Project : Flight Booking Price Prediction

Name : Deepak Kumar
Roll No. : 2401030344
Email ID. : 2401030344@mail.jiit.ac.in
Semester. : 3rd Semester
Course. : Introduction to Python Programming & Machine Learning

Project Overview

This project predicts flight ticket prices based on features such as airline, source, destination, duration, stops, etc.

It uses Python libraries like Pandas, Matplotlib, Seaborn, and Scikit-learn.

GitHub Repository

I have uploaded the complete project on my GitHub repository.

You can view the source code, cleaned dataset, visualizations, and machine learning models here:

Flight Booking Price Prediction – GitHub Repository

https://github.com/Deepak152-coder/Flight-Booking-Price-Prediction

This repository includes:

- *Cleaned dataset (`Clean_Dataset.csv`)*
- *Final project notebook (`FinalProject.py`)*

- *Visualizations and EDA code*
- *Machine learning model training and evaluation*

Technologies Used

- *Python 3.13*
- *Pandas*
- *NumPy*
- *Seaborn & Matplotlib*
- *Scikit-learn*
- *VS Code*

Code:-

```
print("\n" + "-"*50 + " Project: Flight Booking
Price Prediction " + "-"*50 + "\n")

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns

df = pd.read_csv("Clean_Dataset.csv")

print(df.columns)
print("\n" + "-"*80 + "\n")

df = df.drop(columns=["Unnamed: @"],
errors='ignore')
```

```
print(df.head())
print("\n" + "-"*80 + "\n")

df.shape
df.info()
df.describe()

print(df.isnull().sum())
print("\n" + "-"*80 + "\n")

plt.figure(figsize=(15, 5))
sns.lineplot(x=df['airline'], y=df['price'])
plt.title('Airlines Vs Price', fontsize=15)
plt.xlabel('Airline', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.show()

plt.figure(figsize=(15, 5))
sns.lineplot(data=df, x='days_left', y='price',
color='blue')
plt.title('Days Left For Departure Versus Ticket
Price', fontsize=15)
plt.xlabel('Days Left for Departure', fontsize=15)
plt.ylabel('Price', fontsize=15)
plt.show()

plt.figure(figsize=(10, 5))
sns.barplot(x='airline', y='price', data=df)
plt.title("Price Range of All Airlines")
```

```
plt.xlabel("Airline")
plt.ylabel("Price")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

plt.figure(figsize=(10, 8))
sns.barplot(x='class', y='price', data=df,
hue='airline')
plt.title("Flight Prices by Class and Airline")
plt.xlabel("Class")
plt.ylabel("Price")
plt.tight_layout()
plt.show()

fig, ax = plt.subplots(1, 2, figsize=(20, 6))

sns.lineplot(x='days_left', y='price', data=df,
hue='source_city', ax=ax[0])
ax[0].set_title("Price vs Days Left by Source
City")
ax[0].set_xlabel("Days Left")
ax[0].set_ylabel("Price")

sns.lineplot(x='days_left', y='price', data=df,
hue='destination_city', ax=ax[1])
ax[1].set_title("Price vs Days Left by Destination
City")
ax[1].set_xlabel("Days Left")
ax[1].set_ylabel("Price")
```

```
plt.tight_layout()
plt.show()

plt.figure(figsize=(18, 28))

plt.subplot(4, 2, 1)
sns.countplot(x=df["airline"])
plt.title("Frequency of Airline")
plt.xticks(rotation=45)

plt.subplot(4, 2, 2)
sns.countplot(x=df["source_city"])
plt.title("Frequency of Source City")
plt.xticks(rotation=45)

plt.subplot(4, 2, 3)
sns.countplot(x=df["departure_time"])
plt.title("Frequency of Departure Time")
plt.xticks(rotation=45)

plt.subplot(4, 2, 4)
sns.countplot(x=df["stops"])
plt.title("Frequency of Stops")
plt.xticks(rotation=45)

plt.subplot(4, 2, 5)
sns.countplot(x=df["arrival_time"])
plt.title("Frequency of Arrival Time")
plt.xticks(rotation=45)
```

```
plt.subplot(4, 2, 6)
sns.countplot(x=df["destination_city"])
plt.title("Frequency of Destination City")
plt.xticks(rotation=45)

plt.subplot(4, 2, 7)
sns.countplot(x=df["duration"])
plt.title("Flight Duration Frequency")
plt.xticks([], [])

plt.tight_layout()
plt.show()

from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df["airline"] = le.fit_transform(df["airline"])
df["source_city"] =
le.fit_transform(df["source_city"])
df["departure_time"] =
le.fit_transform(df["departure_time"])
df["stops"] = le.fit_transform(df["stops"])
df["arrival_time"] =
le.fit_transform(df["arrival_time"])
df["destination_city"] =
le.fit_transform(df["destination_city"])
df["class"] = le.fit_transform(df["class"])
df.info()
print("\n" + "-"*80 + "\n")
```

```
plt.figure(figsize=(10,5))
df_numeric = df.select_dtypes(include=["int64",
"float64"])
sns.heatmap(df_numeric.corr(), annot=True,
cmap="coolwarm")
plt.show()

from statsmodels.stats.outliers_influence import
variance_inflation_factor
col_list = []
for col in df.columns:
    if ((df[col].dtype != 'object') & (col !=
'price')):
        col_list.append(col)

X = df[col_list]
vif_data = pd.DataFrame()
vif_data["feature"] = X.columns
vif_data["VIF"] =
[variance_inflation_factor(X.values, i) for i in
range(len(X.columns))]
print(vif_data)
print("\n" + "-"*80 + "\n")

df = df.drop(columns=["stops"])

X = df.drop(columns=["price"])
y = df["price"]

X = X.select_dtypes(exclude='object')
```

```
from sklearn.model_selection import
train_test_split
x_train, x_test, y_train, y_test =
train_test_split(X, y, test_size=0.2,
random_state=42)

from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

difference = pd.DataFrame({
    "Actual_Value": np.round(y_test.values, 2),
    "Predicted_Value": np.round(y_pred, 2)
})
pd.set_option('display.max_rows', 10)
print(difference)
print("\n" + "-"*80 + "\n")

from sklearn.metrics import r2_score,
mean_absolute_error, mean_squared_error,
mean_absolute_percentage_error
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mape = mean_absolute_percentage_error(y_test,
y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)
```

```python
print("\nEvaluation Metrics for Linear
Regression:\n")
print("R² Score:", r2)
print("Mean Absolute Error:", mae)
print("Mean Absolute Percentage Error:", mape)
print("Mean Squared Error:", mse)
print("Root Mean Squared Error:", rmse)
print("\n" + "-"*80 + "\n")

plt.figure(figsize=(10, 5))
sns.kdeplot(y_test, label="Actual", fill=True)
sns.kdeplot(y_pred, label="Predicted (Linear
Regression)", fill=True)
plt.title("Actual vs Predicted Flight Price
Distribution - Linear Regression")
plt.xlabel("Price")
plt.ylabel("Density")
plt.legend()
plt.show()

from sklearn.tree import DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train, y_train)
y_pred_dt = dt.predict(x_test)

r2_dt = r2_score(y_test, y_pred_dt)
mae_dt = mean_absolute_error(y_test, y_pred_dt)
mape_dt = mean_absolute_percentage_error(y_test,
y_pred_dt)
mse_dt = mean_squared_error(y_test, y_pred_dt)
```

