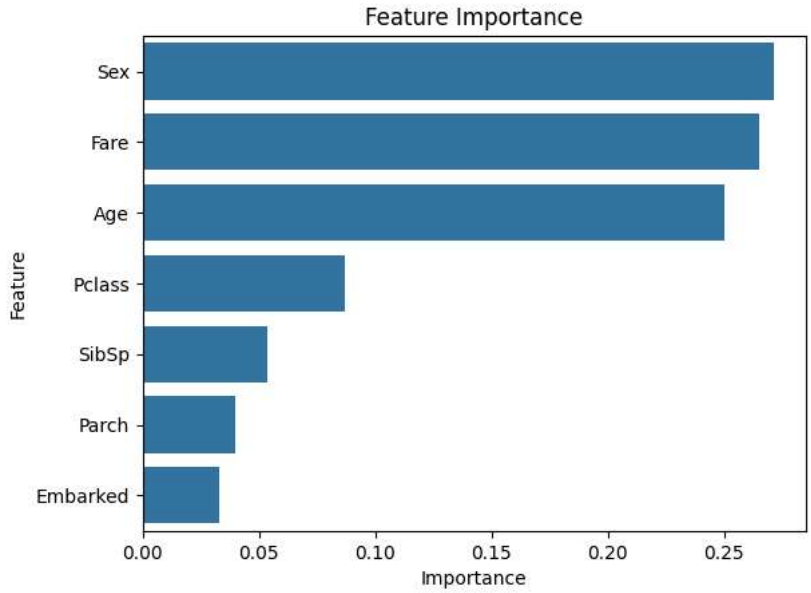
Task 1

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

# Load the dataset
file_path = 'Titanic-Dataset.csv'
data = pd.read_csv(file_path)

# Exploratory Data Analysis (EDA)
print("\nDataset Overview:\n", data.head())
print("\nData Info:\n")
data.info()
print("\nMissing Values:\n", data.isnull().sum())

# Data Cleaning
# Fill missing values
# Assuming 'Age' and 'Embarked' have missing values
if 'Age' in data.columns:
    data['Age'].fillna(data['Age'].median(), inplace=True)
if 'Embarked' in data.columns:
    data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Drop columns with too many missing values (e.g., 'Cabin')
if 'Cabin' in data.columns:
    data.drop('Cabin', axis=1, inplace=True)

# Drop irrelevant columns (e.g., 'PassengerId', 'Name', 'Ticket')
irrelevant_cols = [col for col in ['PassengerId', 'Name', 'Ticket'] if col in data.columns]
data.drop(columns=irrelevant_cols, inplace=True)

# Feature Engineering
# Encode categorical variables
categorical_cols = data.select_dtypes(include=['object']).columns
label_encoders = {}
for col in categorical_cols:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Define features (X) and target (y)
X = data.drop('Survived', axis=1) if 'Survived' in data.columns else data
y = data['Survived'] if 'Survived' in data.columns else None

# Split the dataset 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)

# Model Building
model = RandomForestClassifier(random_state=42)
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluation
print("\nAccuracy Score:", accuracy_score(y_test, y_pred))
print("\nConfusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))

# Feature Importance
importance = model.feature_importances_
feature_importance_df = pd.DataFrame({'Feature': X.columns, 'Importance': importance}).sort_values(by='Importance', ascending=False)
print("\nFeature Importances:\n", feature_importance_df)

