# OPTIMIZING FLIGHT BOOKING DECISIONS THROUGH MACHINE LEARNING PRICE PREDICTIONS

## 1 INTRODUCTION

### 1.1 OVERVIEW:

**Optimizing flight booking decisions through machine learning price prediction involves leveraging machine learning algorithms to analyze historical data and predict future flight prices. By accurately predicting prices, travelers can make informed decisions about when to book their flights, potentially saving money in the process.**

**Machine learning algorithms can take into account a wide range of factors that may influence flight prices, such as the time of year, day of the week, airline, and departure and arrival cities. By analyzing large amounts of historical data, these algorithms can identify patterns and trends that may not be apparent to humans.**

### 1.2 PURPOSE:

**The purpose of an optimizing flight booking decisions through machine learning price prediction project would be to develop a system that can accurately predict the prices of flights in the future based on historical data and other relevant factors. By doing so, this system could help individuals and organizations make more informed decisions about when to book their flights in order to save money and improve their overall travel experience.**

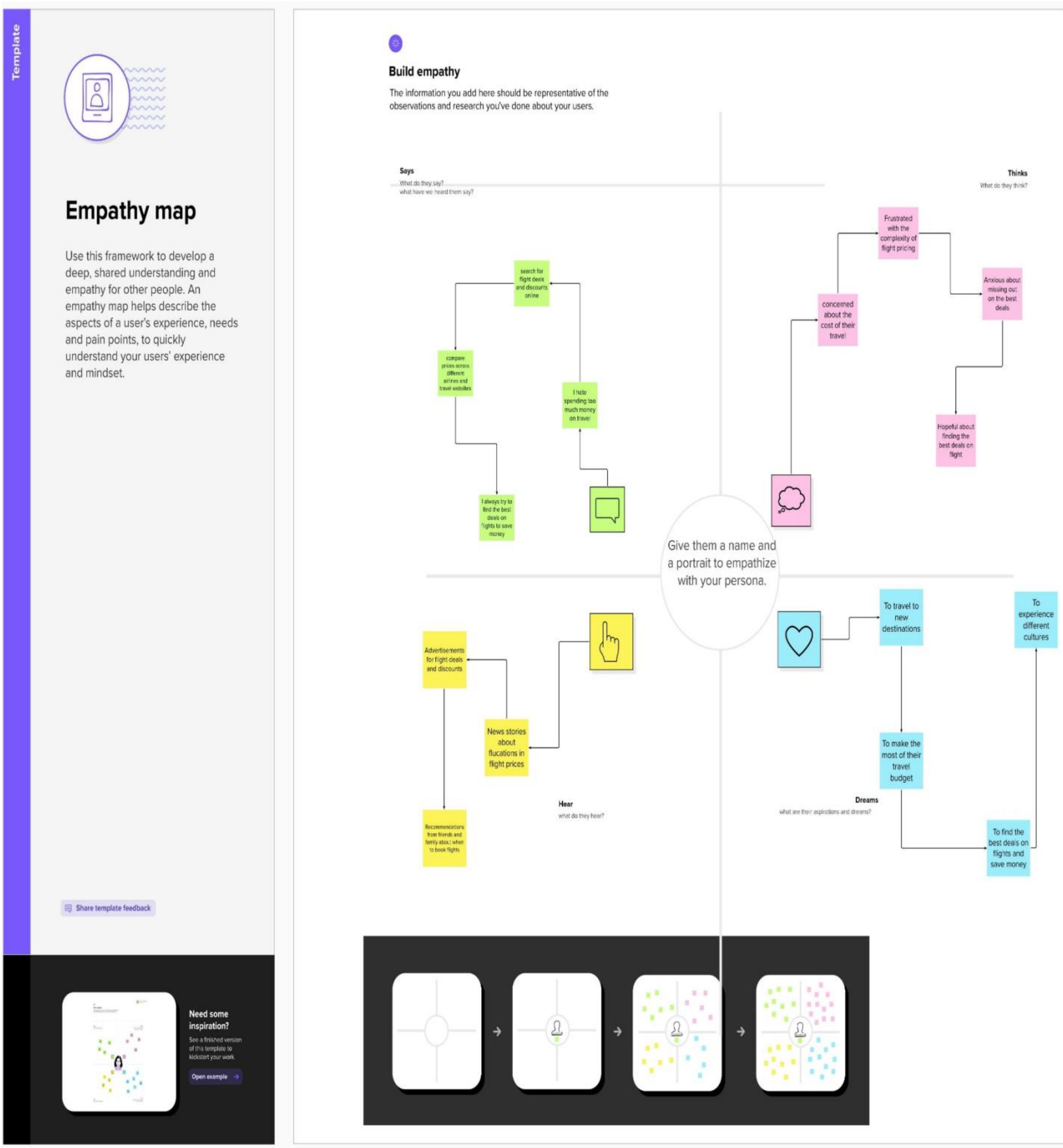
- **Cost savings: By predicting the best time to book a flight, individuals and organizations could potentially save significant amounts of money on travel expenses.**