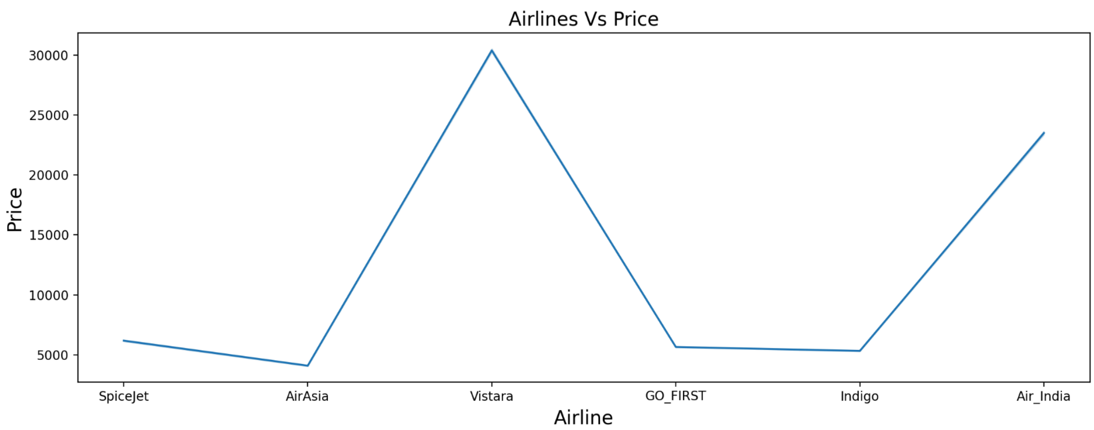
```
rmse_dt = np.sqrt(mse_dt)

print("\nEvaluation Metrics for Decision Tree:\n")
print("R² Score:", r2_dt)
print("Mean Absolute Error:", mae_dt)
print("Mean Absolute Percentage Error:", mape_dt)
print("Mean Squared Error:", mse_dt)
print("Root Mean Squared Error:", rmse_dt)
print("\n" + "-"*80 + "\n")

plt.figure(figsize=(10, 5))
sns.kdeplot(y_test, label="Actual", fill=True)
sns.kdeplot(y_pred_dt, label="Predicted (Decision
Tree)", fill=True)
plt.title("Actual vs Predicted Flight Price
Distribution - Decision Tree Regression")
plt.xlabel("Price")
plt.ylabel("Density")
plt.legend()
plt.show()

from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor()
rfr.fit(x_train, y_train)
y_pred_rf = rfr.predict(x_test)

r2_rf = r2_score(y_test, y_pred_rf)
mae_rf = mean_absolute_error(y_test, y_pred_rf)
mape_rf = mean_absolute_percentage_error(y_test,
y_pred_rf)
```

