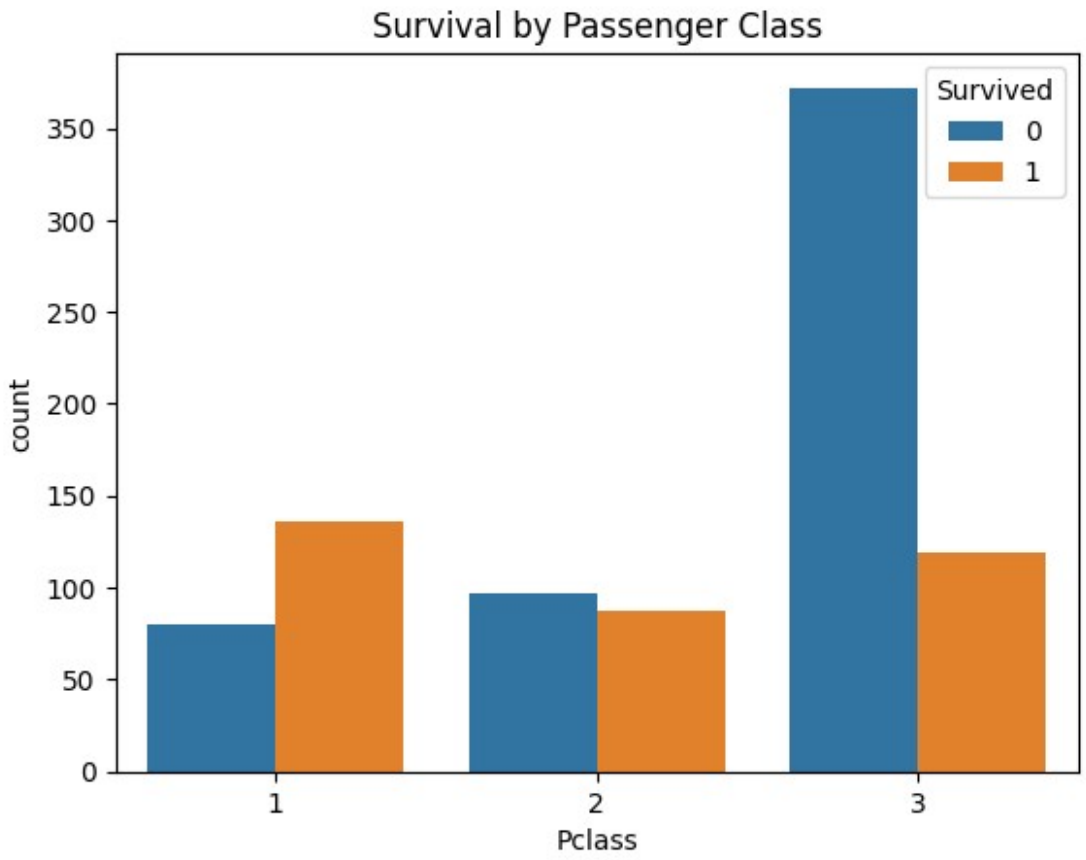
```
# @title Exploratory Data Analysis (EDA)
sns.countplot(x='Survived', data=df)
plt.title("Survival Count")
plt.show()
```




```
sns.countplot(x='Sex', hue='Survived', data=df)
plt.title("Survival by Gender")
plt.show()
```




```
sns.countplot(x='Pclass', hue='Survived', data=df)
plt.title("Survival by Passenger Class")
plt.show()
```




```
# @title Encode Categorical Variables
le = LabelEncoder()

df['Sex'] = le.fit_transform(df['Sex'])
df['Embarked'] = le.fit_transform(df['Embarked'])

# @title Feature Selection
X = df[['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']]
y = df['Survived']

# @title Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# @title Build the Model
model = LogisticRegression(max_iter=1000)
model.fit(X_train, y_train)
```

```
LogisticRegression(max_iter=1000)
```

```
# @title Model Prediction
y_pred = model.predict(X_test)
```

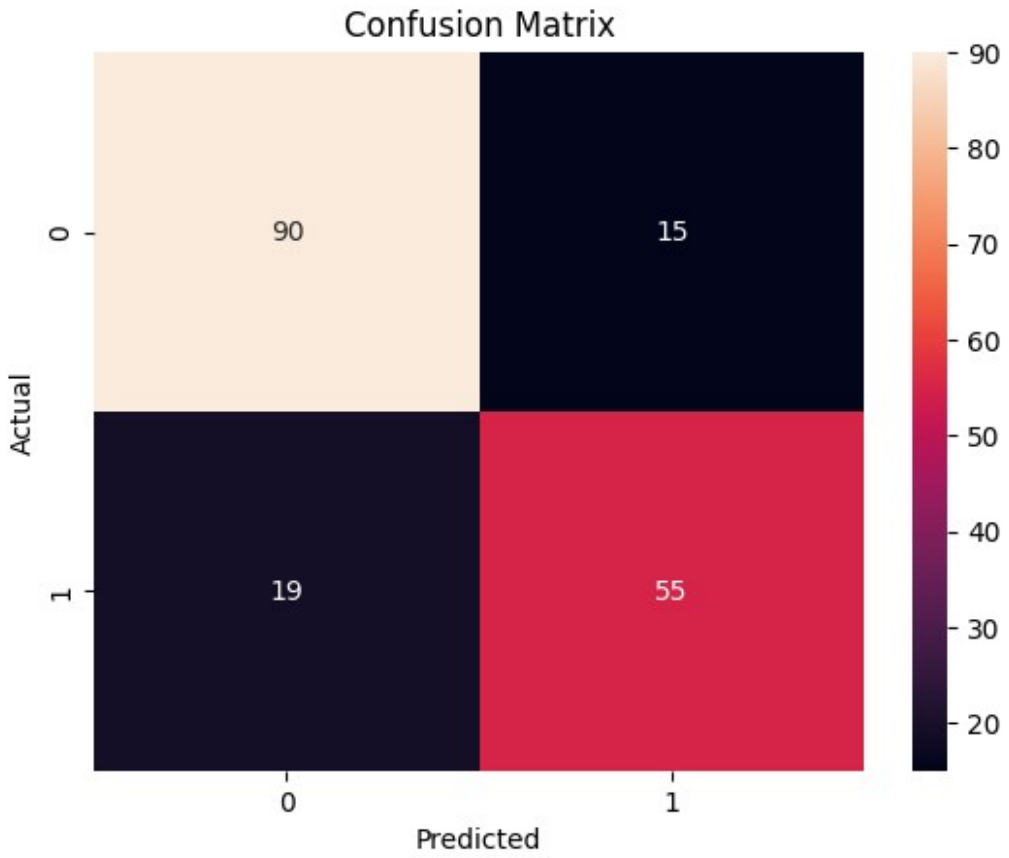
```python
# @title Model Evaluation
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.8100558659217877
```

```python
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.86      0.84       105
           1       0.79      0.74      0.76        74

    accuracy                           0.81       179
   macro avg       0.81      0.80      0.80       179
weighted avg       0.81      0.81      0.81       179
```

```python
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d')
plt.title("Confusion Matrix")
plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



```
# @title Conclusion
print("Logistic Regression model successfully predicts Titanic
survival.")
print("Gender, Passenger Class, and Fare play major roles in
survival.")
```

```
Logistic Regression model successfully predicts Titanic survival.
Gender, Passenger Class, and Fare play major roles in survival.
```