# Visualization of Feature Importance
sns.barplot(x='Importance', y='Feature', data=feature_importance_df)
plt.title('Feature Importance')
plt.show()
```

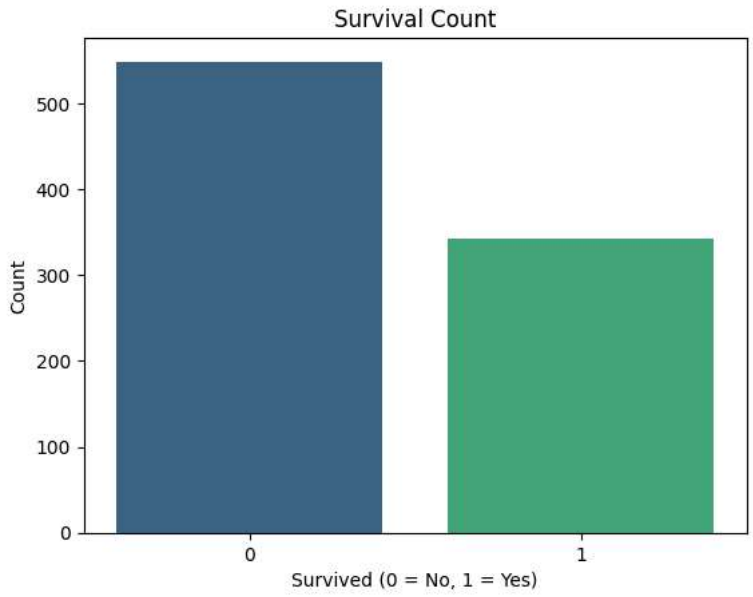
```
# Visualization of Survival Count
if 'Survived' in data.columns:
    sns.countplot(data=data, x='Survived', palette='viridis')
    plt.title('Survival Count')
    plt.xlabel('Survived (0 = No, 1 = Yes)')
    plt.ylabel('Count')
    plt.show()

# Visualization of Age Distribution
if 'Age' in data.columns:
    sns.histplot(data['Age'], kde=True, bins=30, color='blue')
    plt.title('Age Distribution')
    plt.xlabel('Age')
    plt.ylabel('Frequency')
    plt.show()

# Visualization of Fare Distribution
if 'Fare' in data.columns:
    sns.histplot(data['Fare'], kde=True, bins=30, color='green')
    plt.title('Fare Distribution')
    plt.xlabel('Fare')
    plt.ylabel('Frequency')
    plt.show()

# Correlation Heatmap
plt.figure(figsize=(10, 8))
correlation_matrix = data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Correlation Heatmap')
plt.show()
```

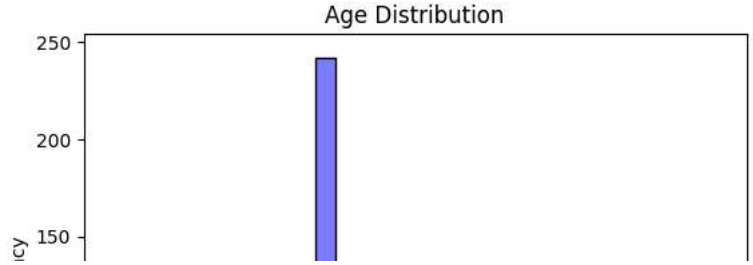
```
Dataset Overview:
    PassengerId  Survived  Pclass  \
0            1         0       3
1            2         1       1
2            3         1       3
3            4         1       1
4            5         0       3

                                                 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

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

Missing Values:
 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
<ipython-input-2-0a4973ed09d2>:24: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

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


  data['Age'].fillna(data['Age'].median(), inplace=True)
<ipython-input-2-0a4973ed09d2>:26: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignm
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

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


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

Accuracy Score: 0.8212290502793296

Confusion Matrix:
 [[92 13]
 [19 55]]

Classification Report:
              precision    recall  f1-score   support

           0       0.83      0.88      0.85       105
```

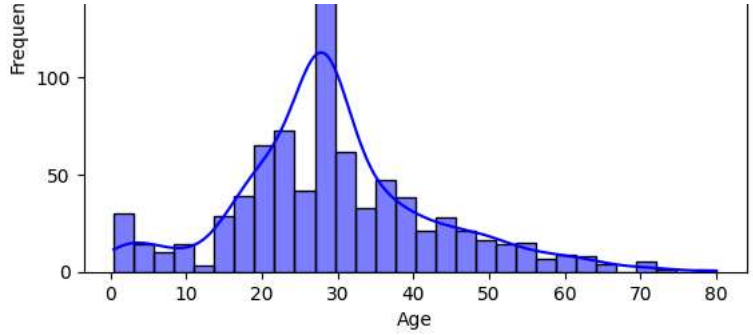
```
           1       0.81      0.74      0.77        74

    accuracy                           0.82       179
   macro avg       0.82      0.81      0.81       179
weighted avg       0.82      0.82      0.82       179