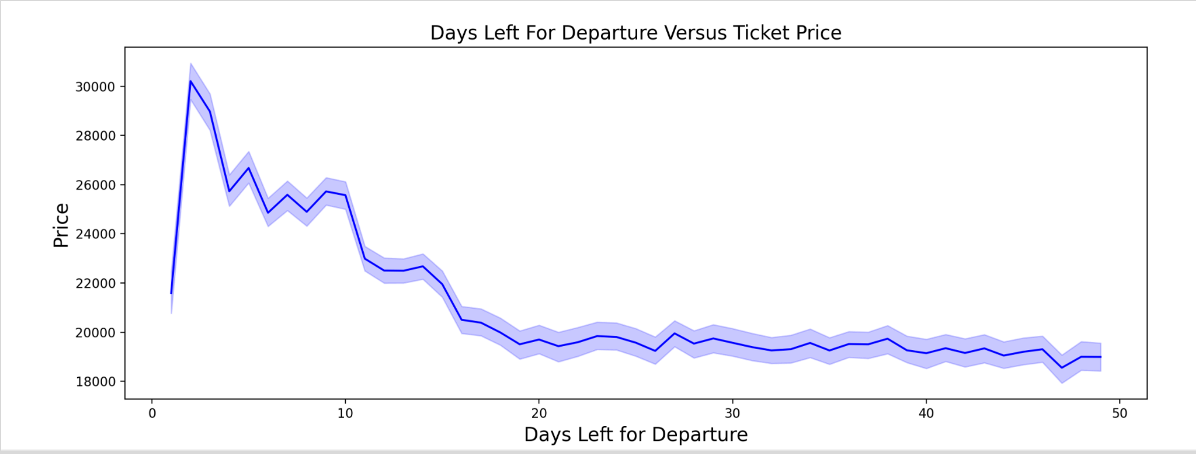
```python
mse_rf = mean_squared_error(y_test, y_pred_rf)
rmse_rf = np.sqrt(mse_rf)

print("\nEvaluation Metrics for Random Forest:\n")
print("R² Score:", r2_rf)
print("Mean Absolute Error:", mae_rf)
print("Mean Absolute Percentage Error:", mape_rf)
print("Mean Squared Error:", mse_rf)
print("Root Mean Squared Error:", rmse_rf)
print("\n" + "-"*80 + "\n")

plt.figure(figsize=(10, 5))
sns.kdeplot(y_test, label="Actual", fill=True)
sns.kdeplot(y_pred_rf, label="Predicted (Random
Forest)", fill=True)
plt.title("Actual vs Predicted Flight Price
Distribution - Random Forest Regression")
plt.xlabel("Price")
plt.ylabel("Density")
plt.legend()
plt.show()

print("\n" + "-"*55 + " End of Project " + "-"*55 +
"\n")
```

