In [1]:
```python
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
Task 1: Data Loading and Initial Assessment
1.1 Import the Titanic dataset from the CSV file.
```

In [2]:
```python
data=pd.read_csv("titanic.csv")
```

In [3]:
```python
data.head()
```

Out[3]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	

```
1.2 Perform initial data checks to identify the number of rows and columns
in the dataset.
```

In [4]:
```python
data.shape
```

Out[4]: (891, 12)

In [5]:
```
data.columns
```

Out[5]:
```
Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
       'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
      dtype='object')
```

In [6]:
```
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
```

1.3 Identify and display the count of null values in the 'Age' and 'Cabin' columns.

In [7]:
```
data.Age.isnull().sum()
```

Out[7]: 177

In [8]:
```
data.Cabin.isnull().sum()
```

Out[8]: 687

Task 2: Null Value Imputation
2.1 Fill the missing values in the 'Age' column using the mean value.

In [9]:
```
data.Age.fillna(data.Age.mean(),inplace=True)
```

In [10]:
```
data.Age.isnull().sum()
```

Out[10]: 0

2.2 Fill the missing values in the 'Fare' column using the median value.

In [11]:
```
data.Fare.isnull().sum()
```

Out[11]:
```
0
```

2.3 Fill the missing values in the 'Embarked' column with the most common value ('S').

In [12]:
```
data.Embarked.fillna('S',inplace=True)
```

In [13]:
```
data.Embarked.isnull().sum()
```

Out[13]:
```
0
```

Task 3: Feature Engineering
3.1 Convert the 'Age' column to an integer type.

In [14]:
```
data.Age.dtype
```

Out[14]:
```
dtype('float64')
```

In [15]:
```
data['Age']=data.Age.astype(int)
```

In [16]:
```
data.Age.dtype
```

Out[16]:
```
dtype('int32')
```

3.2 Create a new binary feature 'Cabin_Exist' indicating the presence or absence of cabin information.

In [17]:
```
data['Cabin_Exist']=~data.Cabin.isnull()
```

In [18]:
```
data.head()
```

Out[18]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	

3.3 Group the 'Age' and 'Fare' columns into quartiles, creating new features 'Age_Group' and 'Fare_range'.

In [19]:
```
data['Age_Group']=pd.qcut(data.Age,q=4,labels=False)
```

In [20]:
```
data.head()
```

Out[20]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	

In [21]:
```
data['Fare_Range']=pd.qcut(data.Fare,q=4,labels=False)
```

In [22]:
```
data.head()
```

Out[22]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	

3.4 Create a 'Family' feature by combining 'Parch' and 'SibSp'.

In [23]:
```
data['Family']=data['SibSp']+data['Parch']
```

In [24]:
```
data.head()
```

Out[24]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	C
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35	1	0	113803	53.1000	C
4	5	0	3	Allen, Mr. William Henry	male	35	0	0	373450	8.0500	

```
3.5 Perform feature selection by dropping irrelevant columns.
```

In [25]:
```
data.drop(['PassengerId','Name','Age','SibSp','Parch','Ticket','Fare','Cabin'
```

In [26]:
```
data.head()
```

Out[26]:

	Survived	Pclass	Sex	Embarked	Cabin_Exist	Age_Group	Fare_Range	Family
0	0	3	male	S	False	0	0	1
1	1	1	female	C	True	3	3	1
2	1	3	female	S	False	1	1	0
3	1	1	female	S	True	2	3	1
4	0	3	male	S	False	2	1	0

```
Task 4: Data Encoding
4.1 Encode categorical data into binary form using one-hot encoding.
```

In [27]:
```
data=pd.get_dummies(data=data,columns=['Sex','Embarked'],drop_first=True)
```

In [28]:
```
data.head()
```

Out[28]:

	Survived	Pclass	Cabin_Exist	Age_Group	Fare_Range	Family	Sex_male	Embarked_Q	Em
0	0	3	False	0	0	1	1	0	
1	1	1	True	3	3	1	0	0	
2	1	3	False	1	1	0	0	0	
3	1	1	True	2	3	1	0	0	
4	0	3	False	2	1	0	1	0	

In [29]:
```
y=data['Survived']
data.drop('Survived',axis=1,inplace=True)
X=data
X.head()
```

Out[29]:

	Pclass	Cabin_Exist	Age_Group	Fare_Range	Family	Sex_male	Embarked_Q	Embarked_S
0	3	False	0	0	1	1	0	1
1	1	True	3	3	1	0	0	0
2	3	False	1	1	0	0	0	1
3	1	True	2	3	1	0	0	1
4	3	False	2	1	0	1	0	1

