PET ENGINEERING COLLEGE

VALLIOOR - 6271171.

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

PROJECT TOPIC: DYNAMIC PRICING

College Code: 9632

Technology: Artificial Intelligence

Total number of student's in a group: 5

Student's details within the group:

- Asma Sahanaz A [Reg no:963222104011]
- Balasundhari S [Reg no:963222104012]
- **Bhavani S [Reg no:963222104013]**
- Cinesh A [Reg no:963222104014]
- Deepa S [Reg no:963222104015]

Table of contents:

Abstract:

This project dives into the creation of " Recommending what you'll love: dynamic pricing." This system analyzes market conditions, consumer behavior, and competitor pricing in real time to adjust prices dynamically, maximizing revenue and profit. By harnessing the power of python's data science libraries, it's implementing dynamic pricing strategies due to versatility and extensive libraries.

Introduction:

Dynamic pricing is an advanced strategy that adjusts prices based on real-time demand and market conditions. This project explores dynamic pricing through comprehensive data collection, visualization, and modeling techniques. We gather extensive datasets on market trends, consumer behavior, and pricing histories. These datasets are then visualized to identify patterns and insights. Utilizing a Random Forest regression model, we predict optimal pricing strategies to maximize revenue. This approach not only enhances pricing accuracy but also adapts swiftly to market fluctuations, ensuring competitive advantage.

Objectives:

1. **Develop a Random Forest Regression Model**: Implement and refine the Random Forest Regression algorithm to suit the dynamic pricing model, ensuring it capture the nuances of pricing data.
2. **Evaluate Model Performance**: Utilize a comprehensive set of evaluation metrics, including Mean Absolute Error (MAE), Mean Squared Error (MSE), and R-squared (R^2), to rigorously assess the accuracy and reliability of the model.
3. **Validate Generalization Ability**: Ensure the model generalizes well to unseen data by employing techniques such as cross-validation and testing on a hold-out dataset to prevent overfitting and underfitting.

4. **Prepare Model for Deployment**: Finalize a robust, high-performing Random Forest Regression model that is ready for deployment in dynamic pricing applications, ensuring it meets business objectives and performance standards.

Methodology:

Python offers a powerful toolkit for implementing dynamic pricing strategies. Here's how you can leverage python for each step of the methodology:

1. **Data collection:**
    - We'll gather user interaction data such as sales figures,inventory levels, customer behavior data.
2. **Data Cleaning and preprocessing:**
    - Libraries: Pandas and Numpy provide functionalities for:
    - Handling missing values and outliers.
    - Standardizing data formats.
3. **Data visualization:**
    - Libraries like Matplotlib or Seaborn help visualize data and model outputs for better understanding and communication.
4. **Model Building:**
    - Python offers various machine learning libraries like scikit-lean, TensorFlow, and PyTorch to build different types of models for dynamic pricing (Regression models, classification models, reinforcement learning models).
5. **Model Evaluation:**
    - Libraries like scikit-learn provide metrics to evaluate model performance (eg.,mean squared error for regression models).

Dataset Description:

The dataset comprises user interaction data collected from a digital platform, including information about user profiles, content items, and user interactions such as time, price and date. Each row in the dataset represents a user's interaction with a specific content item, forming the foundation for personalized content.

Data Wrangling Techniques:

1.Data Description:

- **Head:** Displaying the first few rows of the dataset to get an initial overview.
- **Tail:** Examining the last few rows of the dataset to ensure completeness.
- **Info:** Obtaining information about the dataset structure, data types, and memory usage.
- **Describe:** Generating descriptive statistics for numerical features to understand their distributions and central tendencies.

2. Null Data Handling :

- **Null Data Identification:** Identifying missing values in the dataset.
- **Null Data Imputation:** Filling missing values with appropriate strategies.
- **Null Data Removal:** Eliminating rows or columns with excessive missing values.

3. Data Validation:

- **Data Integrity Check:** Verifying data consistency and integrity to eliminate errors.
- **Data Consistency Verification:** Ensuring data consistency across different columns or datasets.

4.Data Reshaping:

- **Reshaping Rows and Columns:** Transforming the dataset into a suitable format for analysis.
- **Transposing Data:** Converting rows into columns and vice versa as needed.

5.Data Merging:

- Combining Datasets: Merging multiple datasets or data source to enrich the information available for analysis.
- Joining Data: Joining datasets based on common columns or keys.

6.Data Aggregation:

- Grouping Data: Grouping dataset row based on specific criteria.
- Aggregating Data: Computing summary statistics for grouped data.

Data Analysis Techniques:

7.Exploratory Data Analysis:

- **Univariate Analysis**: Analyzing individual variables to understand their• distributions and characteristics.
- **Bivariate Analysis:** Investigating relationship between pairs to variables to identify correlations and dependencies.
- **Multivariate Analysis:** Exploring interactions among multiple variable to uncover complex patterns and trends.

8. Feature Engineering:

- **Creating User Profiles:** Aggregating user interaction data to construct comprehensive user profiles capturing preferences and behaviors.

- **Temporal Analysis:** Incorporating temporal features such as time of the day or day of week to capture temporal trends in user behavior.
- **Content Embeddings:** Generating embeddings for content items to represent their characteristics and relationships

Assumed Scenario:

- **Scenario:** The project aims to recommend personalized content to users based on their historical interactions and preferences. The project aims to provide stakeholders with interactive visualizations to explore user interaction data and gain insights into user behavior and preferences.
- **Objective:** Ensure user engagement and satisfaction by delivering relevant• and tailored content recommendations. Enhance decision making and understanding through intuitive visual representations of data
- **Target Audience:** Digital platform users seeking personalized content recommendations across various domains.Project stakeholders including data analysts, product managers, and executives seeking actionable insights from the dataset.

Data Visualization Techniques:

1.Univariate Visualizations:

- Histograms: Displaying the distribution of numerical variables.
- Bar Charts: Visualizing the frequency distribution of categorical variables.

2.Bivariate Visualizations:

- Scatter Plots: Showing the relationship between two numerical variables.
- Box Plots: Illustrating the distribution of a numerical variables across different categories.

3.Multivariate Visualizations:

- Pair Plot: Visualizing pairwise relationships between multiple numerical variables.

4.Interactive Visualizations:

- Interactive Scatter Plots: Providing tooltips or zooming functionality for enhanced exploration.
- Interactive Dashboards: Creating dynamic dashboards to allow users to interact with visualizations.

Model Development:

1.Algorithm Selection:

- **Choice of Random Forest Regression**: We selection Random Forest Regression due to its robustness in handling large datasets with high dimensionality and its ability to model complex, non-linear relationships essential for dynamic pricing.
- **Feature Importance Analysis**: Random Forest Regression provides insights into feature importance, enabling us to identify and prioritize key factors that influence pricing decisions, thereby improving model interpretability and strategic decision-making.

2.Model Training:

- **Data Preprocessing**: We initiate model training by preprocessing the data, which includes handling missing values and numerical features to ensure the dataset is clean and ready for modeling.

- **Model Training Execution**: With the preprocessed data, we train the Random Forest Regression model on the training dataset, ensuring that the model learns from the data patterns and relationships necessary for accurate dynamic pricing predictions.

3.Model Evaluation:

- **Evaluation Metrics**: We evaluate the model using key metrics such as Mean Absolute Error(MAE), Mean Squared Error(MSE), and R-squared(R^2) to quantitatively assess the model's accuracy and reliability in predicting prices.
- **Cross-Validation and Generalization**: To validate the model's performance and ensure it generalizes well to new, unseen data, we employ cross-validation techniques and test the model on a separate hold-out dataset, thereby confirming its robustness and effectiveness.

Evaluation Metrics:

Accuracy Metrics:

- **Mean Absolute Error (MAE):** MAE measures the average absolute difference between predicted and actual values, providing a direct assessment of prediction accuracy without considering the direction of errors.
- **Mean Squared Error (MSE):** MSE calculates the average squared difference between predicted and actual values, emphasizing larger errors. It provides a measure of overall model accuracy by considering the magnitude of errors.
- **R-squared (R^2):** R^2 indicates the proportion of the variance in the dependent variable that is predictable from the independent variables. Higher R^2 values signify better model fit and predictive accuracy, with 1 representing a perfect fit.

Model Selection:

- **Random Forest Regression**: Selected for its ability to handle complex, non-linear relationships in dynamic pricing data and its robustness to overfitting.
- **Consideration of Alternatives**: Evaluated other regression algorithms but chose Random Forest Regression due to its superior performance and interpretability.
- **Alignment with Project Goal:** Random Forest Regression aligns well with the project's objectives of accurate pricing predictions and robust model performance

Existing Work:

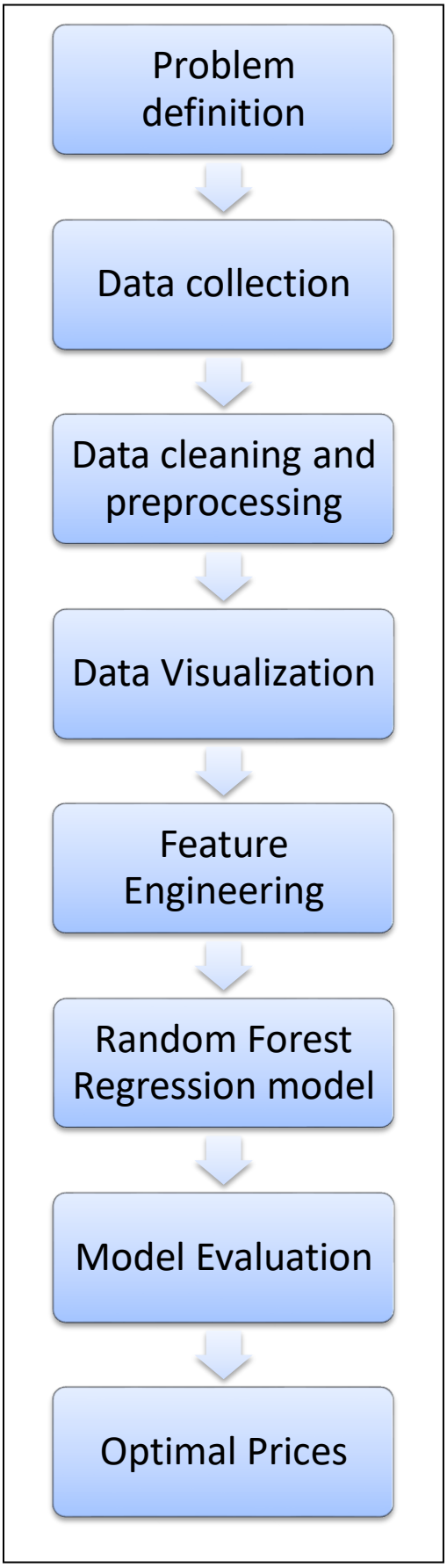
Existing work show machine learning excelling in dynamic pricing. From airlines using it for ticket prices to ride-sharing services for fares, it's transforming how business adapt to changing markets. Research explores various models (regression , classification, reinforcement learning) to optimize pricing strategies based on data and customer behavior.