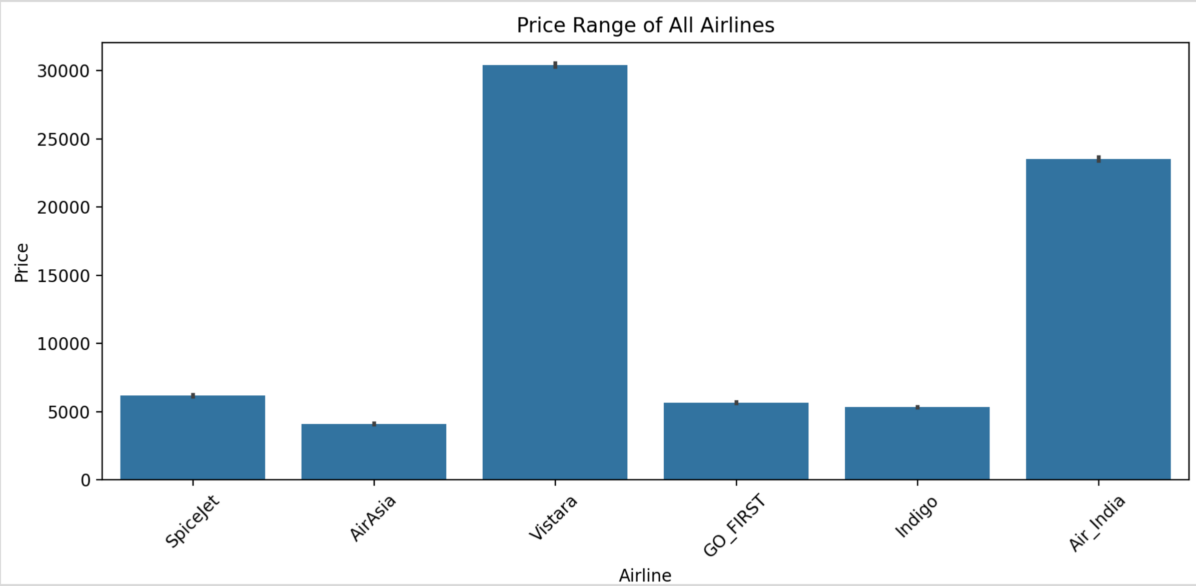
Output (screenshots):-

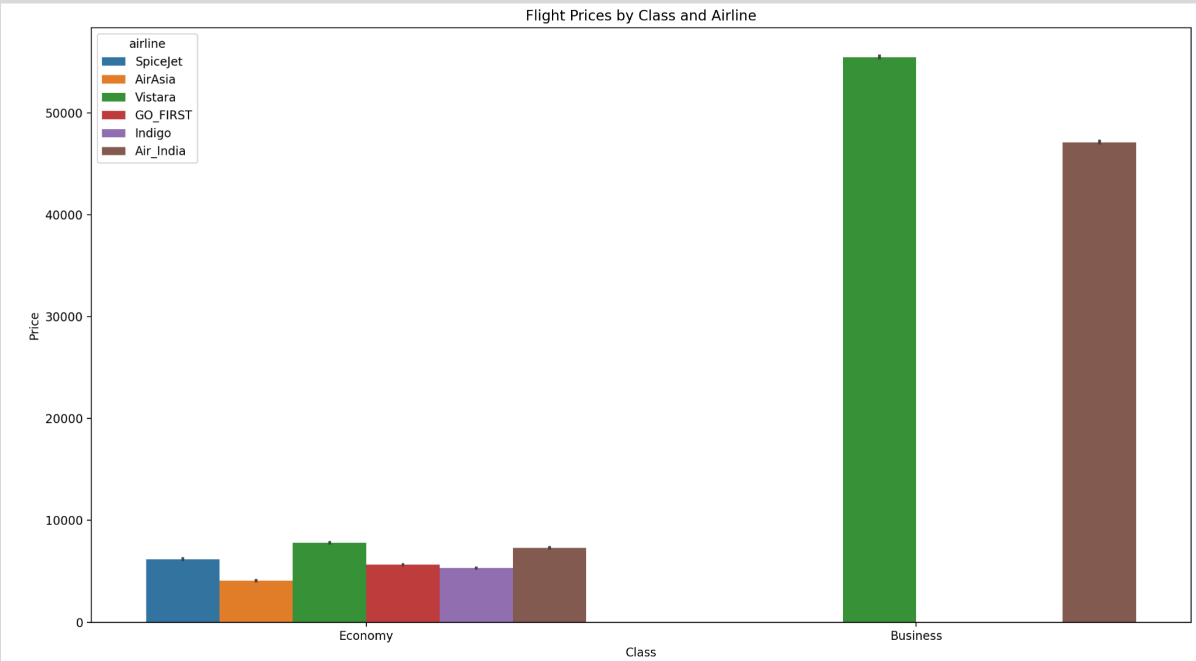
```
------------------------------------------------- Project: Flight Booking Price Prediction -------------------------------------------------

Index(['Unnamed: 0', 'airline', 'flight', 'source_city', 'departure_time',
       'stops', 'arrival_time', 'destination_city', 'class', 'duration',
       'days_left', 'price'],
      dtype='object')

-------------------------------------------------------------------------------

   Unnamed: 0   airline   flight source_city departure_time stops   arrival_time destination_city    class  duration  days_left  price
0           0  SpiceJet  SG-8709       Delhi        Evening  zero          Night           Mumbai  Economy      2.17          1   5953
1           1  SpiceJet  SG-8157       Delhi  Early_Morning  zero        Morning           Mumbai  Economy      2.33          1   5953
2           2   AirAsia   I5-764       Delhi  Early_Morning  zero  Early_Morning           Mumbai  Economy      2.17          1   5956
3           3   Vistara   UK-995       Delhi        Morning  zero      Afternoon           Mumbai  Economy      2.25          1   5955
4           4   Vistara   UK-963       Delhi        Morning  zero        Morning           Mumbai  Economy      2.33          1   5955

-------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        300153 non-null  int64
 1   airline           300153 non-null  object
 2   flight            300153 non-null  object
 3   source_city       300153 non-null  object
 4   departure_time    300153 non-null  object
 5   stops             300153 non-null  object
 6   arrival_time      300153 non-null  object
 7   destination_city  300153 non-null  object
 8   class             300153 non-null  object
 9   duration          300153 non-null  float64
 10  days_left         300153 non-null  int64
 11  price             300153 non-null  int64
dtypes: float64(1), int64(3), object(8)
memory usage: 27.5+ MB
Unnamed: 0          0
airline             0
flight              0
source_city         0
departure_time      0
stops               0
arrival_time        0
destination_city    0
class               0
duration            0
days_left           0
price               0
dtype: int64

--------------------------------------------------------------------------------
```




Days Left For Departure Versus Ticket Price
Price
Days Left for Departure
30000
28000
26000
24000
22000
20000
18000
0
10
20
30
40
50


Price Range of All Airlines
Price
Airline
SpiceJet
AirAsia
Vistara
GO_FIRST
Indigo
Air_India
30000
25000
20000
15000
10000
5000
0


