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

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True)
```

```
data =pd.read_csv('/content/Titanic-Dataset.csv')
```

```
print(data.head())
```

```
   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
```

```
print(data.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
None
```

```
print(data.describe())
```

```
       PassengerId    Survived      Pclass         Age       SibSp  \
count   891.000000  891.000000  891.000000  714.000000  891.000000
mean    446.000000    0.383838    2.308642   29.699118    0.523008
std     257.353842    0.486592    0.836071   14.526497    1.102743
min       1.000000    0.000000    1.000000    0.420000    0.000000
25%     223.500000    0.000000    2.000000   20.125000    0.000000
50%     446.000000    0.000000    3.000000   28.000000    0.000000
75%     668.500000    1.000000    3.000000   38.000000    1.000000
max     891.000000    1.000000    3.000000   80.000000    8.000000

            Parch        Fare
count  891.000000  891.000000
mean     0.381594   32.204208
std      0.806057   49.693429
min      0.000000    0.000000
25%      0.000000    7.910400
50%      0.000000   14.454200
75%      0.000000   31.000000
max      6.000000  512.329200
```

```
print(data.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
```

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

```
/tmp/ipython-input-1981374165.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chair
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are set

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = c


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

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

```
/tmp/ipython-input-4247733614.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chair
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are set

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = c


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

```
data.drop('Cabin', axis=1, inplace=True)
```

```
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})
```

```
data=pd.get_dummies(data, columns=['Embarked'], drop_first=True)
```

```
x = data.drop(['Survived', 'Name', 'Ticket'], axis=1)
y = data['Survived']
```

Gemini

```
from sklearn.model_selection import train_test_split

# Drop the 'Ticket' column from x before splitting
x = x.drop('Ticket', axis=1)

x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```

```
from sklearn.linear_model import LogisticRegression
```

```
model = LogisticRegression(max_iter=100)
```

```
model.fit(x_train, y_train) # Removed the redundant fit call
```

```
/usr/local/lib/python3.12/dist-packages/sklearn/linear_model/_logistic.py:465: ConvergenceWarning: lbfgs failed to converge (st
STOP: TOTAL NO. OF ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```

▾ LogisticRegression

LogisticRegression()

```
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
```

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

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

```
Accuracy: 0.8044692737430168
```

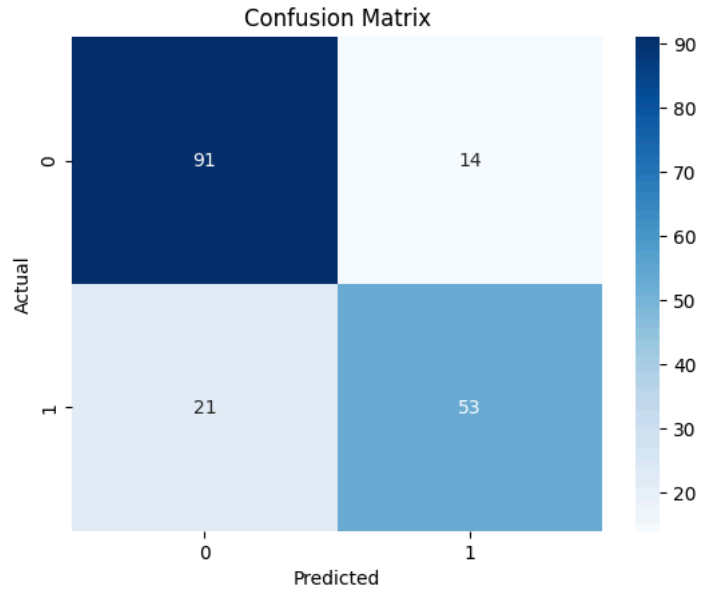
```
print("prediction for first 5 passengers:", model.predict(x_test[:5]))
```

```
prediction for first 5 passengers: [0 0 0 1 1]
```

```
print("actual:", y_test[:5].values)
```

```
actual: [1 0 0 1 1]
```

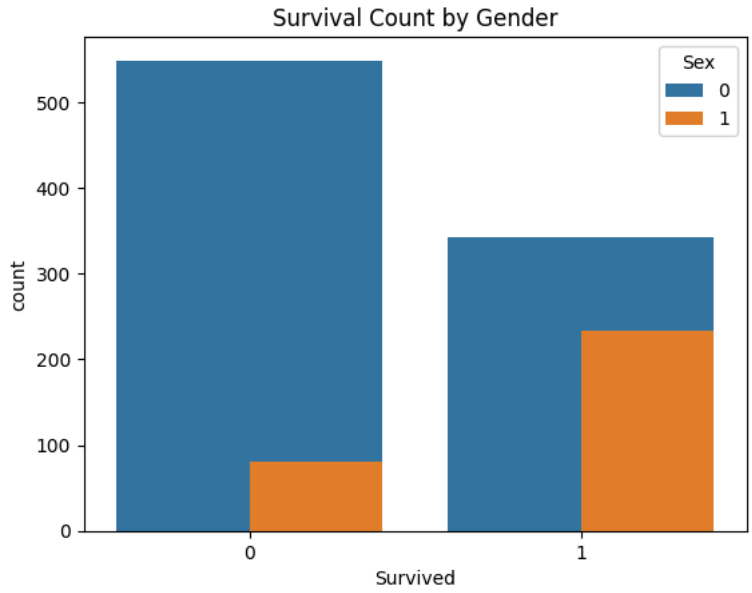
```
cm=confusion_matrix(y_test, y_pred)
sns.heatmap(cm, annot=True, fmt='d',cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted')