- **Increased convenience: By having access to accurate price predictions, individuals and organizations could better plan their travel itineraries and make more informed decisions about their travel plans.**

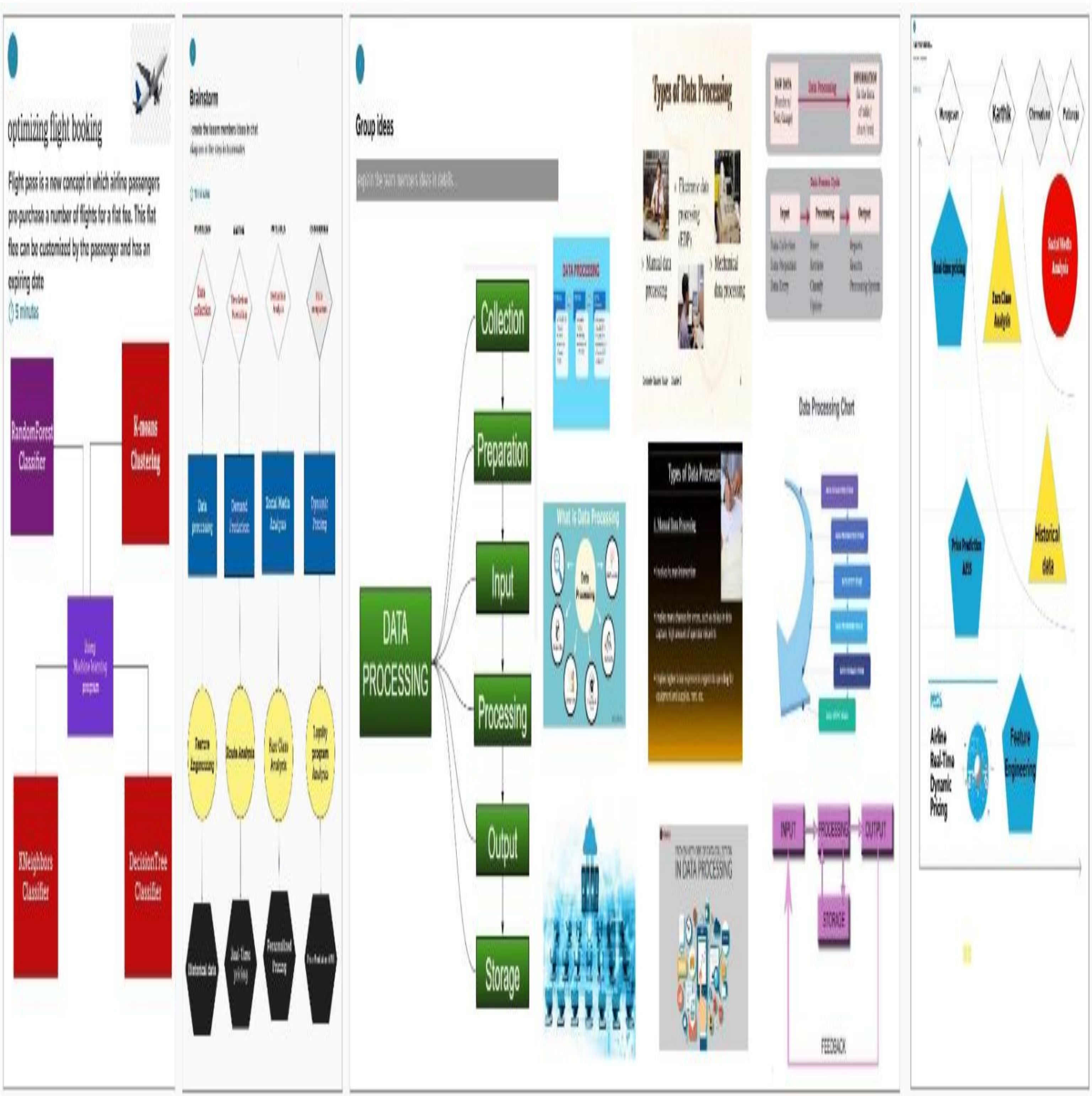
- **Improved customer satisfaction: By reducing the stress and uncertainty associated with booking flights, individuals and organizations could have a more positive overall travel experience.**

