Introduction :

Flight price prediction plays a crucial role in the travel and airline industry, helping passengers make cost-effective booking decisions and assisting airlines in optimizing pricing strategies. The dynamic nature of airfare prices, influenced by factors such as demand, seasonality, and operational costs, makes it challenging to estimate ticket prices accurately.

This project aims to develop a **machine learning model** that predicts flight ticket prices based on key features, including airline, source, destination, date of journey, total travel duration, number of stops, and departure/arrival times. By leveraging historical flight data, we apply **data preprocessing techniques** such as handling missing values, extracting meaningful date-time attributes, and encoding categorical variables.

We use a **Random Forest Regressor**, a powerful ensemble learning method, to train the model on preprocessed data. The dataset is split into training and testing sets to evaluate model performance using the **R^2 score**, ensuring its predictive accuracy and reliability.

By implementing this machine learning solution, we aim to provide a **data-driven approach** for price forecasting, benefiting both travelers and businesses. Passengers can plan trips based on estimated airfare trends, while travel agencies and airline companies can enhance their revenue management strategies. The trained model is saved for future use, allowing real-time predictions and continuous improvements with updated data.

This project demonstrates the potential of machine learning in the **aviation sector**, offering a scalable and automated approach to flight price estimation. Future enhancements may include incorporating additional factors like weather conditions, holiday seasons, and seat availability to further refine prediction

Code :

```
import pandas as pd

import numpy as np

import pickle

from sklearn.model_selection import train_test_split
```

```
from sklearn.preprocessing import OneHotEncoder
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score

# Load dataset
df = pd.read_csv("Data_Train.csv")

# Handle missing values
df.dropna(inplace=True)

# Extract day and month from Date_of_Journey
df["Journey_day"] = pd.to_datetime(df["Date_of_Journey"],
format="%d/%m/%Y").dt.day
df["Journey_month"] = pd.to_datetime(df["Date_of_Journey"],
format="%d/%m/%Y").dt.month
df.drop(["Date_of_Journey"], axis=1, inplace=True)

# Convert departure and arrival times
df["Dep_hour"] = pd.to_datetime(df["Dep_Time"]).dt.hour
df["Dep_min"] = pd.to_datetime(df["Dep_Time"]).dt.minute
df.drop(["Dep_Time"], axis=1, inplace=True)



df["Arrival_hour"] = pd.to_datetime(df["Arrival_Time"]).dt.hour
df["Arrival_min"] = pd.to_datetime(df["Arrival_Time"]).dt.minute
df.drop(["Arrival_Time"], axis=1, inplace=True)



# Convert Duration to hours and minutes
duration = df["Duration"].str.extract(r'(?:(\d+)h)?(?:\s*(\d+)m)?').fillna(0)
df["Duration_hours"] = duration[0].astype(int)
```

```
df["Duration_mins"] = duration[1].astype(int)

df.drop(["Duration", "Route", "Additional_Info"], axis=1, inplace=True)


# Convert Total_Stops to numerical values

df.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4},
inplace=True)


from sklearn.preprocessing import OneHotEncoder


# One-hot encode categorical features

encoder = OneHotEncoder(drop="first", sparse_output=False) # Use sparse_output
instead of sparse

encoded_cols = encoder.fit_transform(df[["Airline", "Source", "Destination"]])


# Convert to DataFrame

encoded_df = pd.DataFrame(encoded_cols,
columns=encoder.get_feature_names_out())



df_final = df.drop(["Airline", "Source", "Destination"], axis=1)

df_final = pd.concat([df_final, encoded_df], axis=1)


# Handle any remaining NaN values

df_final.fillna(0, inplace=True)


# Split data into training and test sets

X = df_final.drop(["Price"], axis=1)


y = df_final["Price"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

```
# Train RandomForestRegressor
model = RandomForestRegressor(n_estimators=100,
random_state=42) model.fit(X_train, y_train)

# Evaluate the model
y_pred = model.predict(X_test)
accuracy = r2_score(y_test,
y_pred) print(f"Model R² Score:
{accuracy:.2f}")

# Save the trained model
with open("flight_price.pkl", "wb") as file:
pickle.dump(model, file)
print("Model saved as flight_price.pkl")
```

Output :