Proposed Work:

The project proposes a novel approach to dynamic pricing that leverages machine learning for both personalization and fairness. Here's breakdown:

1. **Customer Segmentation**: We'll use machine learning to group customers based on purchase history and price sensitivity, creating targeted pricing strategies.
2. **Fairness-Aware Pricing Model:** A machine learning model will predict optimal prices for each segment, considering demand, competition, and fairness constraints to avoid disadvantaging any group.
3. **Real-time Price Recommendations:** we'll integrate real-time data feeds(inventory, competitor prices) for continuous price adjustments and personalized recommendations to customers.

Flow chart:


Problem definition
Data collection
Data cleaning and preprocessing
Data Visualization
Feature Engineering
Random Forest Regression model
Model Evaluation
Optimal Prices

Code:

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import dash
import dash_core_components as dcc
import dash_html_components as html
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

data=pd.read_csv("Dataset.csv")
#data=pd.DataFrame(data)
#head
data=data.head(10)
print("the head of the dataset:")
print(data)
#tail
print("the tail of the dataset:")
print(data.tail())
#info
print("the info of the dataset:")
print(data.info())
#describe
print("Describe about the dataset:")
print(data.describe())
#null data identifying
print("Null data identification:")
print(data.isnull().sum())
print(pd.isnull(data))
#replace empty cells into value
```

```
print("Null Imputation:")
print(data.fillna(100))
#remove the null data in the dataset
print("Null Data Removal:")
print(data.dropna())
#checking datatype of the column
print("datatype of the columns:")
print(data.dtypes)
#find duplicates
print("duplicates in the dataset:")
print(data.duplicated())
#remove duplicates
print("after removing the duplicates in the dataset:\n")
print(data.drop_duplicates)
#data consistency
#checking if a column contains unique
print(data["class"].unique())
#data reshaping
#transpose the dataframe
print("the transpose of the dataframe:")
print(data.T)
#data merging
data1=data.head(2)
data2=data.tail(2)
print(pd.concat([data1,data2], axis=0))
#data aggregation
print("data aggregation:")
data=pd.DataFrame(data)
print("grouping data:")
print(data.groupby('flight')['price'].sum())
print("aggregating data:")
print(data.agg({'price': 'sum'}))
```

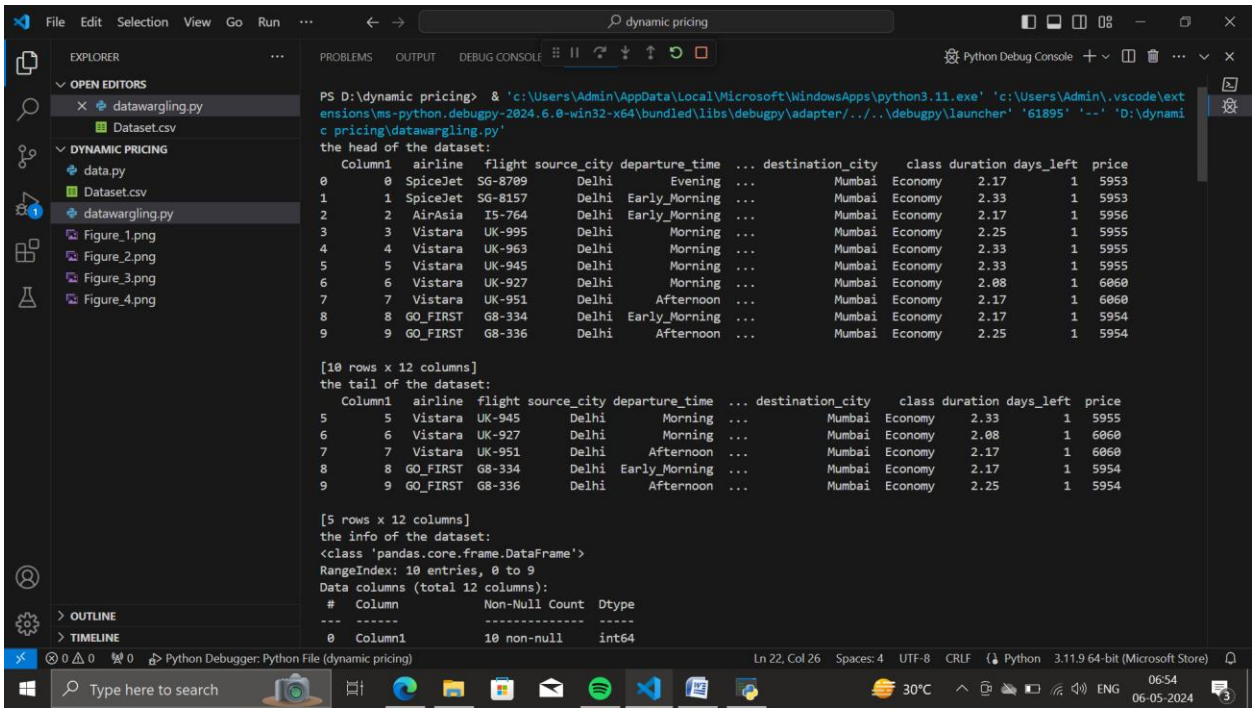
```
#Univariate analysis - histogram
plt.hist(data["duration"])
plt.title("fligt duration")
plt.show()
#Bivariate analysis - Scatter plot
x=data["airline"]
y=data["flight"]
plt.scatter( x,y)
plt.xlabel("Airline")
plt.ylabel("Flight")
plt.grid(True)
plt.title("Flights in the airline")
plt.show()
#Multivariate analysis - Pair plot
sns.pairplot(data)
plt.show()
#create user profile
print("\nuser profile:\n")
def create_user_profile(username, email, age, country):
    user_profile = {
        "username": username,
        "email": email,
        "age": age,
        "country": country
    }
    return user_profile
#Temporal Analysis
user_profile = create_user_profile("bhavani", "sb.bhavani.sb@gmail.com", 19, "India")
print(user_profile)
data.set_index('duration', inplace=True)
plt.figure(figsize=(10, 6))
data['price'].plot()
```

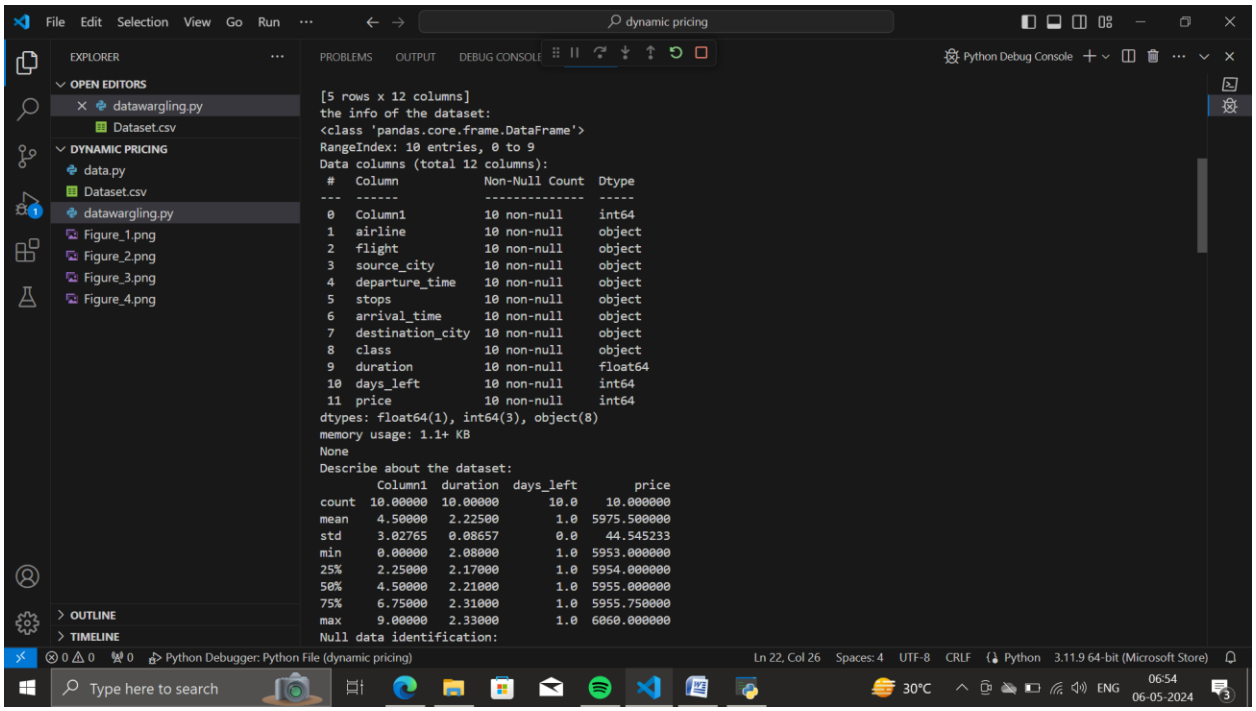
```
plt.title('Temporal Analysis')
plt.xlabel('Duration')
plt.ylabel('Price')
plt.grid(True)
plt.show()
#import the dataset
data=pd.read_csv("Dataset.csv")
data=data.head(10)
#Univariate analysis - histogram
plt.hist(data["duration"])
plt.title("Histogram")
plt.xlabel("Duration")
plt.ylabel("Frequency")
plt.show()
#Univariate analysis - bar chart
plt.bar(data['airline'].value_counts().index,
data['duration'].value_counts().values)
plt.xlabel("Airline")
plt.ylabel("Duration")
plt.title("Bar Chart ")
plt.show()
#Biunivariate analysis - scatter plot
x = data["duration"]
y = data["price"]
plt.scatter(x, y, color='red', marker='o')
plt.grid(True)
plt.xlabel("Duration")
plt.ylabel("Price")
plt.title("Scatter Plot")
plt.show()
#Biunivariate analysis - Box plot
x = data["airline"]
y = data["price"]
sns.boxplot(x="airline",y="price",data=data)
plt.xlabel('Airline')
plt.ylabel('Price')
plt.title('Box Plot')
plt.show()
#Multivariate visualization - pair plot
sns.pairplot(data)
```

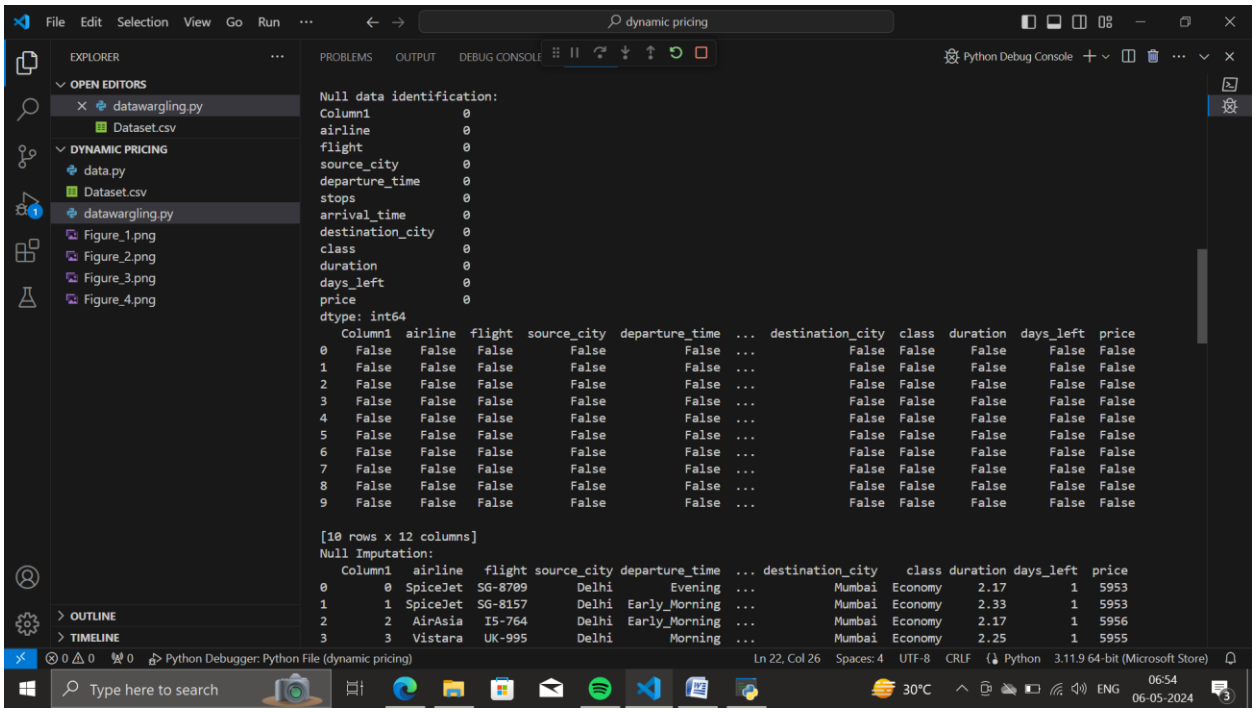
```
plt.title('Pair Plot')
plt.show()
#Interactive visualization - Scatter plot
fig = px.scatter(data, x='flight', y='price', hover_data=['duration'])
fig.show()
#Interactive visualization - Dashboard
app = dash.Dash(__name__)
app.layout = html.Div([
 dcc.Graph(
 id='interactive-plot',
 figure={
 'data': [
 {'x': data['flight'], 'y': data['arrival_time'], 'mode': 'markers', 'type': 'dashboard'}
 ],
 'layout': {
 'title': 'Interactive Scatter Plot',
 'xaxis': {'title': 'Flight'},
 'yaxis': {'title': 'Arrival_time'}
 }
 }
 )
])
if __name__ == '__main__':
 app.run_server(debug=True)
# Load the dataset
dataset = pd.read_csv('dataset.csv')
# Display the first few rows
print(dataset.head())
# Check for missing values and handle them
print(dataset.isnull().sum())
dataset = dataset.dropna()
X = dataset[['duration','days_left']]
y = dataset['price']
# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train recommendation models using the training data
rf_regressor = RandomForestRegressor(n_estimators=100, random_state=42)
rf_regressor.fit(X_train, y_train)

