Titanic Classification

This below code provides a basic framework for building and evaluating a machine learning model to predict survival on the Titanic.

1. Importing Libraries: The first lines of code import the necessary libraries for data manipulation, machine learning modeling, and evaluation.

In [1]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
```

2.Loading the Dataset: The Titanic dataset is loaded using Pandas.

In [2]:
```
df = pd.read_csv("Titanic-Dataset.csv")
```

3.Data Preprocessing:

Removing Unnecessary Columns: Columns like PassengerId, Name, Ticket, and Cabin are not likely to have significant predictive power, so they are dropped from the dataset. Mapping Categorical Variables: The 'Sex' and 'Embarked' columns are mapped from categorical values (like 'male', 'female') to numerical values (0, 1, 2). Handling Missing Values: Missing values in the 'Age' and 'Embarked' columns are filled with the median value of the respective columns. Missing 'Fare' values are filled with the median fare.

In [3]:
```
df.drop(['PassengerId', 'Name', 'Ticket', 'Cabin'], axis=1, inplace=True)
df['Sex'] = df['Sex'].map({'male': 0, 'female': 1})
df['Embarked'] = df['Embarked'].map({'S': 0, 'C': 1, 'Q': 2})
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df['Fare'].fillna(df['Fare'].median(), inplace=True)
```

4.Feature Selection: The features (X) and the target variable (y) are separated.

In [4]:
```
X = df.drop('Survived', axis=1)
y = df['Survived']
```

5.Splitting the Data: The dataset is split into training and testing sets to evaluate the model's performance.

In [5]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6.Model Training: A Random Forest classifier is instantiated and trained on the training data.

In [6]:
```
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)
```

Out[6]:
```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

7.Making Predictions: The trained model is used to make predictions on the testing data.

In [8]:
```
y_pred = clf.predict(X_test)
```

8.Model Evaluation: The performance of the model is evaluated using accuracy and a classification report.

In [9]:
```
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
print("Classification Report:")
print(classification_report(y_test, y_pred))
```

```
Accuracy: 0.8268156424581006
Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.88      0.86       105
           1       0.81      0.76      0.78        74

    accuracy                           0.83       179
   macro avg       0.82      0.82      0.82       179
weighted avg       0.83      0.83      0.83       179
```

In [ ]: