import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

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

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, roc\_auc\_score

file\_path = r'C:\Users\santh\dataset\Data\_Train.xlsx'

# Read the Excel file into a DataFrame

data = pd.read\_excel(file\_path)

# Handling Null Values

df.dropna(inplace=True)

#converting the Date\_of\_Journey to 'datetime'

df['Date\_of\_Journey'] = pd.to\_datetime(df.Date\_of\_Journey, format='%d/%m/%Y')

#Extracting day & creating new columns Journey\_day

df['Journey\_day'] = df['Date\_of\_Journey'].dt.day

#extracting month & creating new columns Journey\_day

df['Journey\_month'] = df['Date\_of\_Journey'].dt.month

df.drop(['Date\_of\_Journey'], axis=1, inplace=True)

df['Arrival\_Time'] = pd.to\_datetime(df.Arrival\_Time)

df['Arrival\_hour'] = df['Arrival\_Time'].dt.hour

df['Arrival\_minute'] = df['Arrival\_Time'].dt.minute

df.drop(['Arrival\_Time'], axis=1, inplace=True)

#error may occur

df['Dep\_Time'] = pd.to\_datetime(df.Dep\_Time)

df['Dep\_hour'] = df['Dep\_Time'].dt.hour

df['Dep\_minute'] = df['Dep\_Time'].dt.minute

df.drop(['Dep\_Time'], axis=1, inplace=True)

# 'Duration'--> Time taken by plane to reach destination.

# It is the differnce between Departure time and Arrival time

# Assigning and converting Duration column into list to extract values one-by-one in for-loop

duration = list(df["Duration"])

for i in range(len(duration)):

if len(duration[i].split()) != 2: # Check if duration contains only hour or mins

if "h" in duration[i]:

duration[i] = duration[i].strip() + " 0m" # Adds 0 minute #why strip is used here?

else:

duration[i] = "0h " + duration[i] # Adds 0 hour

duration\_hours = []

duration\_mins = []

for i in range(len(duration)):

duration\_hours.append(int(duration[i].split(sep = "h")[0])) # Extract hours from duration

duration\_mins.append(int(duration[i].split(sep = "m")[0].split()[-1])) # Extracts only minutes from duration

df['Duration\_hours'] = duration\_hours

df['Duration\_minutes'] = duration\_mins

df.drop(['Duration'], axis=1, inplace=True)

#Handling Categorical Data

#Here Handling Categorical data by using OneHotEncodingb

# Graph of Airline vs Price

sns.catplot(y='Price', x='Airline', data=df.sort\_values('Price', ascending=False), kind='boxen', height=6, aspect=3)

# Assume 'model', 'source\_encoder', and 'destination\_encoder' are globally initialized

from sklearn.ensemble import RandomForestClassifier

from sklearn.preprocessing import LabelEncoder

import pandas as pd

# Placeholder: Sample training data for demonstration

training\_data = pd.DataFrame({

'Source': ['Delhi', 'Mumbai', 'Chennai', 'Kolkata'],

'Destination': ['Chennai', 'Delhi', 'Kolkata', 'Mumbai'],

'Journey\_day': [10, 20, 5, 12],

'Journey\_month': [10, 5, 8, 3],

'Dep\_hour': [12, 8, 18, 15],

'Duration\_hours': [3, 4, 2, 5],

'Duration\_minutes': [30, 45, 20, 40],

'Price\_change': [1, 0, 1, 0] # 1 = price increase, 0 = price decrease

})

# Encode the categorical variables

source\_encoder = LabelEncoder()

destination\_encoder = LabelEncoder()

training\_data['Source'] = source\_encoder.fit\_transform(training\_data['Source'])

training\_data['Destination'] = destination\_encoder.fit\_transform(training\_data['Destination'])

# Features and target

X\_train = training\_data[['Source', 'Destination', 'Journey\_day', 'Journey\_month', 'Dep\_hour', 'Duration\_hours', 'Duration\_minutes']]

y\_train = training\_data['Price\_change']

# Train the RandomForestClassifier

model = RandomForestClassifier()

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

# Function to get user input and predict

def get\_input\_and\_predict():

# Get user input for source, destination, and time (date)

source = input("Enter source (city): ")

destination = input("Enter destination (city): ")

journey\_date = input("Enter date of journey (DD/MM/YYYY): ")

# Convert the input date to datetime

try:

journey\_date = pd.to\_datetime(journey\_date, format='%d/%m/%Y')

except ValueError:

print("Error: Invalid date format. Please enter in DD/MM/YYYY format.")

return

journey\_day = journey\_date.day

journey\_month = journey\_date.month

# Encode source and destination

try:

source\_encoded = source\_encoder.transform([source])[0]

destination\_encoded = destination\_encoder.transform([destination])[0]

except ValueError:

print("Error: The source or destination entered is not in the training data.")

return

# Extract hour from the current time

dep\_time = pd.to\_datetime('now') # Assuming current time as departure time

dep\_hour = dep\_time.hour

# Duration (fixed example for simplicity)

duration\_hours = 3

duration\_minutes = 30

# Prepare the input data for prediction

input\_data = pd.DataFrame({

'Source': [source\_encoded],

'Destination': [destination\_encoded],

'Journey\_day': [journey\_day],

'Journey\_month': [journey\_month],

'Dep\_hour': [dep\_hour],

'Duration\_hours': [duration\_hours],

'Duration\_minutes': [duration\_minutes]

})

# Predict the probability of price change

try:

price\_change\_probability = model.predict\_proba(input\_data)[:, 1][0]

except Exception as e:

print(f"Error during prediction: {e}")

return

# Convert probability to percentage

price\_change\_percentage = price\_change\_probability \* 100

# Output results with increase or decrease message

if price\_change\_probability > 0.5:

print(f"\nPrice will likely increase by {price\_change\_percentage:.2f}%")

else:

print(f"\nPrice will likely decrease by {price\_change\_percentage:.2f}%")

# Provide the best time to book

best\_booking\_time = "Best time to book: Try to book 1-2 months in advance"

print(best\_booking\_time)

# Run the function

get\_input\_and\_predict()