# Make predictions on the test set
```

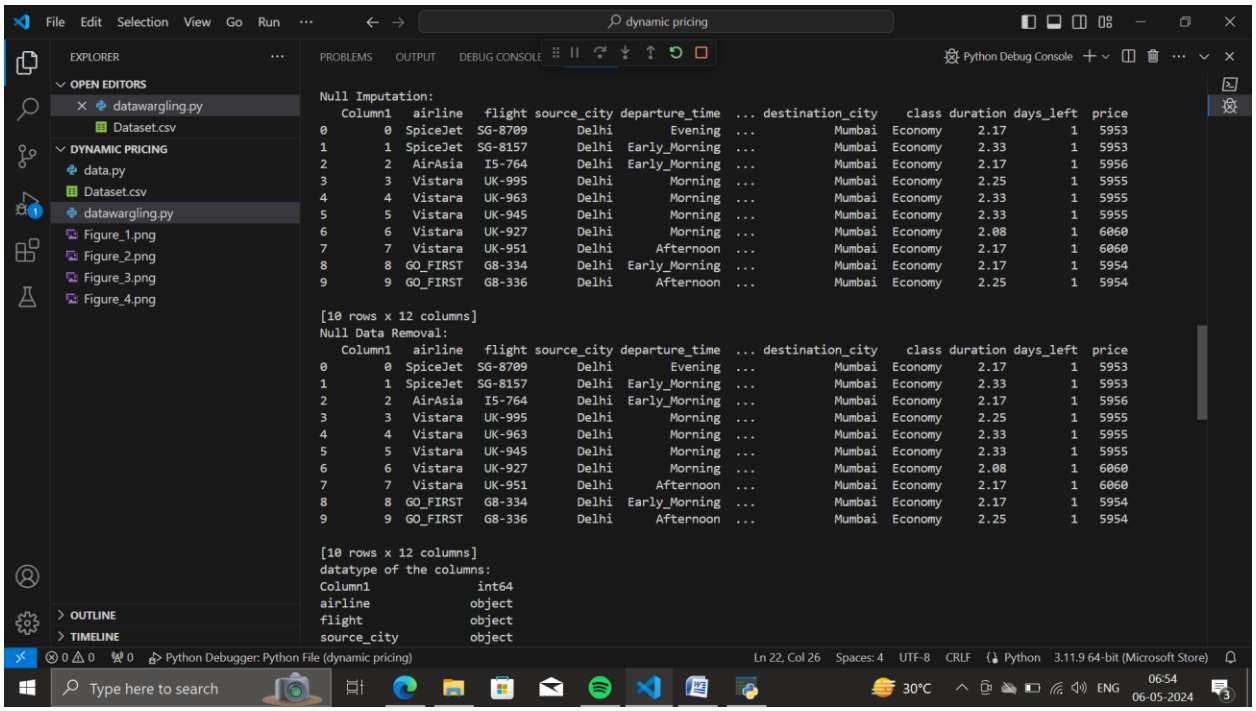
```
y_pred = rf_regressor.predict(X_test)