## 2 PROBLEM DEFINITION & DESIGN THINKING

### 2.1 EMPATHY MAP:



## 2.2 IDEATION & BRAINSTORMING MAP:



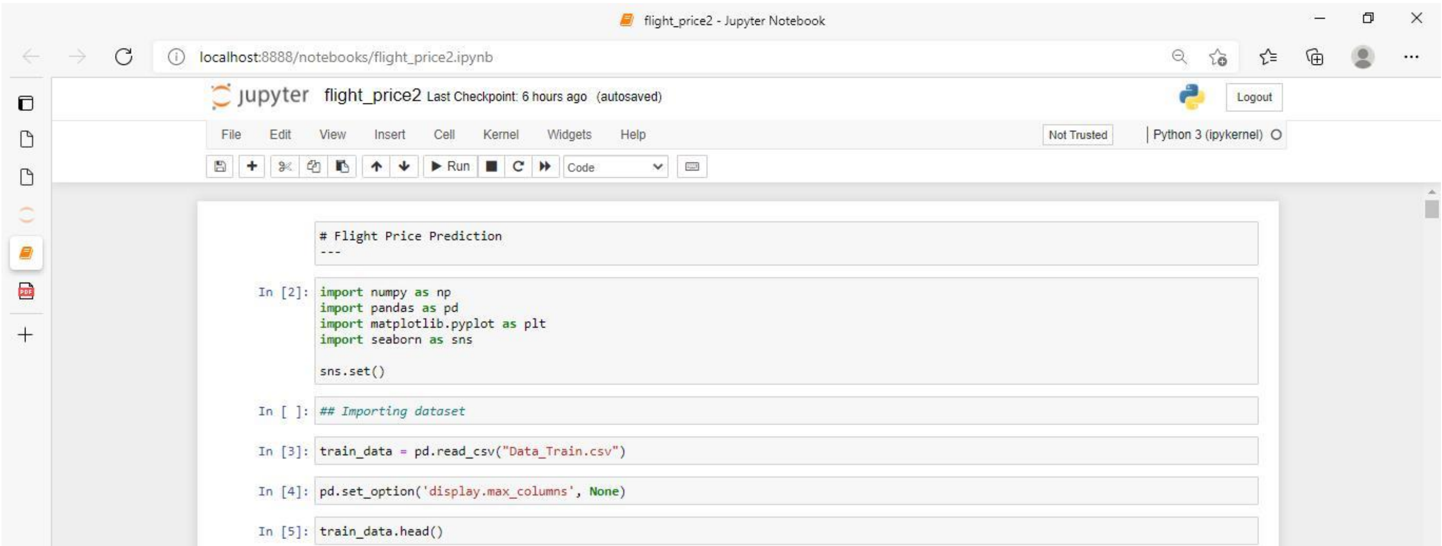
# 3 RESULT

## 3.1 DATA MODEL:

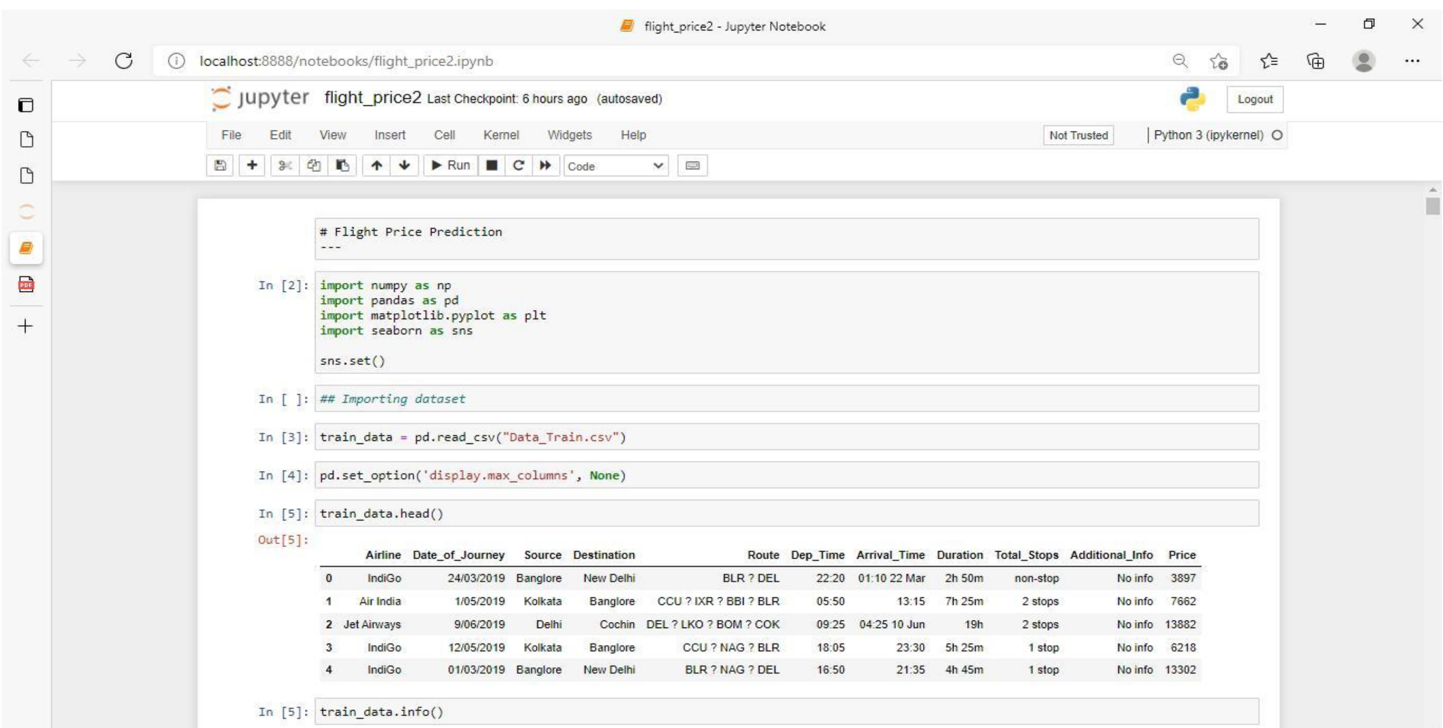
<table>
<tr><th>OBJECT NAME</th><th>FIELDS IN THE OBJECT</th></tr>
<tr><td>CUSTOMER</td><td>
<table>
<tr><th>FIELD LABEL</th><th>DATA TYPE</th></tr>
<tr><td>First Name</td><td>Text</td></tr>
<tr><td>Last Name</td><td>Text</td></tr>
<tr><td>Email</td><td>Text</td></tr>
<tr><td>Phone No</td><td>Number</td></tr>
<tr><td>Address</td><td>Text</td></tr>
</table>
</td></tr>
<tr><td>FLIGHT</td><td>
<table>
<tr><th>FIELD LABEL</th><th>DATA TYPE</th></tr>
<tr><td>Airline</td><td>Text</td></tr>
<tr><td>Flight</td><td>Number</td></tr>
<tr><td>Departure Airport</td><td>Text</td></tr>
<tr><td>Arrival Airport</td><td>Text</td></tr>
<tr><td>Arrival Time</td><td>Date & Time</td></tr>
<tr><td>Price</td><td>Currancy</td></tr>
<tr><td>Numper of Seats</td><td>Number</td></tr>
</table>
</td></tr>
</table>