Flight Prices by Class and Airline
airline
SpiceJet
AirAsia
Vistara
GO_FIRST
Indigo
Air_India
Price
Class
Economy
Business
50000
40000
30000
20000
10000
0

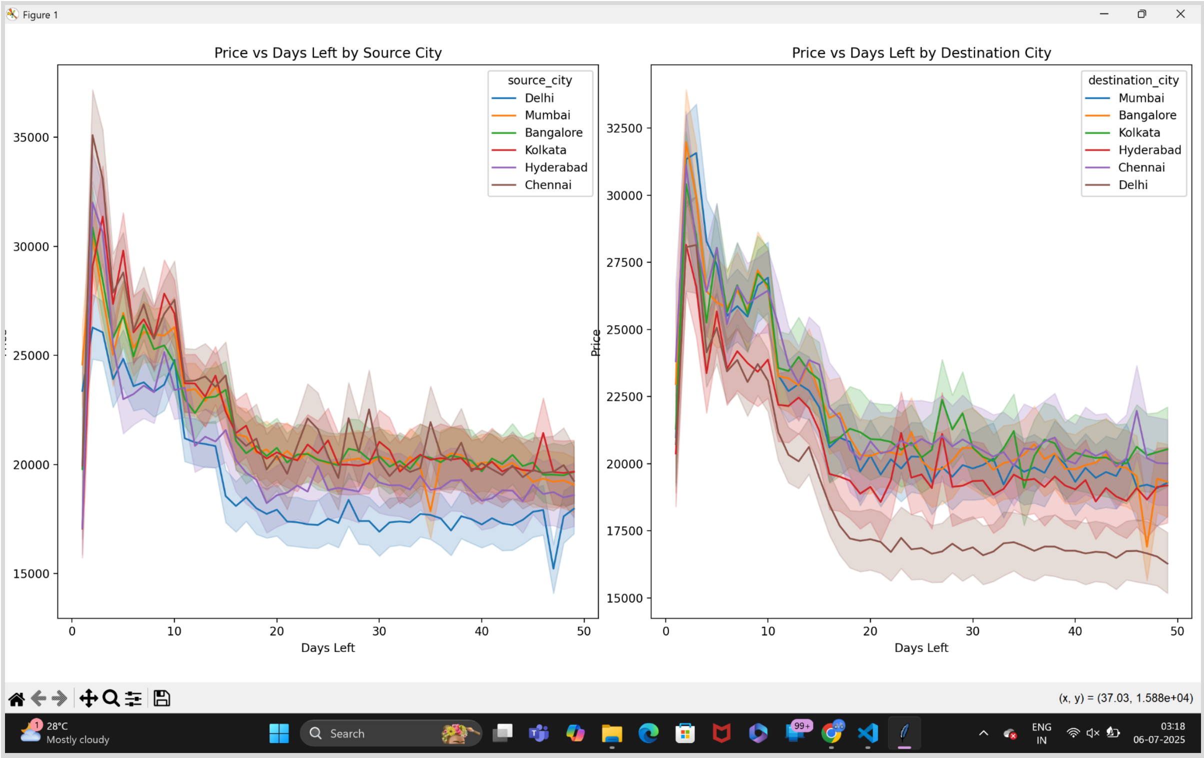


Figure 1
Price vs Days Left by Source City
source_city
Delhi
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
Days Left
Price vs Days Left by Destination City
destination_city
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
Delhi
Price
Days Left
(x, y) = (37.03, 1.588e+04)