# Calculate evaluation metrics
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print("Mean Squared Error (MSE):", mse)
print("Mean Absolute Error (MAE):", mae)
print("R-squared (R2) Score:", r2)
```

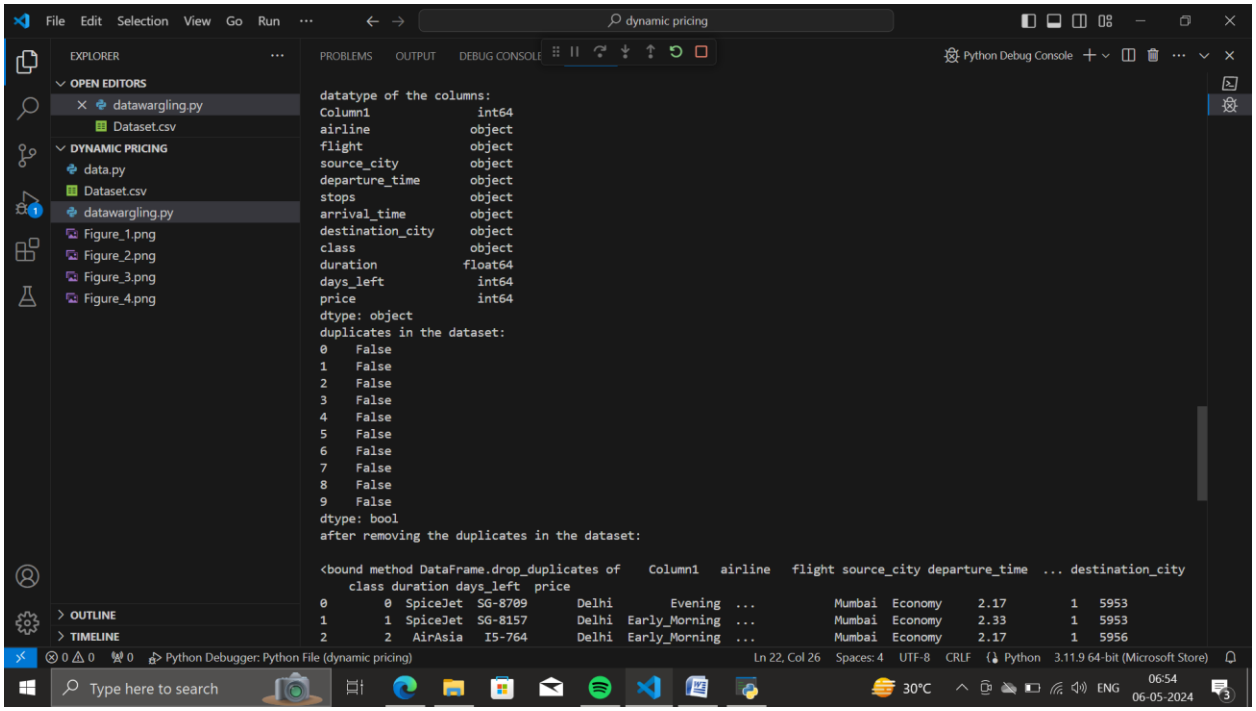
Output:

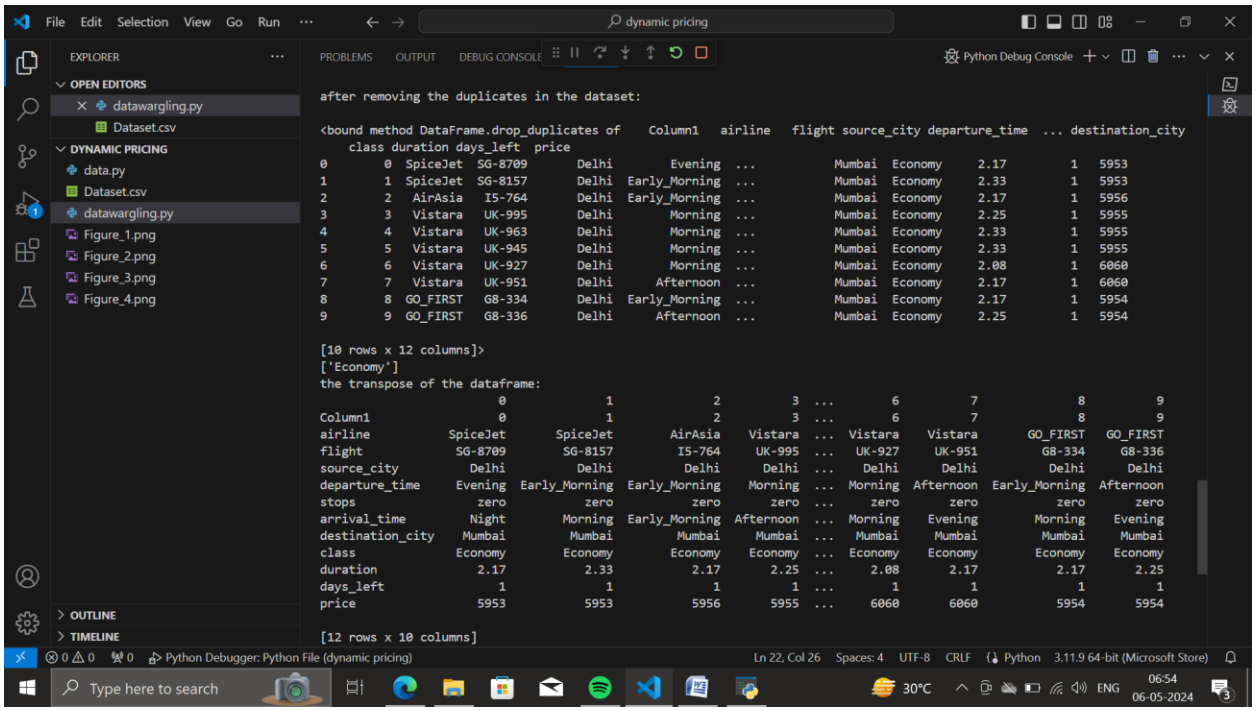


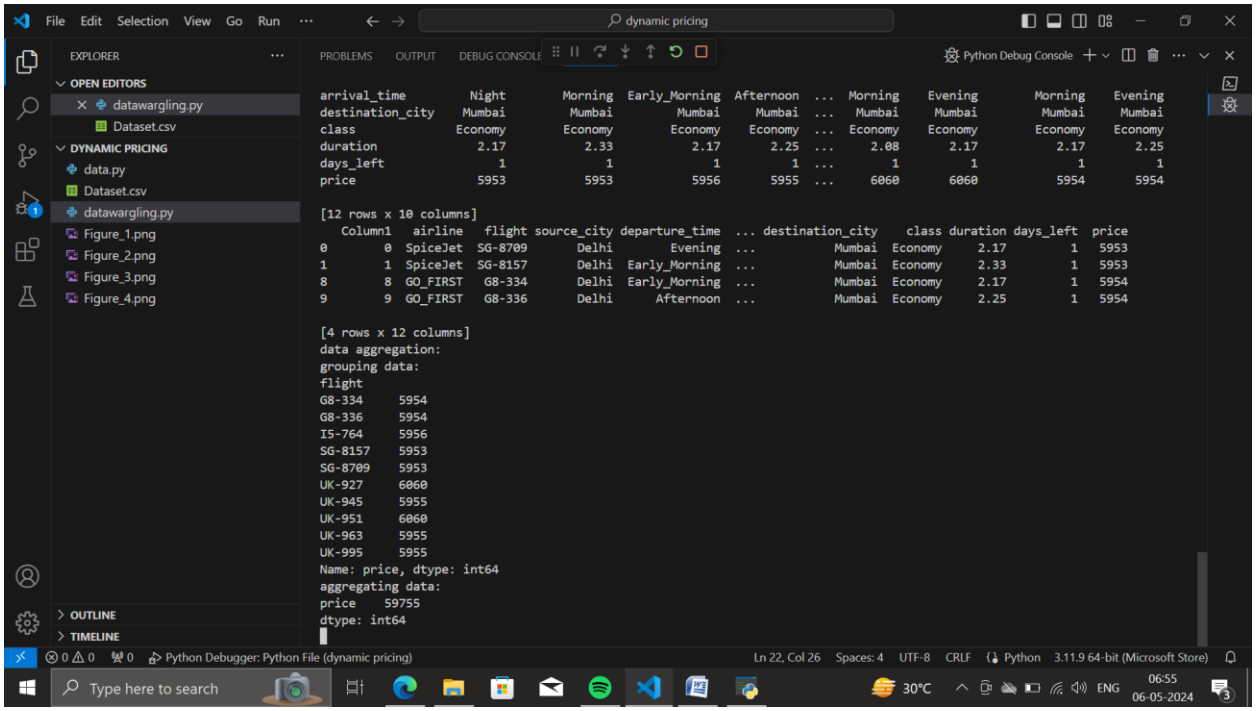

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
[5 rows x 12 columns]
the info of the dataset:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10 entries, 0 to 9
Data columns (total 12 columns):
Column Non-Null Count Dtype
--- ------ -------------- -----
0 Column1 10 non-null int64
1 airline 10 non-null object
2 flight 10 non-null object
3 source_city 10 non-null object
4 departure_time 10 non-null object
5 stops 10 non-null object
6 arrival_time 10 non-null object
7 destination_city 10 non-null object
8 class 10 non-null object
9 duration 10 non-null float64
10 days_left 10 non-null int64
11 price 10 non-null int64
dtypes: float64(1), int64(3), object(8)
memory usage: 1.1+ KB
None
Describe about the dataset:
Column1 duration days_left price
count 10.00000 10.00000 10.0 10.000000
mean 4.50000 2.22500 1.0 5975.500000
std 3.02765 0.08657 0.0 44.545233
min 0.00000 2.08000 1.0 5953.000000
25% 2.25000 2.17000 1.0 5954.000000
50% 4.50000 2.21000 1.0 5955.000000
75% 6.75000 2.31000 1.0 5955.750000
max 9.00000 2.33000 1.0 6060.000000
Null data identification:
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Python Debugger: Python File (dynamic pricing)
Type here to search
30°C
ENG
06:54
06-05-2024

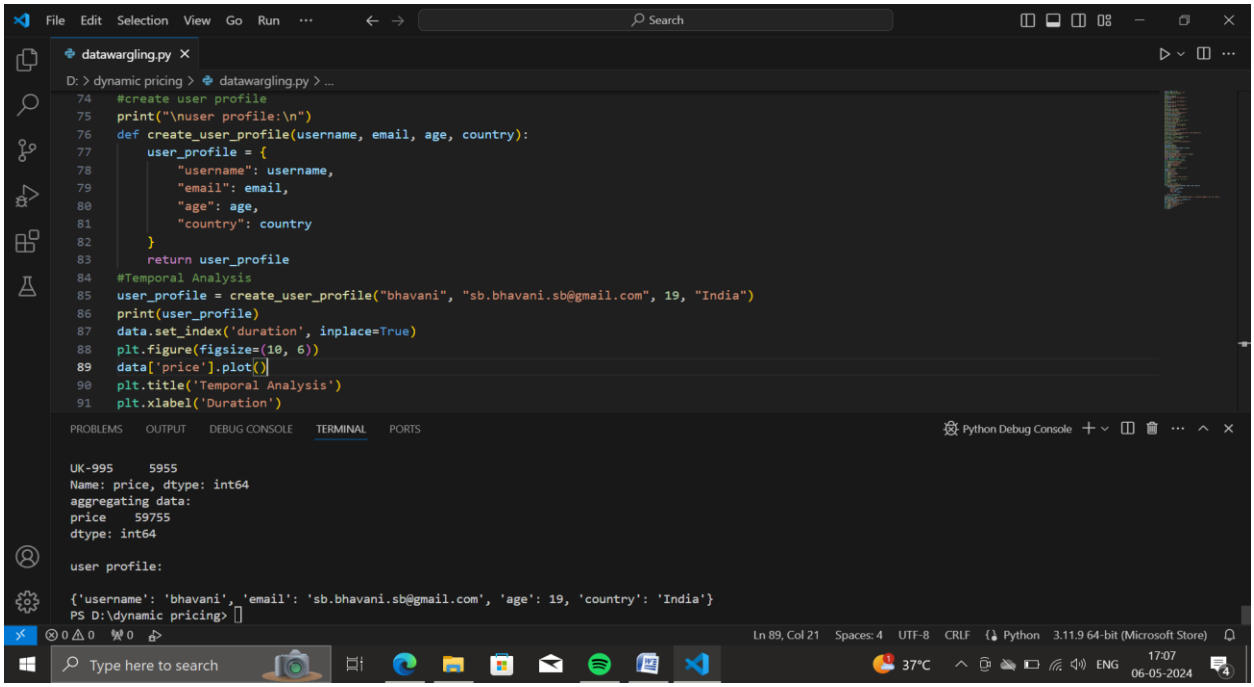

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
Null data identification:
Column1 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
Column1 airline flight source_city departure_time ... destination_city class duration days_left price
0 False False False False False ... False False False False False
1 False False False False False ... False False False False False
2 False False False False False ... False False False False False
3 False False False False False ... False False False False False
4 False False False False False ... False False False False False
5 False False False False False ... False False False False False
6 False False False False False ... False False False False False
7 False False False False False ... False False False False False
8 False False False False False ... False False False False False
9 False False False False False ... False False False False False
[10 rows x 12 columns]
Null Imputation:
Column1 airline flight source_city departure_time ... destination_city class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi Early_Morning ... Mumbai Economy 2.17 1 5956
3 3 Vistara UK-995 Delhi Morning ... Mumbai Economy 2.25 1 5955
