**Task 2**

**Explanation:**

This code builds a machine learning pipeline to classify whether passengers were transported or not (likely for a Titanic or space-related dataset). It starts by loading training (train.csv) and test (test.csv) datasets using pandas. The data is cleaned by forward-filling missing values (ffill()) and converting categorical features like Cabin into meaningful subcategories (CabinDeck). The categorical variables HomePlanet, Destination, and CabinDeck are encoded using LabelEncoder, transforming them into numerical values. Unnecessary columns (Name, Cabin, PassengerId) are removed to prevent irrelevant data from affecting the model.Next, the dataset is split into features(X) and the target variable (y: Transported). The data is further divided into training and validation sets using train\_test\_split(). Since numerical values of different scales can impact model performance, StandardScaler is used to normalize the data. A Random Forest Classifier is then trained on the processed training set and evaluated on the validation set using accuracy score and classification report, which provide detailed performance metrics.Finally, predictions are made on the test set, and the results are saved in a submission.csv file with the PassengerId and predicted transport status. This structured approach ensures effective data preprocessing, feature engineering, and model training, making it suitable for handling categorical and numerical data efficiently.

**Dataset:**

<https://www.kaggle.com/competitions/spaceship-titanic/data>

**Code and Output:**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.ensemble import RandomForestClassifier

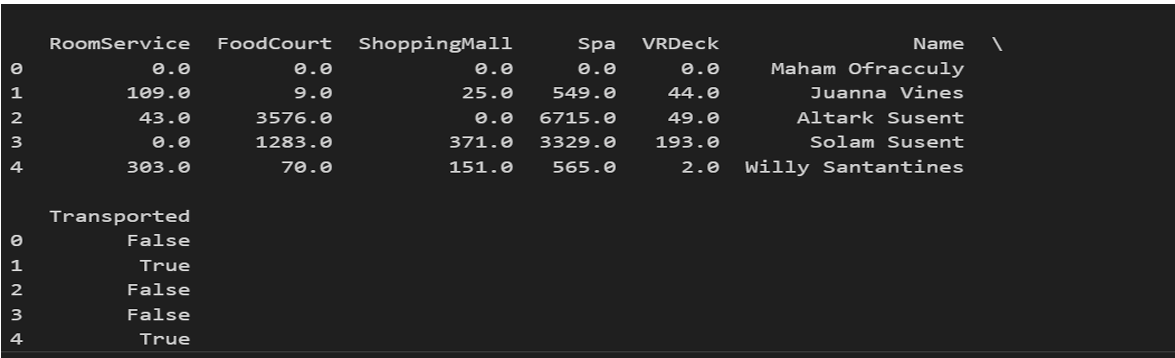
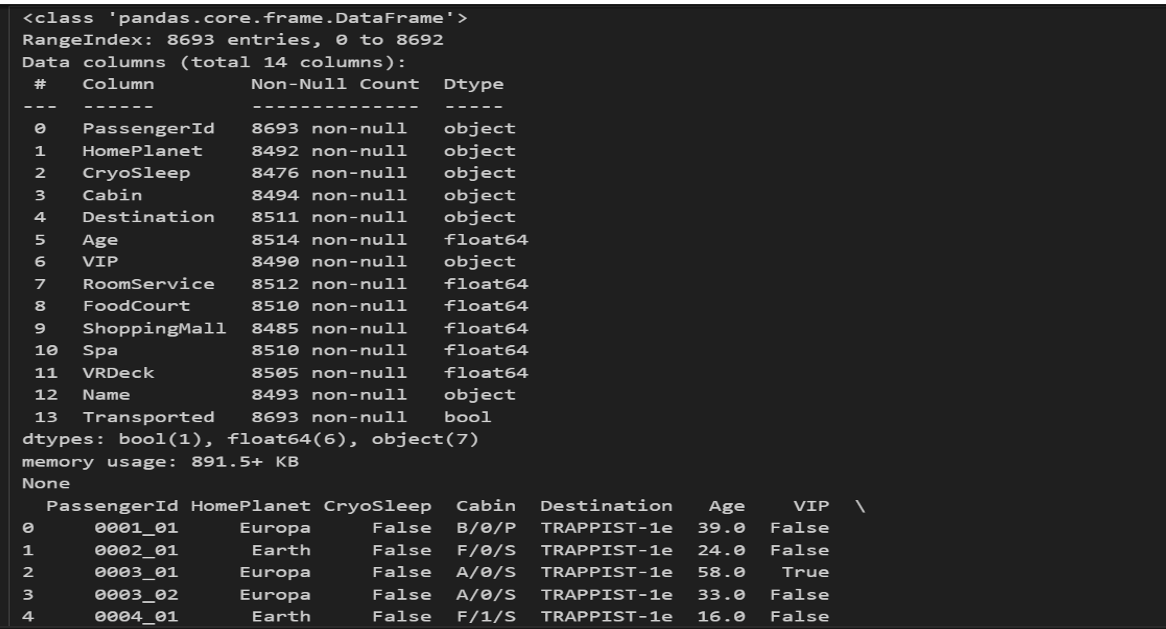
from sklearn.metrics import accuracy\_score, classification\_report