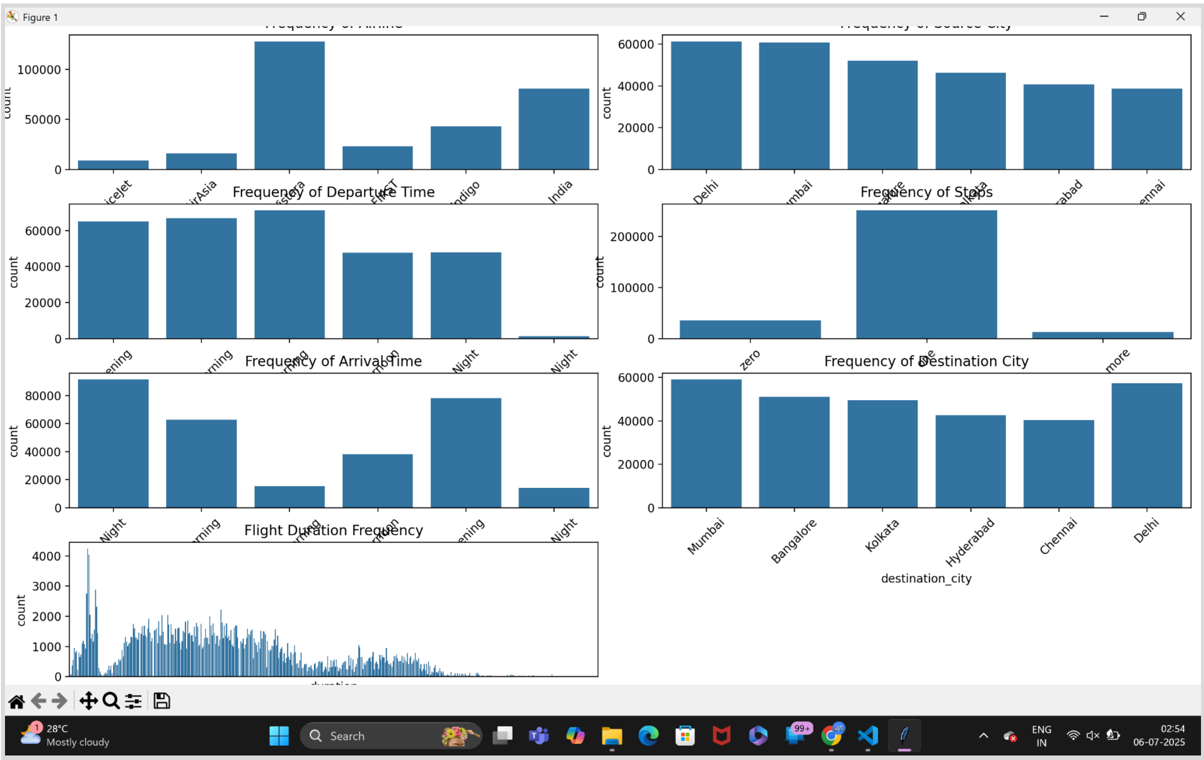
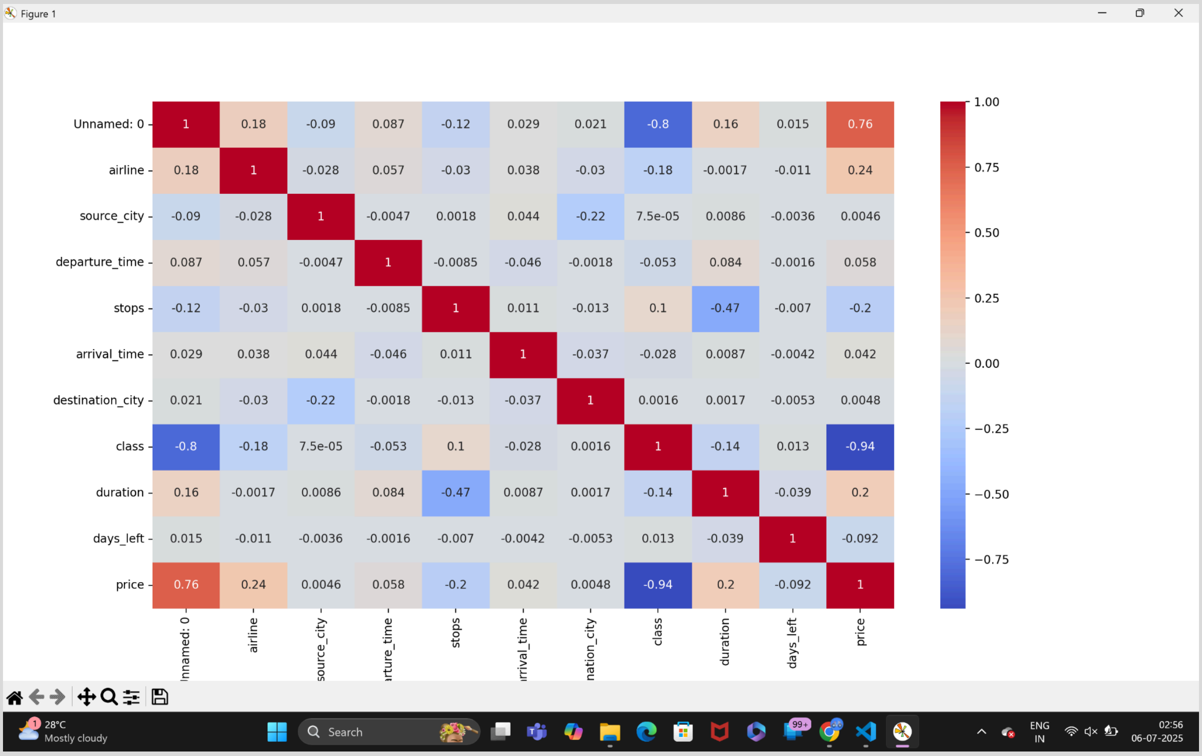


Figure 1
count
Frequency of Departure Time
Frequency of Stops
zero
more
Frequency of Arrival Time
Frequency of Destination City
Mumbai
Bangalore
Kolkata
Hyderabad
Chennai
Delhi
destination_city
Flight Duration Frequency

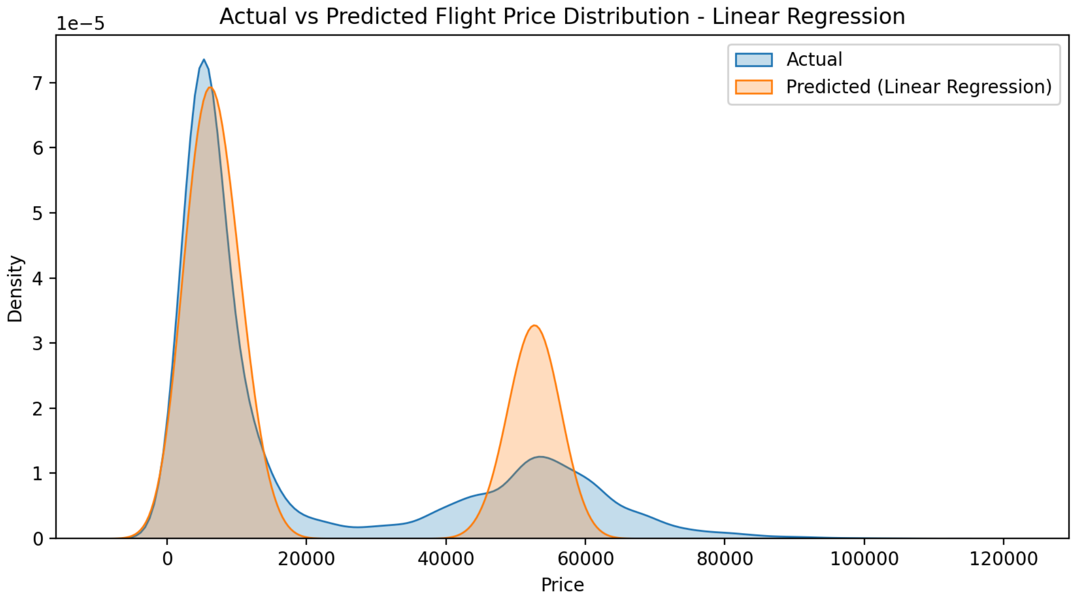
```
--------------------------------------------------------------------------------

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 300153 entries, 0 to 300152
Data columns (total 12 columns):
 #   Column            Non-Null Count   Dtype
---  ------            --------------   -----
 0   Unnamed: 0        300153 non-null  int64
 1   airline           300153 non-null  int64
 2   flight            300153 non-null  object
 3   source_city       300153 non-null  int64
 4   departure_time    300153 non-null  int64
 5   stops             300153 non-null  int64
 6   arrival_time      300153 non-null  int64
 7   destination_city  300153 non-null  int64
 8   class             300153 non-null  int64
 9   duration          300153 non-null  float64
 10  days_left         300153 non-null  int64
 11  price             300153 non-null  int64
dtypes: float64(1), int64(10), object(1)
memory usage: 27.5+ MB

--------------------------------------------------------------------------------
```



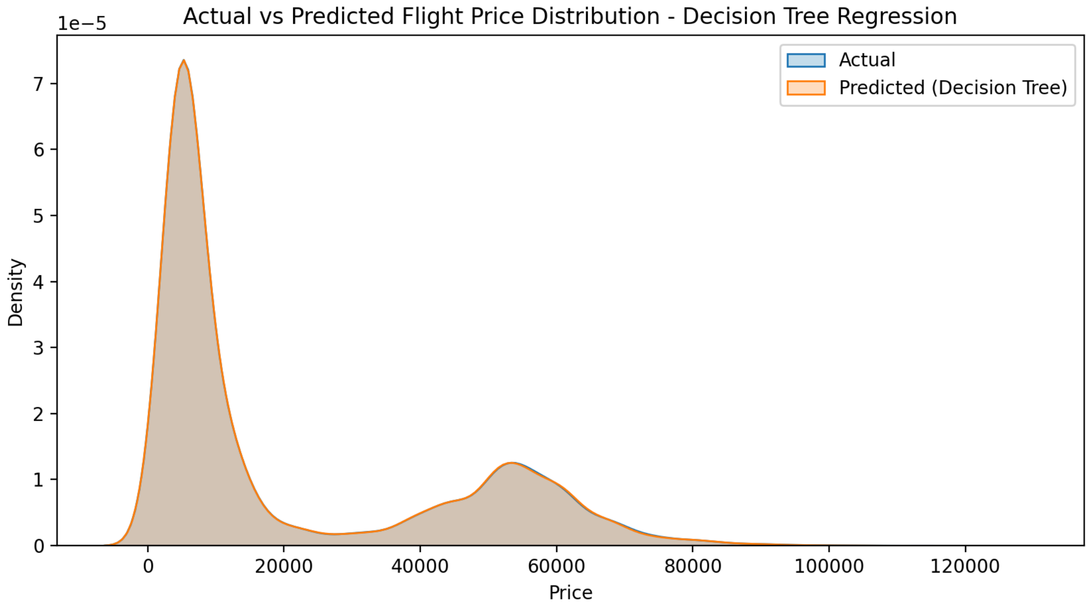

```
--------------------------------------------------------------------------------

          feature       VIF
0      Unnamed: 0  6.420074
1         airline  3.776008
2     source_city  2.981667
3  departure_time  2.908666
4           stops  1.460119
5    arrival_time  3.922821
6  destination_city  3.114813
7           class  4.512035
8        duration  4.701263
9       days_left  4.385177

--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------

       Actual_Value  Predicted_Value
0              7366          4360.60
1             64831         51759.92
2              6195          6740.62
3             60160         55571.84
4              6578          5089.16
...             ...              ...
60026          5026          4608.09
60027          3001          4338.02
60028          6734          5234.68
60029          5082          3015.97
60030         66465         59700.01

[60031 rows x 2 columns]

--------------------------------------------------------------------------------
```

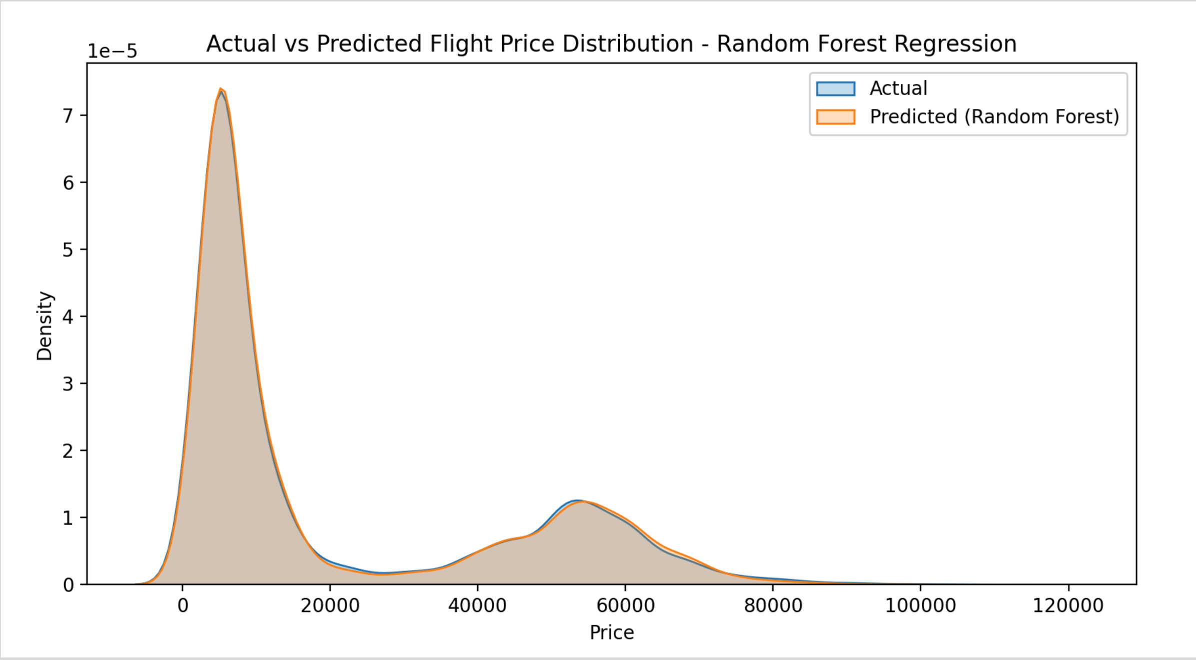
```
---------------------------------------------------------------------------------

Evaluation Metrics for Linear Regression:

R² Score: 0.8978129229043199
Mean Absolute Error: 4471.671851098374
Mean Absolute Percentage Error: 0.3489698343342798
Mean Squared Error: 52675626.83063885
Root Mean Squared Error: 7257.797657047133

---------------------------------------------------------------------------------
```




```
----------------------------------------------------------------------------

Evaluation Metrics for Decision Tree:

R² Score: 0.9794973673601999
Mean Absolute Error: 1018.0866385700722
Mean Absolute Percentage Error: 0.06277874517684691
Mean Squared Error: 10568743.687311556
Root Mean Squared Error: 3250.9604253684106

----------------------------------------------------------------------------
```



```
--------------------------------------------------------------------------------

Evaluation Metrics for Random Forest:

R² Score: 0.9870516375326348
Mean Absolute Error: 992.7353355766187
Mean Absolute Percentage Error: 0.06290489400939815
Mean Squared Error: 6674651.323671272
Root Mean Squared Error: 2583.534656951842

--------------------------------------------------------------------------------

------------------------------------------------------- End of Project -------------------------------------------------------
```



Conclusion :-

This project successfully demonstrates how flight ticket prices can be predicted using machine learning models.
Through data preprocessing, feature analysis, and model evaluation, we explored multiple approaches — with Random Forest showing the most accurate results.

GitHub Repository :-

You can find the complete project and code here:

[*Flight Booking Price Prediction – GitHub Repository*](https://github.com/Deepak152-coder/Flight-Booking-Price-Prediction)
https://github.com/Deepak152-coder/Flight-Booking-Price-Prediction

Acknowledgements :-

Thanks to:

=> Datasets from Flight Price Prediction

=> Mentors :-

Himani Gupta (himani.gupta@mail.jiit.ac.in)

Ankur Gupta (ankur.gupta@mail.jiit.ac.in)

=> Python open-source community