## 3.2 ACTIVITY & SCREENSHOT:

- **Importing the libraries**




- **Read the dataset:**



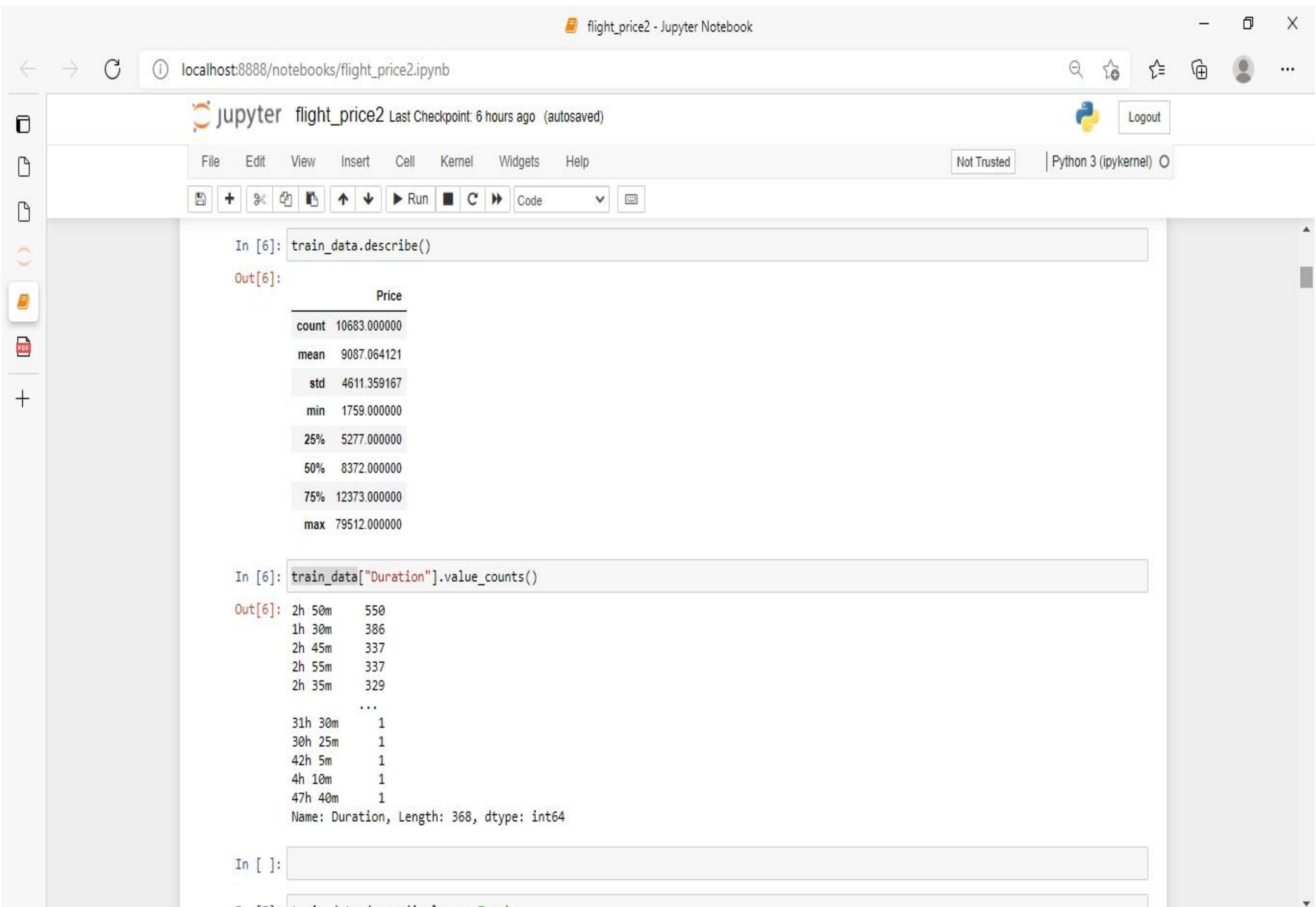

**In pandas we have a function called read_csv() to read the dataset. As a parameter we**

**Have to give the directory of csv file.**

## ➢ Data Preparation:

**As we have understood how the data is let's pre-process the collected data.**

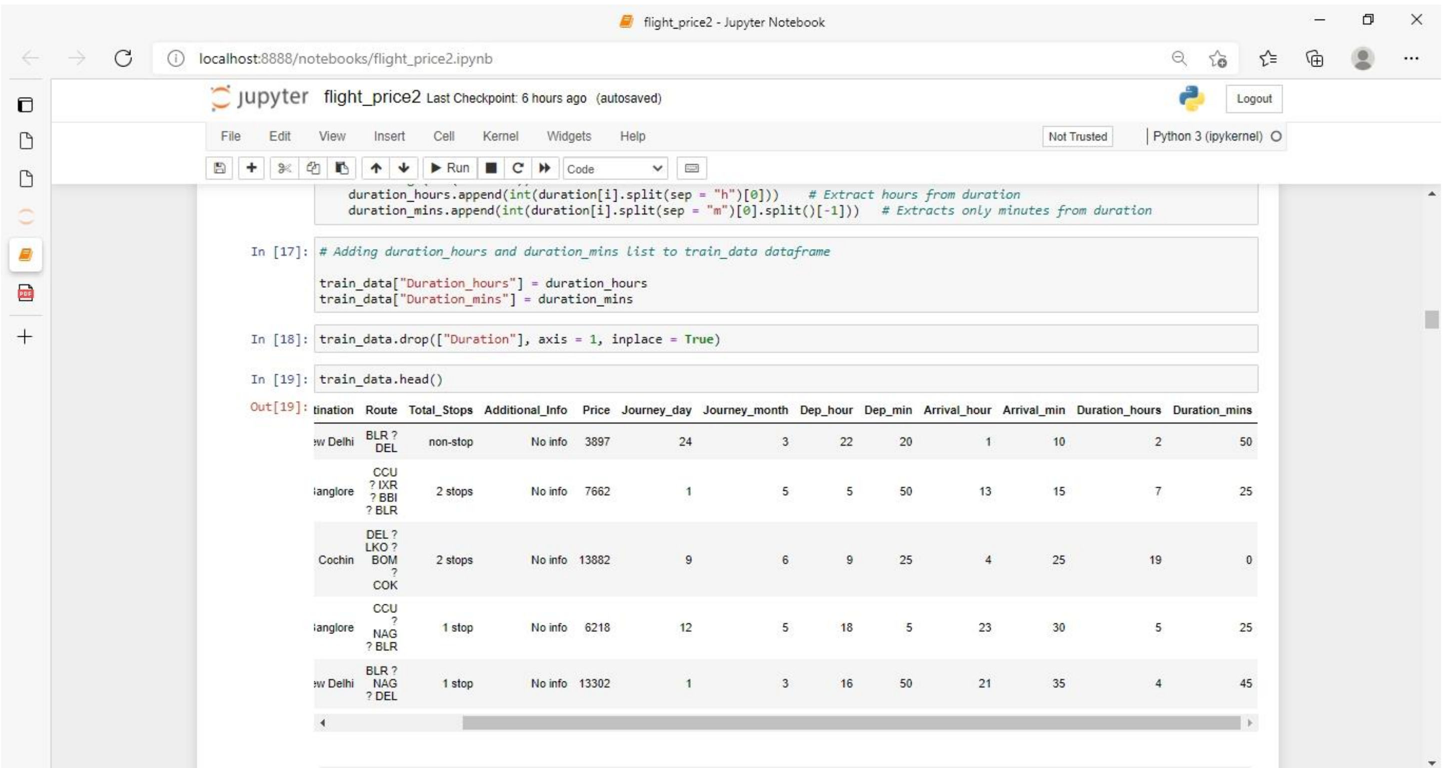
- **Handling missing values**
- **Handling categorical data**
- **Handling outliers**
- **Scaling Techniques**
- **Splitting dataset into training and test set**



## ➢ Label Encodings:

**Label encoding converts the data in machine readable form, but it assigns a unique number (starting from**

**0) To each class of data.'Airline','Source','Destination','Total_Stops','City1','City2','City3','Additional_Info' into number format.**




```
    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
```

In [17]:
```