In [30]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test=train_test_split(X,y,test_size=0.3,random_sta
```

In [31]:
```
X_train.shape
```

Out[31]: (623, 8)

In [32]:
```
X_test.shape
```

Out[32]: (268, 8)

In [33]:
```
y_train.shape
```

Out[33]: (623,)

In [34]:
```
y_test.shape
```

Out[34]: (268,)

```
Task 5: Data Scaling
```

5.1 Apply MinMaxScaler to scale the dataset and normalize features for model training.

In [35]:
```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

In [36]:
```
X_train=scaler.fit_transform(X_train)
X_train
```

Out[36]:
```
array([[0.        , 0.        , 0.66666667, ..., 0.        , 0.        ,
        0.        ],
       [1.        , 0.        , 0.33333333, ..., 1.        , 0.        ,
        1.        ],
       [1.        , 0.        , 0.        , ..., 0.        , 0.        ,
        1.        ],
       ...,
       [0.5       , 0.        , 0.66666667, ..., 1.        , 0.        ,
        1.        ],
       [0.        , 1.        , 1.        , ..., 1.        , 0.        ,
        0.        ],
       [0.5       , 0.        , 0.66666667, ..., 1.        , 0.        ,
        0.        ]])
```

In [37]:
```
X_test=scaler.fit_transform(X_test)
```

In [38]:
```
X_test
```

Out[38]:
```
array([[1.        , 0.        , 0.        , ..., 1.        , 0.        ,
        1.        ],
       [1.        , 0.        , 0.        , ..., 1.        , 0.        ,
        1.        ],
       [1.        , 0.        , 0.33333333, ..., 0.        , 1.        ,
        0.        ],
       ...,
       [0.5       , 1.        , 0.33333333, ..., 0.        , 0.        ,
        0.        ],
       [0.5       , 0.        , 0.33333333, ..., 1.        , 0.        ,
        1.        ],
       [0.5       , 0.        , 0.        , ..., 1.        , 0.        ,
        1.        ]])
```

Task 6: Model Training and Evaluation
6.1 Train a Logistic Regression model on the preprocessed data.

In [39]:
```
from sklearn.linear_model import LogisticRegression
```

In [40]:
```
model=LogisticRegression()
```

In [41]:
```
model.fit(X_train,y_train)
```

Out[41]:

```
▾ LogisticRegression ⓘ ?
(https://scikit-
learn.org/1.4/modules/generated/sklearn.linear_model.LogisticR
LogisticRegression()
```

In [42]:
```
predict=model.predict(X_test)
predict
```

Out[42]:
```
array([0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0,
       0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0,
       0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0,
       1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1,
       0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0,
       0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0,
       1, 1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1,
       0, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0,
       1, 1, 0, 0], dtype=int64)
```