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Python Debugger: Python File (dynamic pricing)
Type here to search
30°C
ENG
06:54
06-05-2024

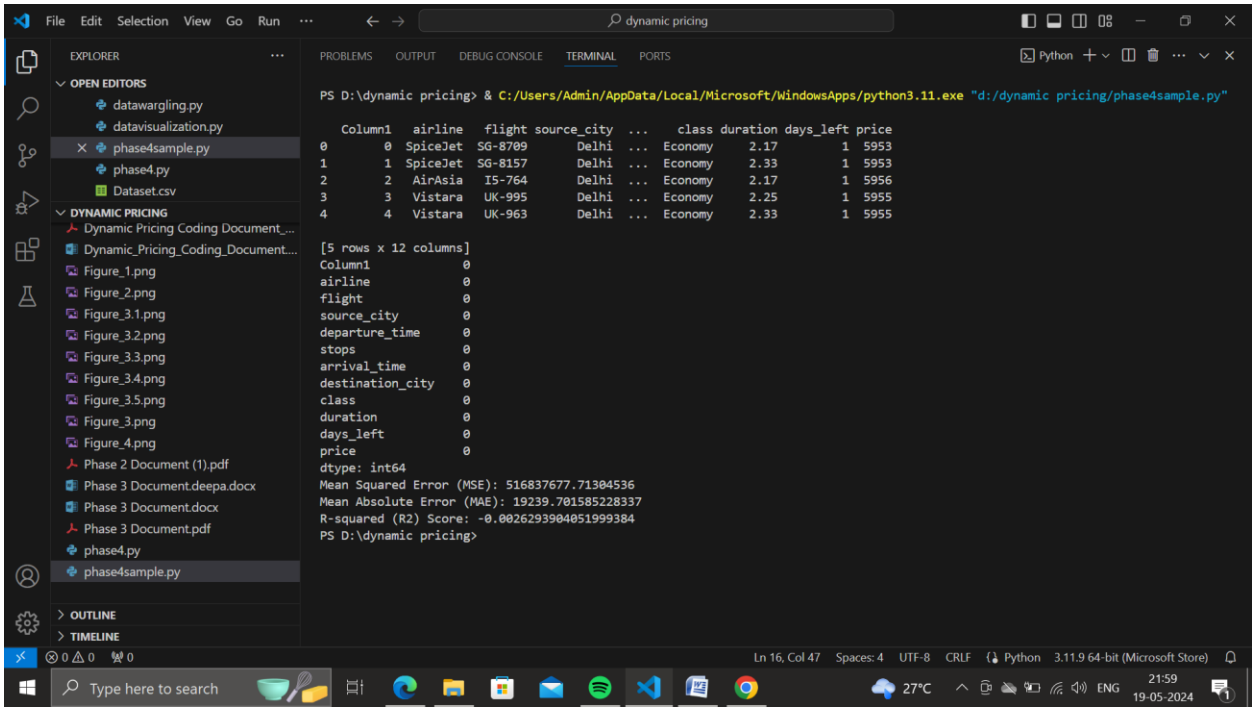

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
Null Imputation:
Column1 airline flight source_city departure_time ... destination_city class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi Early_Morning ... Mumbai Economy 2.17 1 5956
3 3 Vistara UK-995 Delhi Morning ... Mumbai Economy 2.25 1 5955
4 4 Vistara UK-963 Delhi Morning ... Mumbai Economy 2.33 1 5955
5 5 Vistara UK-945 Delhi Morning ... Mumbai Economy 2.33 1 5955
6 6 Vistara UK-927 Delhi Morning ... Mumbai Economy 2.08 1 6060
7 7 Vistara UK-951 Delhi Afternoon ... Mumbai Economy 2.17 1 6060
8 8 GO_FIRST G8-334 Delhi Early_Morning ... Mumbai Economy 2.17 1 5954
9 9 GO_FIRST G8-336 Delhi Afternoon ... Mumbai Economy 2.25 1 5954
[10 rows x 12 columns]
Null Data Removal:
Column1 airline flight source_city departure_time ... destination_city class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi Early_Morning ... Mumbai Economy 2.17 1 5956
3 3 Vistara UK-995 Delhi Morning ... Mumbai Economy 2.25 1 5955
4 4 Vistara UK-963 Delhi Morning ... Mumbai Economy 2.33 1 5955
5 5 Vistara UK-945 Delhi Morning ... Mumbai Economy 2.33 1 5955
6 6 Vistara UK-927 Delhi Morning ... Mumbai Economy 2.08 1 6060
7 7 Vistara UK-951 Delhi Afternoon ... Mumbai Economy 2.17 1 6060
8 8 GO_FIRST G8-334 Delhi Early_Morning ... Mumbai Economy 2.17 1 5954
9 9 GO_FIRST G8-336 Delhi Afternoon ... Mumbai Economy 2.25 1 5954
[10 rows x 12 columns]
datatype of the columns:
Column1 int64
airline object
flight object
source_city object
0 0 Python Debugger: Python File (dynamic pricing)
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Type here to search
30°C
ENG
06:54
06-05-2024

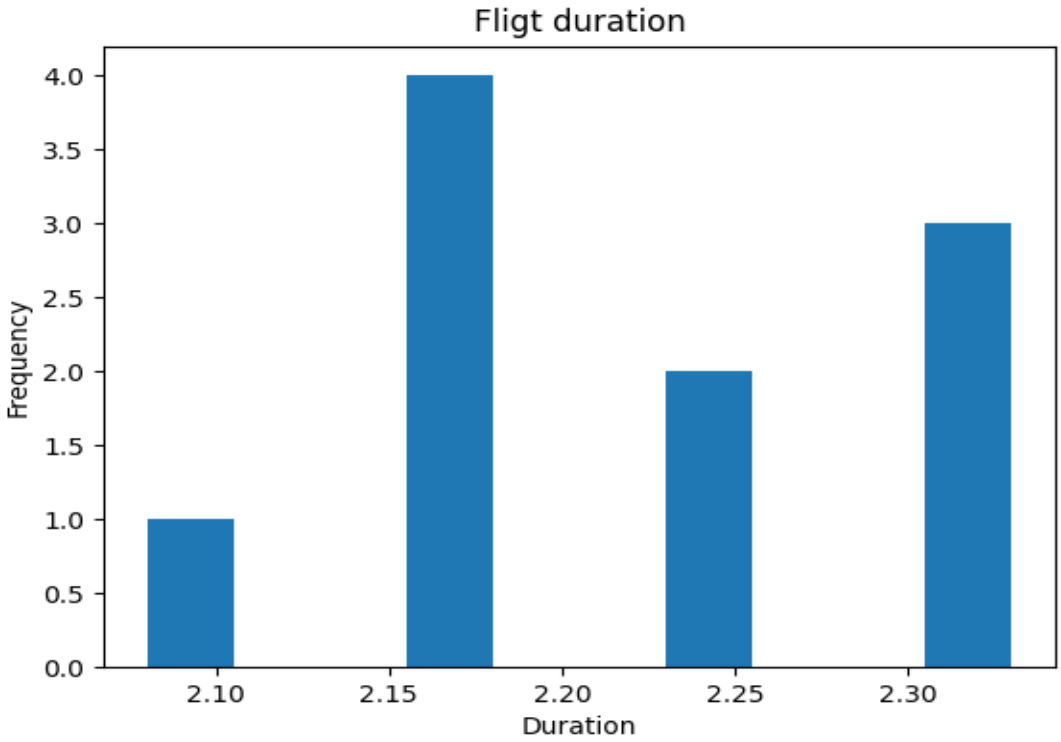

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
datatype of the columns:
Column1 int64
airline object
flight object
source_city object
departure_time object
stops object
arrival_time object
destination_city object
class object
duration float64
days_left int64
price int64
dtype: object
duplicates in the dataset:
0 False
1 False
2 False
3 False
4 False
5 False
6 False
7 False
8 False
9 False
dtype: bool
after removing the duplicates in the dataset:
<bound method DataFrame.drop_duplicates of Column1 airline flight source_city departure_time ... destination_city
class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi Early_Morning ... Mumbai Economy 2.17 1 5956
0 Python Debugger: Python File (dynamic pricing)
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Type here to search
30°C
ENG
06:54
06-05-2024

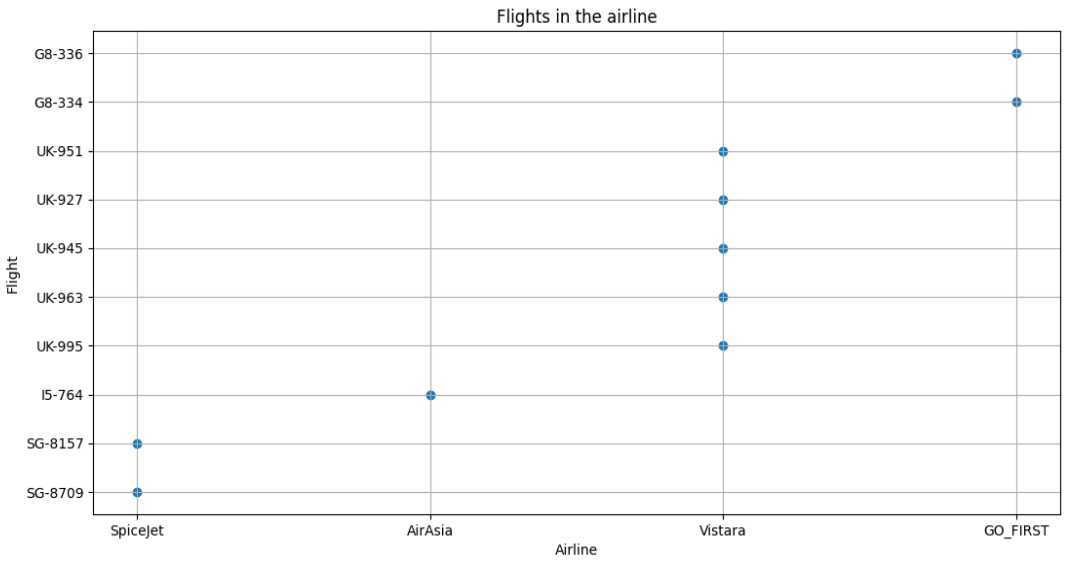

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
