In [3]:
```
import pandas as pd
import numpy as np
```

In [5]:
```
data=pd.read_csv("C:\\Users\\vtu10\\Downloads\\Titanic-Dataset.csv")
```

In [6]:
```
data
```

Out[6]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fa
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.25
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.28
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.92
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.10
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05
...	...	...	...	...	...	...	...	...	...	
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75

891 rows × 12 columns

In [7]:
```
data.head()
```

Out[7]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
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
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500

```
Data Preparation:

Load the dataset.
Handle missing values.
Encode categorical variables.
Feature selection.
Split the data into training and testing sets.
```

In [8]:
```
# Fill missing values for 'Age' with the median value
data['Age'].fillna(data['Age'].median(), inplace=True)

# Fill missing values for 'Embarked' with the most common value
data['Embarked'].fillna(data['Embarked'].mode()[0], inplace=True)

# Drop the 'Cabin' column as it has too many missing values
data.drop(columns=['Cabin'], inplace=True)
```

```
C:\Users\vtu10\AppData\Local\Temp/ipykernel_20476/1041460282.py:2: Future
Warning: A value is trying to be set on a copy of a DataFrame or Series t
hrough chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wo
rk 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(valu
e) instead, to perform the operation inplace on the original object.


  data['Age'].fillna(data['Age'].median(), inplace=True)
C:\Users\vtu10\AppData\Local\Temp/ipykernel_20476/1041460282.py:5: Future
Warning: A value is trying to be set on a copy of a DataFrame or Series t
hrough chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never wo
rk 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(valu
e) instead, to perform the operation inplace on the original object.


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

In [9]:
```
# Convert 'Sex' into numerical values
data['Sex'] = data['Sex'].map({'male': 0, 'female': 1})

# One-hot encode 'Embarked'
data = pd.get_dummies(data, columns=['Embarked'], drop_first=True)
```

In [10]:
```
# Select relevant features
features = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked_Q'
X = data[features]
y = data['Survived']
```

In [11]:
```
from sklearn.model_selection import train_test_split

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, ra
```

In [12]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusi

# Initialize the model
model = RandomForestClassifier(n_estimators=100, random_state=42)

# Train the model
model.fit(X_train, y_train)

# Make predictions
y_pred = model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
report = classification_report(y_test, y_pred)
confusion = confusion_matrix(y_test, y_pred)

accuracy, report, confusion
```

Out[12]:
```
(0.7988826815642458,
 '              precision    recall  f1-score   support\n\n           0
0.82      0.84      0.83       105\n           1       0.76      0.74
0.75        74\n\n    accuracy                           0.80       179\n
macro avg       0.79      0.79      0.79       179\nweighted avg       0.
80      0.80      0.80       179\n',
 array([[88, 17],
        [19, 55]], dtype=int64))
```

In [ ]:
```

```