6.2 Evaluate the model's accuracy.

In [43]:
```
from sklearn.metrics import accuracy_score
```

In [44]:
```
accuracy_score(y_test,predict)
```

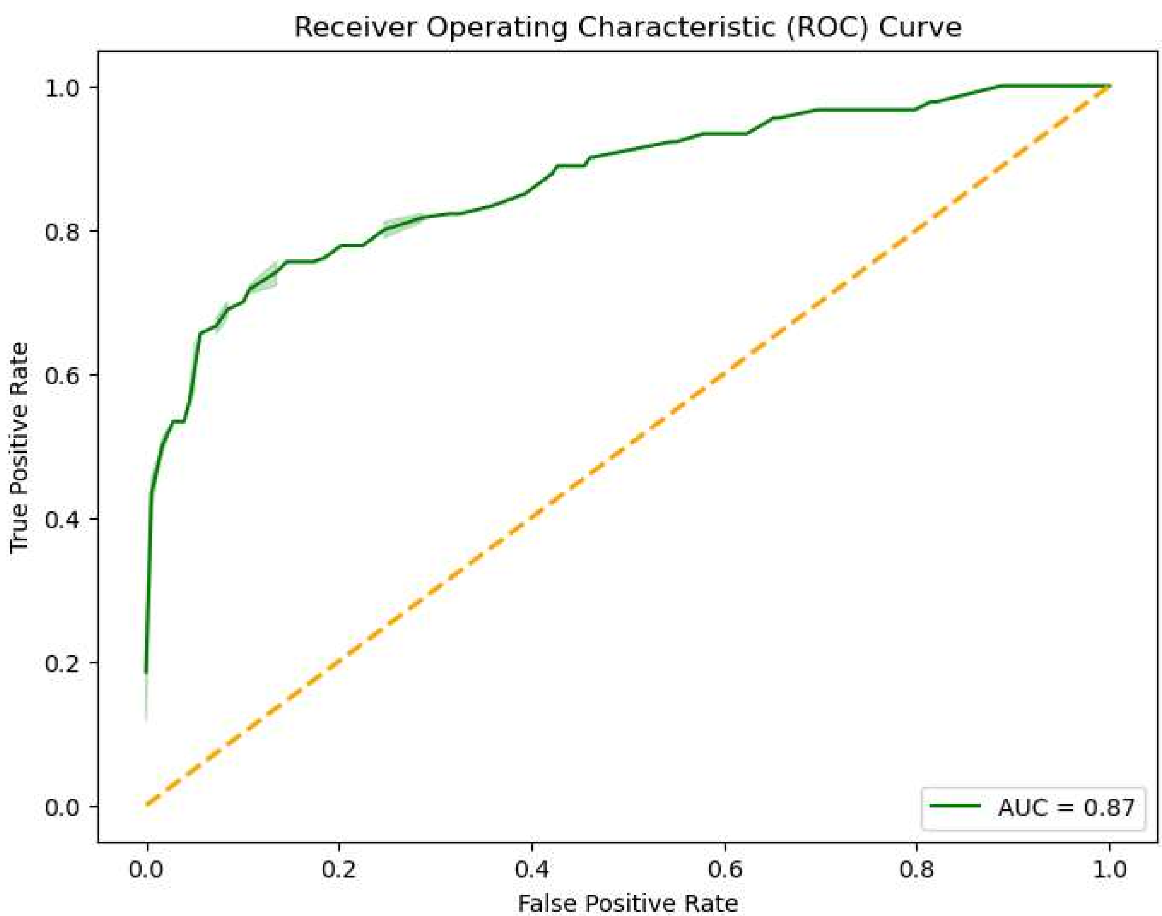
Out[44]: 0.8208955223880597

6.3 Calculate the AUC score of the model.

In [45]:
```
from sklearn.metrics import roc_auc_score
y_score=model.predict_proba(X_test)[:,1]
roc_auc_score(y_test,y_score)
```

Out[45]: 0.8684144818976279

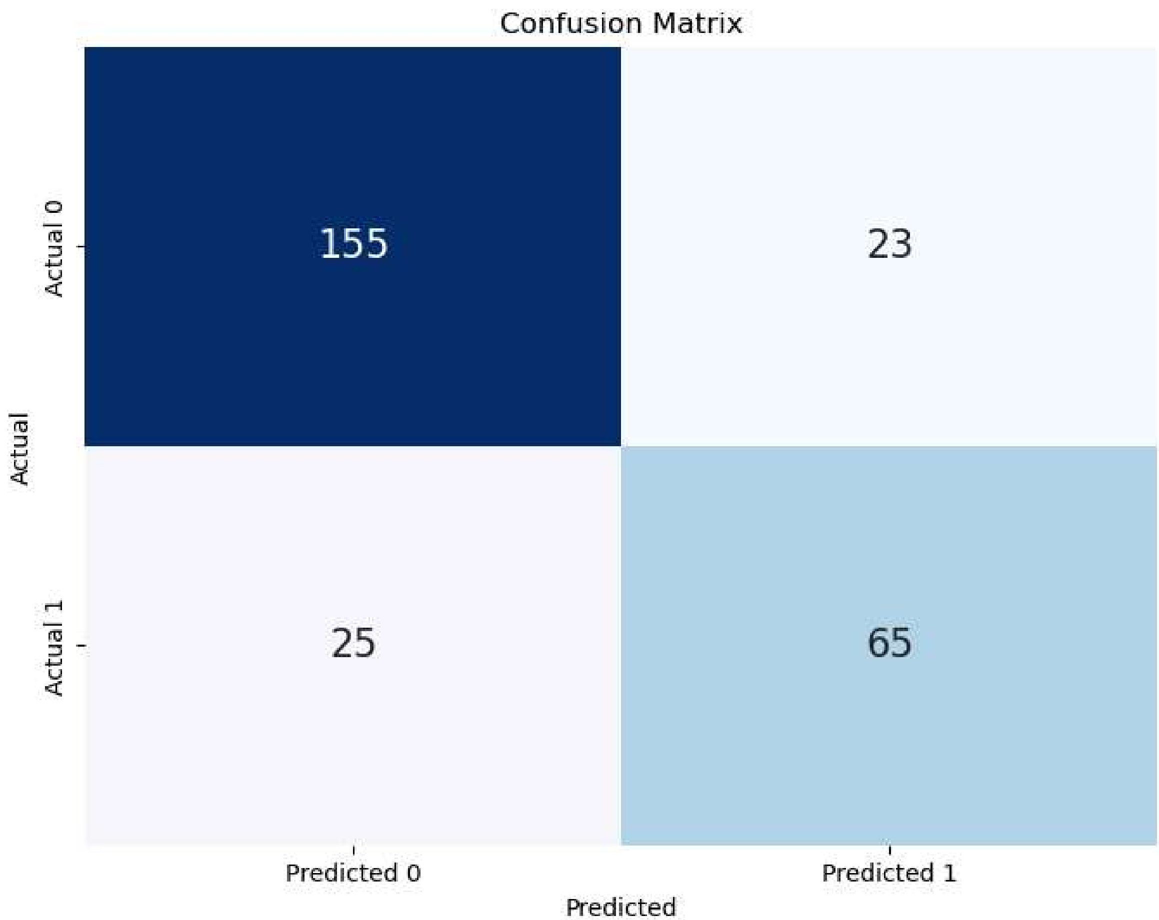
In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
fpr, tpr, thresholds = roc_curve(y_test, y_score)
roc_auc = roc_auc_score(y_test, y_score)
# Create a DataFrame with the ROC curve data
roc_df = pd.DataFrame({'False Positive Rate': fpr, 'True Positive Rate': tpr,
plt.figure(figsize=(8, 6))
sns.lineplot(data=roc_df, x='False Positive Rate', y='True Positive Rate', lal
plt.plot([0, 1], [0, 1], color='orange', lw=2, linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend(loc='lower right')
plt.show()
```



In [47]:
```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,predict)
```

Out[47]:
```
array([[155,  23],
       [ 25,  65]], dtype=int64)
```

In [48]:
```
from sklearn.metrics import confusion_matrix
import seaborn as sns
conf_matrix = confusion_matrix(y_test, predict)
conf_df = pd.DataFrame(conf_matrix, index=['Actual 0', 'Actual 1'], columns=[
plt.figure(figsize=(8, 6))
sns.heatmap(conf_df, annot=True, fmt='d', cmap='Blues', cbar=False, annot_kws=
plt.title('Confusion Matrix')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.show()
```



In [49]:
```
from sklearn.metrics import precision_score
precision_score(y_test,predict)
```

Out[49]: 0.7386363636363636

In [50]:
```
from sklearn.metrics import recall_score
recall_score(y_test,predict)
```

Out[50]: 0.7222222222222222

In [51]:
```python
from sklearn.metrics import f1_score
f1_score(y_test,predict)
```

Out[51]: 0.7303370786516854

```
Task 7: Conclusion
7.1 Summarize the key findings from the exploration, feature engineering,
and model training process.
Data Loading and Initial Assessment:The dataset was loaded successfully,
and initial checks revealed that there were 891 rows and 12 columns. Some
columns had missing values, with 'Cabin' having a significant number of
null values (687 out of 891).
Null Value Imputation:Missing values in the 'Age' column were filled with
the mean value, those in the 'Fare' column were filled with the median
value, and missing values in the 'Embarked' column were filled with the
most common value ('S').
Feature Engineering: The 'Age' column was converted to an integer type, a
new binary feature 'Cabin_Exist' indicating the presence or absence of
cabin information was created, and both 'Age' and 'Fare' columns were
grouped into quartiles to create 'Age_Group' and 'Fare_Range' features.
Additionally, a 'Family' feature was created by combining 'Parch' and
'SibSp'.
Feature Selection: Irrelevant columns were dropped, including
'PassengerId', 'Name', 'Age', 'SibSp', 'Parch', 'Ticket', 'Fare', and
'Cabin'.
Data Encoding: Categorical data was encoded into binary form using one-hot
encoding for the 'Sex' and 'Embarked' columns.
Model Training and Evaluation: A Logistic Regression model was trained on
the preprocessed data. The model achieved an accuracy of approximately 82%,
an AUC score of 0.87, precision of 0.74, recall of 0.72, and an F1 score of
0.73.


7.2 Insights and Potential Improvements:
Insights:
  - The Logistic Regression model shows good performance in terms of
accuracy, AUC score, and other evaluation metrics.
  - The engineered features, such as 'Cabin_Exist', 'Age_Group',
'Fare_Range', and 'Family', contribute to the model's predictive power.
Potential Improvements:
  - Feature Engineering:Further exploration and creation of new features
could enhance the model's performance.
  - Hyperparameter Tuning:Fine-tuning the hyperparameters of the Logistic
Regression model or trying different models might improve predictive
accuracy.
  - Handling Imbalanced Data: If the dataset is imbalanced, applying
techniques such as oversampling or undersampling could be considered to
improve model performance.
  - Ensemble Models: Trying ensemble models like Random Forest or Gradient
Boosting might capture more complex patterns in the data.
```