after removing the duplicates in the dataset:
<bound method DataFrame.drop_duplicates of Column1 airline flight source_city departure_time ... destination_city
class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi Early_Morning ... Mumbai Economy 2.17 1 5956
3 3 Vistara UK-995 Delhi Morning ... Mumbai Economy 2.25 1 5955
4 4 Vistara UK-963 Delhi Morning ... Mumbai Economy 2.33 1 5955
5 5 Vistara UK-945 Delhi Morning ... Mumbai Economy 2.33 1 5955
6 6 Vistara UK-927 Delhi Morning ... Mumbai Economy 2.08 1 6060
7 7 Vistara UK-951 Delhi Afternoon ... Mumbai Economy 2.17 1 6060
8 8 GO_FIRST G8-334 Delhi Early_Morning ... Mumbai Economy 2.17 1 5954
9 9 GO_FIRST G8-336 Delhi Afternoon ... Mumbai Economy 2.25 1 5954
[10 rows x 12 columns]>
['Economy']
the transpose of the dataframe:
0 1 2 3 ... 6 7 8 9
Column1 0 1 2 3 ... 6 7 8 9
airline SpiceJet SpiceJet AirAsia Vistara ... Vistara Vistara GO_FIRST GO_FIRST
flight SG-8709 SG-8157 I5-764 UK-995 ... UK-927 UK-951 G8-334 G8-336
source_city Delhi Delhi Delhi Delhi ... Delhi Delhi Delhi Delhi
departure_time Evening Early_Morning Early_Morning Morning ... Morning Afternoon Early_Morning Afternoon
stops zero zero zero zero ... zero zero zero zero
arrival_time Night Morning Early_Morning Afternoon ... Morning Evening Morning Evening
destination_city Mumbai Mumbai Mumbai Mumbai ... Mumbai Mumbai Mumbai Mumbai
class Economy Economy Economy Economy ... Economy Economy Economy Economy
duration 2.17 2.33 2.17 2.25 ... 2.08 2.17 2.17 2.25
days_left 1 1 1 1 ... 1 1 1 1
price 5953 5953 5956 5955 ... 6060 6060 5954 5954
[12 rows x 10 columns]
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Python Debugger: Python File (dynamic pricing)
Type here to search
30°C
ENG
06:54
06-05-2024

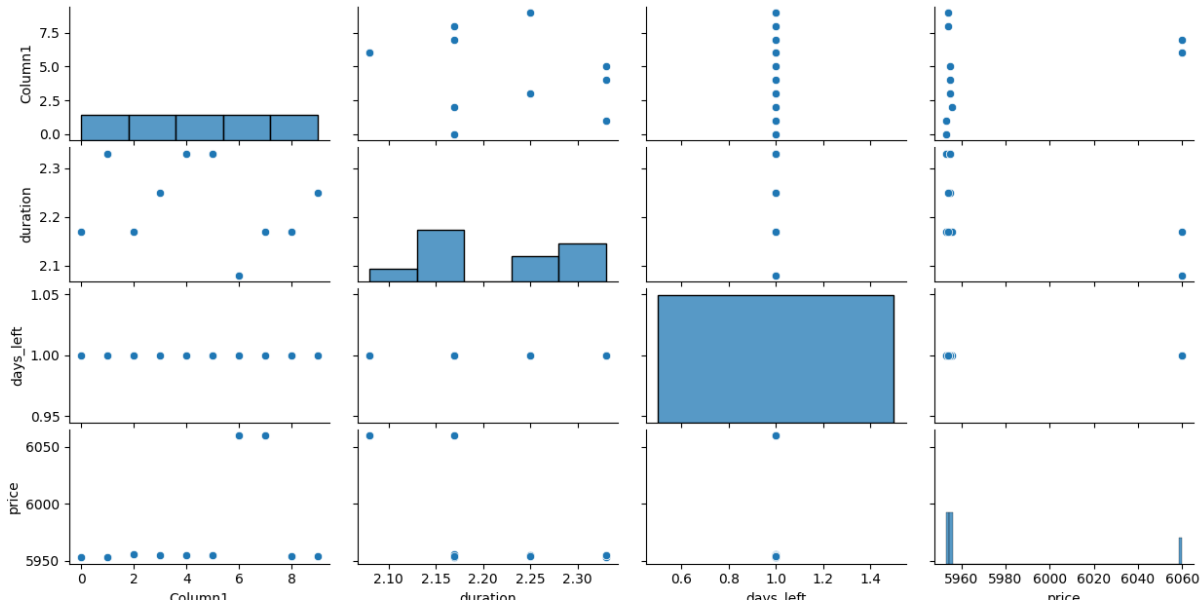

File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
Dataset.csv
DYNAMIC PRICING
data.py
Dataset.csv
datawargling.py
Figure_1.png
Figure_2.png
Figure_3.png
Figure_4.png
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE
Python Debug Console
arrival_time Night Morning Early_Morning Afternoon ... Morning Evening Morning Evening
destination_city Mumbai Mumbai Mumbai Mumbai ... Mumbai Mumbai Mumbai Mumbai
class Economy Economy Economy Economy ... Economy Economy Economy Economy
duration 2.17 2.33 2.17 2.25 ... 2.08 2.17 2.17 2.25
days_left 1 1 1 1 ... 1 1 1 1
price 5953 5953 5956 5955 ... 6060 6060 5954 5954
[12 rows x 10 columns]
Column1 airline flight source_city departure_time ... destination_city class duration days_left price
0 0 SpiceJet SG-8709 Delhi Evening ... Mumbai Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi Early_Morning ... Mumbai Economy 2.33 1 5953
8 8 GO_FIRST G8-334 Delhi Early_Morning ... Mumbai Economy 2.17 1 5954
9 9 GO_FIRST G8-336 Delhi Afternoon ... Mumbai Economy 2.25 1 5954