Feature Importances:
     Feature  Importance
1        Sex    0.271410
5       Fare    0.265010
2        Age    0.249995
0     Pclass    0.086957
3      SibSp    0.053685
4      Parch    0.039897
6   Embarked    0.033044
```

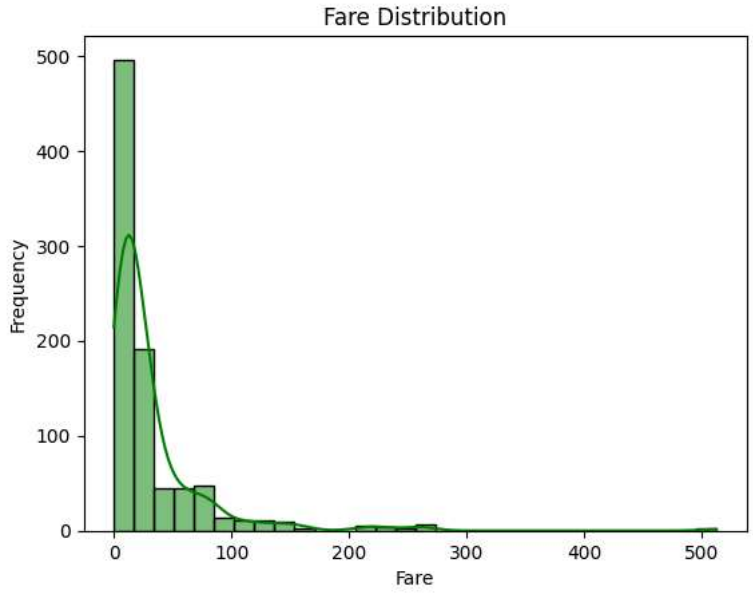


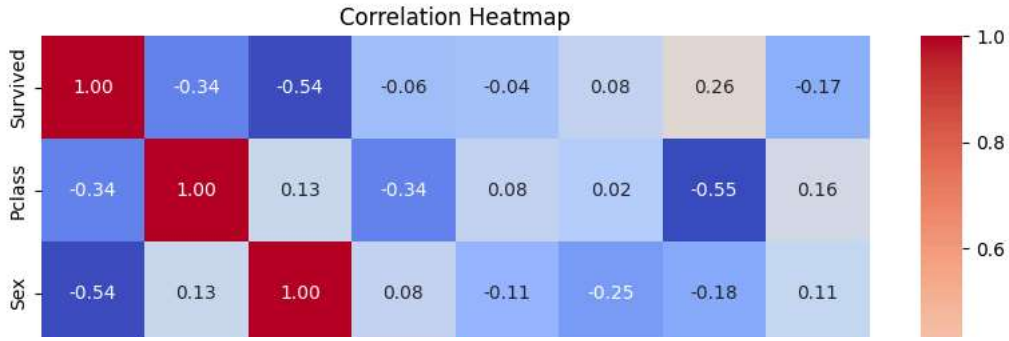

```
<ipython-input-2-0a4973ed09d2>:76: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `legenc

  sns.countplot(data=data, x='Survived', palette='viridis')
```






Frequen
100
50
0
0
10
20
30
40
50
60
70
80
Age


Fare Distribution
500
400
300
200
100
0
Frequency
0
100
200
300
400
500
Fare


Correlation Heatmap
Survived
Pclass
Sex
1.00
-0.34
-0.54
-0.06
-0.04
0.08
0.26
-0.17
-0.34
1.00
0.13
-0.34
0.08
0.02
-0.55
0.16
-0.54
0.13
1.00
0.08
-0.11
-0.25
-0.18
0.11
1.0
0.8
0.6