train\_df =pd.read\_csv("train.csv")

test\_df =pd.read\_csv("test.csv")

print(train\_df.info())

print(train\_df.head())



train\_df= train\_df.ffill().infer\_objects(copy=False)

test\_df= test\_df.ffill().infer\_objects(copy=False)

if 'Cabin' in train\_df.columns:

train\_df['Cabin'] =train\_df['Cabin'].astype(str)

if 'Cabin' in test\_df.columns:

test\_df['Cabin'] = test\_df['Cabin'].astype(str)

if isinstance(test\_df, pd.DataFrame) and 'Cabin' in test\_df.columns:

test\_df['CabinDeck'] =test\_df['Cabin'].astype(str).str.split('/').str[0]

if isinstance(train\_df, pd.DataFrame) and 'Cabin' in train\_df.columns:

train\_df['CabinDeck'] =train\_df['Cabin'].astype(str).str.split('/').str[0]

categorical\_cols=['HomePlanet','Destination','CabinDeck']

encoder =LabelEncoder()

for col in categorical\_cols:

if col in train\_df.columns:

train\_df[col] =encoder.fit\_transform(train\_df[col].astype(str))

if col in test\_df.columns:

test\_df[col] =encoder.transform(test\_df[col].astype(str))

columns\_to\_drop=['Name','Cabin','PassengerId']

train\_df.drop([col for col in columns\_to\_drop if col in train\_df.columns],axis=1,inplace=True)

if 'PassengerId' in test\_df.columns:

test\_passenger\_ids = test\_df['PassengerId']

test\_df.drop([col for col in columns\_to\_drop if col in test\_df.columns], axis=1,inplace=True)

X = train\_df.drop(columns=['Transported'])

y =train\_df['Transported'].astype(int)

X\_train, X\_val, y\_train, y\_val =train\_test\_split(X, y, test\_size=0.2,random\_state=42)

scaler =StandardScaler()

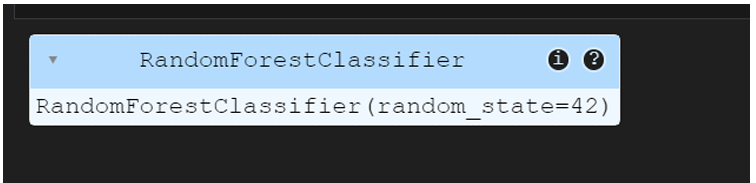
X\_train =scaler.fit\_transform(X\_train)

X\_val =scaler.transform(X\_val)

test\_df =scaler.transform(test\_df)

model=RandomForestClassifier(n\_estimators=100,random\_state=42)

model.fit(X\_train, y\_train)



y\_pred=model.predict(X\_val)

print("aaccuracy:",accuracy\_score(y\_val,y\_pred))

print(classification\_report(y\_val,y\_pred))

test\_predictions=model.predict(test\_df)

submission =pd.DataFrame({'PassengerId':test\_passenger\_ids,'Passengers Transported':test\_predictions})

submission.to\_csv("submission.csv",index=False)

print("submission file saved")