[4 rows x 12 columns]
data aggregation:
grouping data:
flight
G8-334 5954
G8-336 5954
I5-764 5956
SG-8157 5953
SG-8709 5953
UK-927 6060
UK-945 5955
UK-951 6060
UK-963 5955
UK-995 5955
Name: price, dtype: int64
aggregating data:
price 59755
dtype: int64
Ln 22, Col 26 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Python Debugger: Python File (dynamic pricing)
Type here to search
30°C
ENG
06:55
06-05-2024

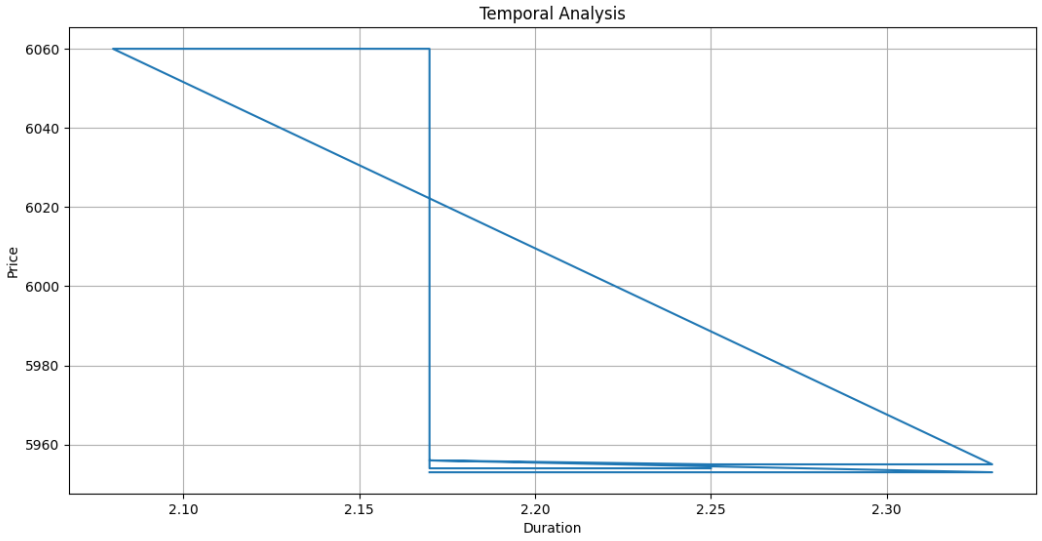

File Edit Selection View Go Run
Search
datawargling.py
D: > dynamic pricing > datawargling.py > ...
#create user profile
print("\nuser profile:\n")
def create_user_profile(username, email, age, country):
user_profile = {
"username": username,
"email": email,
"age": age,
"country": country
}
return user_profile
#Temporal Analysis
user_profile = create_user_profile("bhavani", "sb.bhavani.sb@gmail.com", 19, "India")
print(user_profile)
data.set_index('duration', inplace=True)
plt.figure(figsize=(10, 6))
data['price'].plot()
plt.title('Temporal Analysis')
plt.xlabel('Duration')
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python Debug Console
UK-995 5955
Name: price, dtype: int64
aggregating data:
price 59755
dtype: int64
user profile:
{'username': 'bhavani', 'email': 'sb.bhavani.sb@gmail.com', 'age': 19, 'country': 'India'}
PS D:\dynamic pricing>
Ln 89, Col 21 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Type here to search
37°C
ENG
17:07
06-05-2024


File Edit Selection View Go Run
dynamic pricing
EXPLORER
OPEN EDITORS
datawargling.py
datavisualization.py
phase4sample.py
phase4.py
Dataset.csv
DYNAMIC PRICING
Dynamic Pricing Coding Document_...
Dynamic_Pricing_Coding_Document....
Figure_1.png
Figure_2.png
Figure_3.1.png
Figure_3.2.png
Figure_3.3.png
Figure_3.4.png
Figure_3.5.png
Figure_3.png
Figure_4.png
Phase 2 Document (1).pdf
Phase 3 Document.deepa.docx
Phase 3 Document.docx
Phase 3 Document.pdf
phase4.py
phase4sample.py
OUTLINE
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
Python
PS D:\dynamic pricing> & C:/Users/Admin/AppData/Local/Microsoft/WindowsApps/python3.11.exe "d:/dynamic pricing/phase4sample.py"
Column1 airline flight source_city ... class duration days_left price
0 0 SpiceJet SG-8709 Delhi ... Economy 2.17 1 5953
1 1 SpiceJet SG-8157 Delhi ... Economy 2.33 1 5953
2 2 AirAsia I5-764 Delhi ... Economy 2.17 1 5956
3 3 Vistara UK-995 Delhi ... Economy 2.25 1 5955
4 4 Vistara UK-963 Delhi ... Economy 2.33 1 5955
[5 rows x 12 columns]
Column1 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
Mean Squared Error (MSE): 516837677.71304536
Mean Absolute Error (MAE): 19239.701585228337
R-squared (R2) Score: -0.0026293904051999384
PS D:\dynamic pricing>
Ln 16, Col 47 Spaces: 4 UTF-8 CRLF Python 3.11.9 64-bit (Microsoft Store)
Type here to search