plt.ylabel('Actual')
plt.show()
```

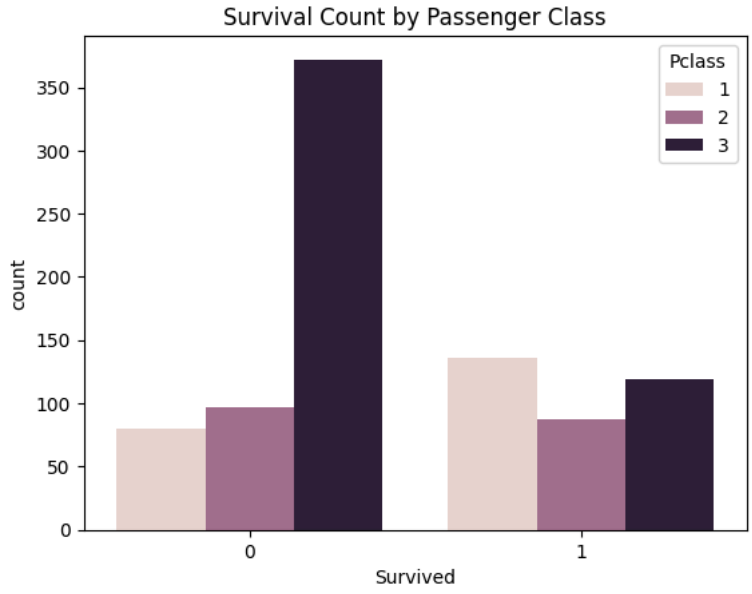


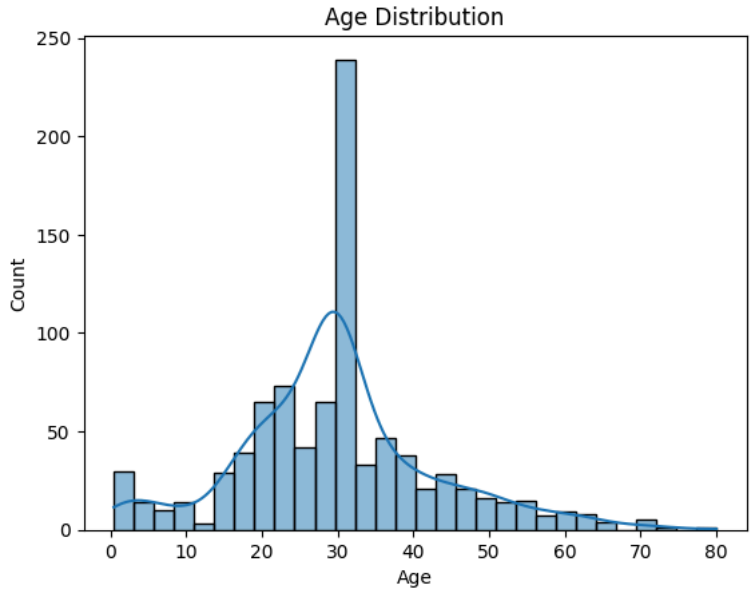

```
print("classification report:\n",classification_report(y_test, y_pred))
```

```
classification report:
              precision    recall  f1-score   support

           0       0.81      0.87      0.84       105
           1       0.79      0.72      0.75        74

    accuracy                           0.80       179
   macro avg       0.80      0.79      0.80       179
weighted avg       0.80      0.80      0.80       179
```

```
sns.countplot(x='Survived',data=data)
plt.title("Survival Count(0=Died,1=Survived)")
sns.countplot(x='Survived',hue='Sex',data=data)
plt.title("Survival Count by Gender")
plt.show()
sns.countplot(x='Survived',hue='Pclass',data=data)
plt.title("Survival Count by Passenger Class")
plt.show()
sns.histplot(data['Age'],bins=30,kde=True)
plt.title("Age Distribution")
plt.show()
```








```
from sklearn.ensemble import RandomForestClassifier
```

```
rf_model = RandomForestClassifier(n_estimators=100, random_state=42)
```

```
rf_model.fit(x_train, y_train)
rf_pred = rf_model.predict(x_test)
```

```
print(classification_report(y_test, rf_pred))
print(confusion_matrix(y_test, rf_pred))
print("Accuracy:", accuracy_score(y_test, rf_pred))
```

```
              precision    recall  f1-score   support

           0       0.82      0.87      0.84       105
           1       0.79      0.73      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

[[91 14]
 [20 54]]
Accuracy: 0.8100558659217877
```

```
import joblib
```

```
joblib.dump(rf_model,"titanic_model.pkl")
```

```
['titanic_model.pkl']
```

Start coding or generate with AI.