27°C
ENG
21:59
19-05-2024

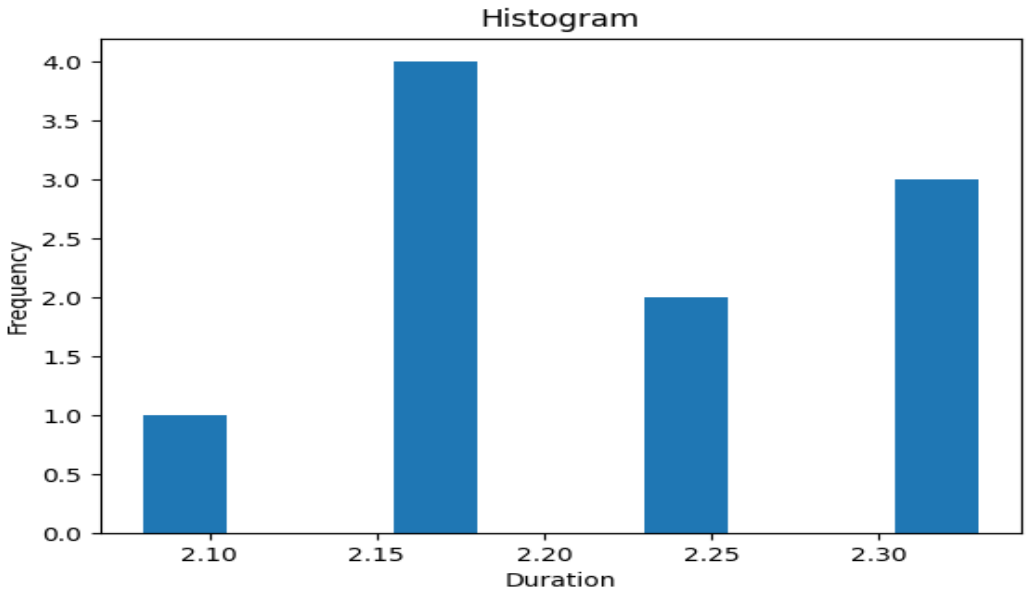

Fligt duration
Frequency
Duration
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2.10
2.15
2.20
2.25
2.30


Flights in the airline
Flight
Airline
G8-336
G8-334
UK-951
UK-927
UK-945
UK-963
UK-995
I5-764
SG-8157
SG-8709
SpiceJet
AirAsia
Vistara
GO_FIRST

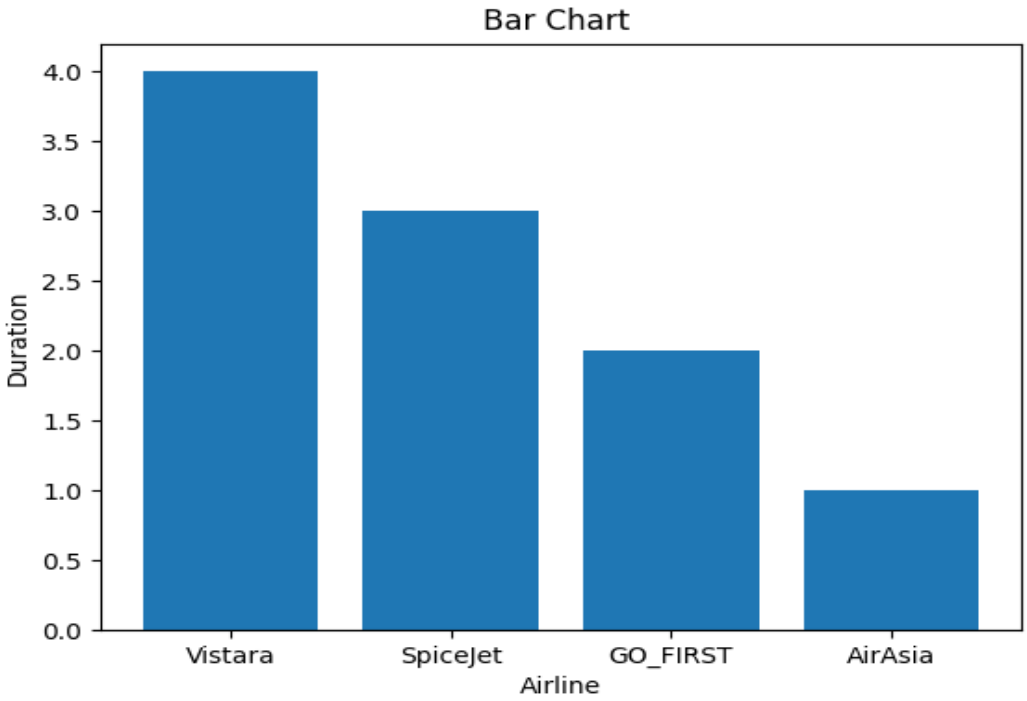

Column1
duration
days_left
price
Column1
duration
days_left
price


Temporal Analysis
Price
Duration

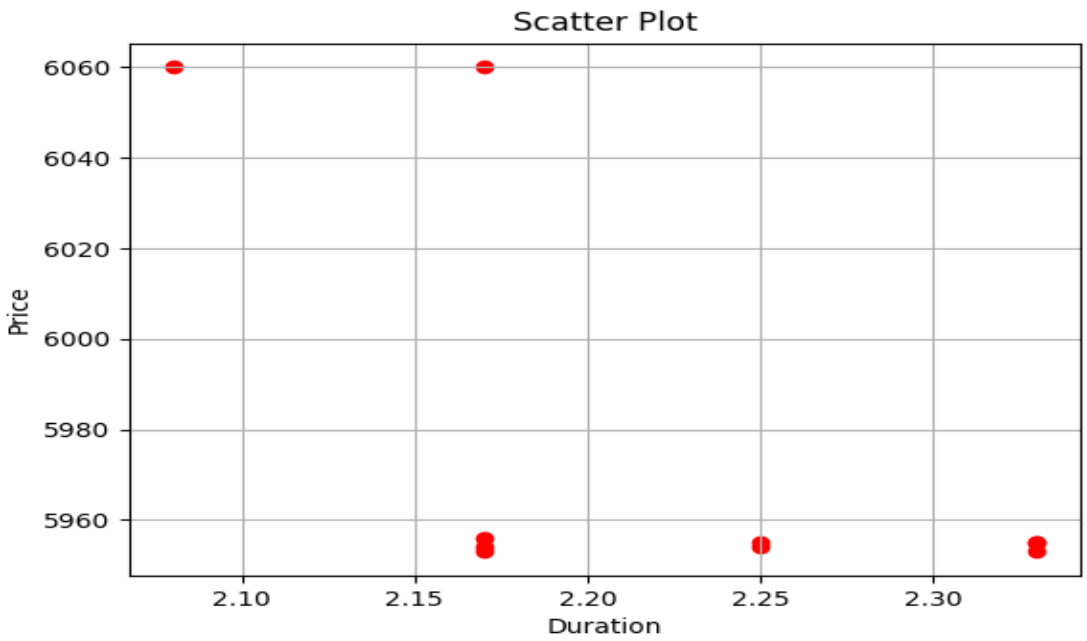
#univariate analysis - Histogram



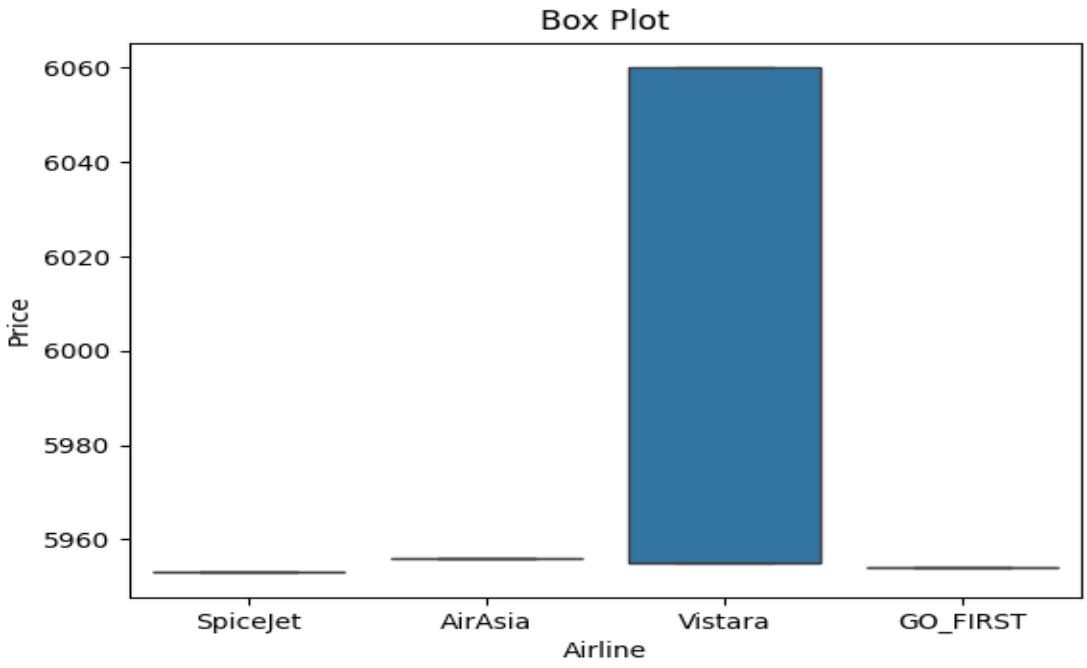
#univariate analysis – Bar chart



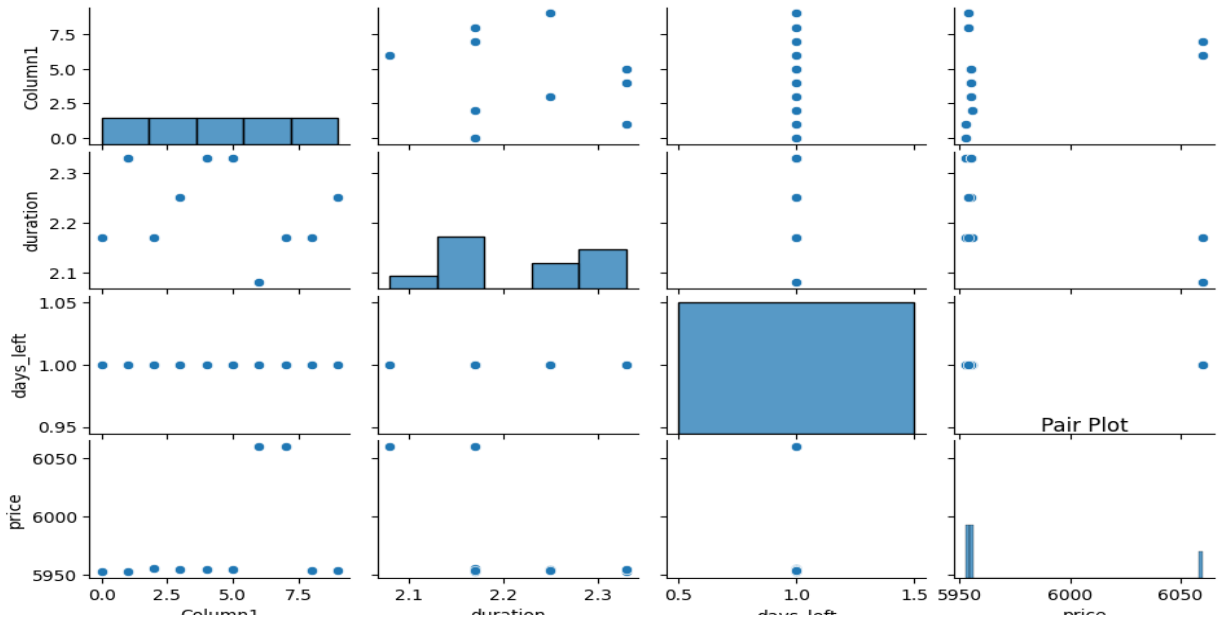
#Biunivariate analysis - Scatter plot



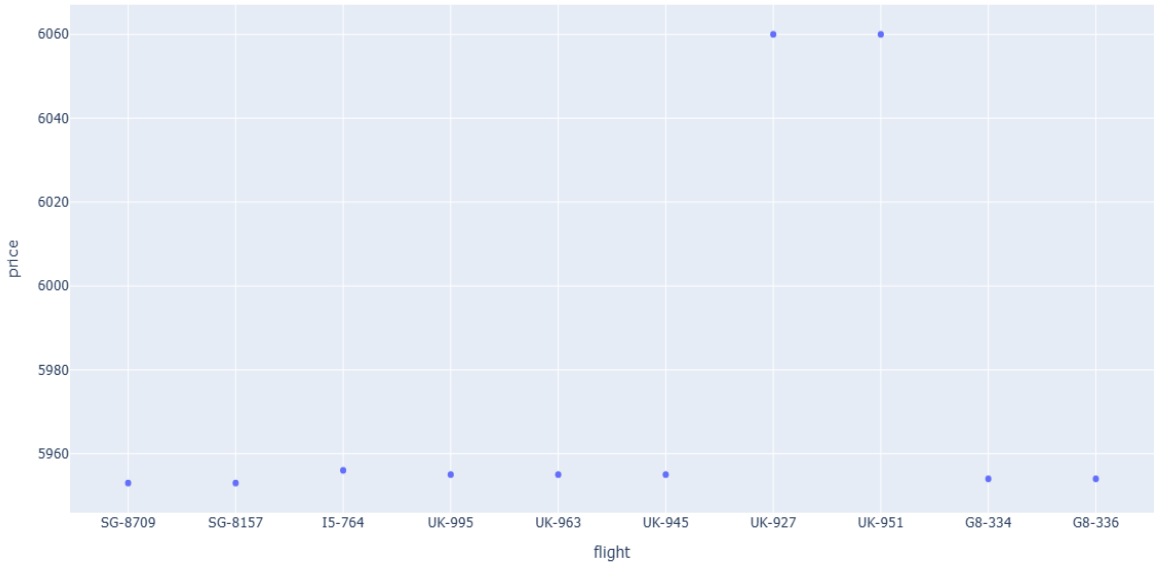
#Biunivariate analysis - Box plot



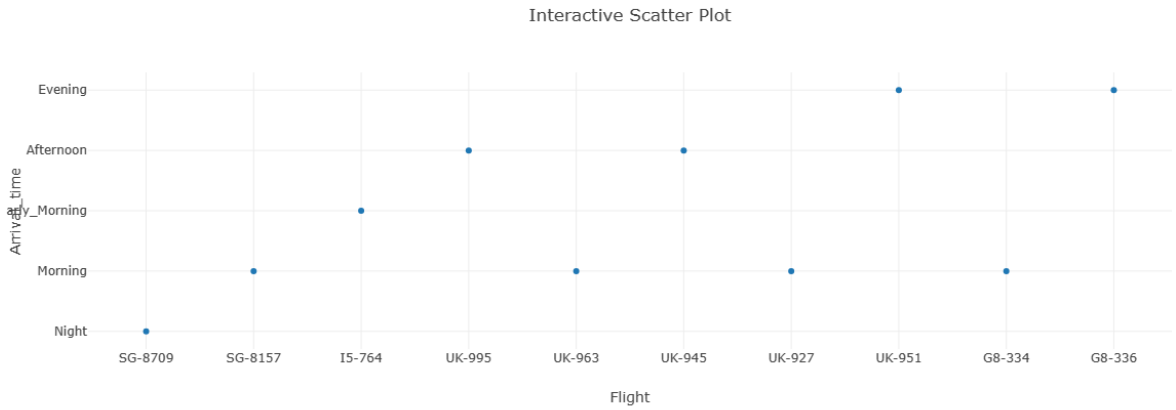
#Multivariate visualization – Pair plot



#Interactive visualization – Scatter plot



#Interactive visualization – Dashboard



Future work:

Here are 3 future directions for AI model that analyze market conditions, consumer behavior, and competitor pricing in real-time for dynamic pricing:

1. **Advanced Customer Targeting:** We can develop deeper into customer segmentation by incorporating social media behavior and demographics to tailor pricing strategies even more effectively.
2. **Real-Time Learning with Reinforcement Learning:** Exploring reinforcement learning models could allow the pricing strategy to continuously adapt to customer response and market changes in realtime.
3. **Explainable AI for Fairness and Trust:** Implementing Explainable AI techniques can build trust by not explaining price recommendations but also demonstrating the fairness considerations used in the model.

Future Enchancements:

Advanced Feature Engineering: Explore techniques like dimensionality reduction (e.g., Principal Component Analysis) to handle high-dimensional data and potentially extract more informative features.

Deep Learning Models: Investigate the use of recurrent neural networks (RNNs) or convolutional neural networks (CNNs) to capture temporal patterns and complex relationships within transaction sequences, especially if your data exhibits such characteristics.

Conclusion:

In conclusion, our project on dynamic pricing leverages robust data collection, insightful data visualization, and sophisticated data modeling to optimize pricing strategies. By employing the Random Forest Regression model, we enhance prediction accuracy and adaptability in fluctuating markets. The integration of these advanced techniques enables a comprehensive analysis of pricing dynamics, leading to more informed and strategic decision-making. Our approach not only improves revenue management but also fosters competitive advantage in a dynamic economic landscape.