# Adding duration_hours and duration_mins list to train_data dataframe

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins
```

In [18]:
```
train_data.drop(["Duration"], axis = 1, inplace = True)
```

In [19]:
```
train_data.head()
```

Out[19]:

| tination | Route | Total_Stops | Additional_Info | Price | Journey_day | Journey_month | Dep_hour | Dep_min | Arrival_hour | Arrival_min | Duration_hours | Duration_mins |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| ew Delhi | BLR ? DEL | non-stop | No info | 3897 | 24 | 3 | 22 | 20 | 1 | 10 | 2 | 50 |
| anglore | CCU ? IXR ? BBI ? BLR | 2 stops | No info | 7662 | 1 | 5 | 5 | 50 | 13 | 15 | 7 | 25 |
| Cochin | DEL ? LKO ? BOM ? COK | 2 stops | No info | 13882 | 9 | 6 | 9 | 25 | 4 | 25 | 19 | 0 |
| anglore | CCU ? NAG ? BLR | 1 stop | No info | 6218 | 12 | 5 | 18 | 5 | 23 | 30 | 5 | 25 |
| ew Delhi | BLR ? NAG ? DEL | 1 stop | No info | 13302 | 1 | 3 | 16 | 50 | 21 | 35 | 4 | 45 |

## ➢ Replacing Missing Value:

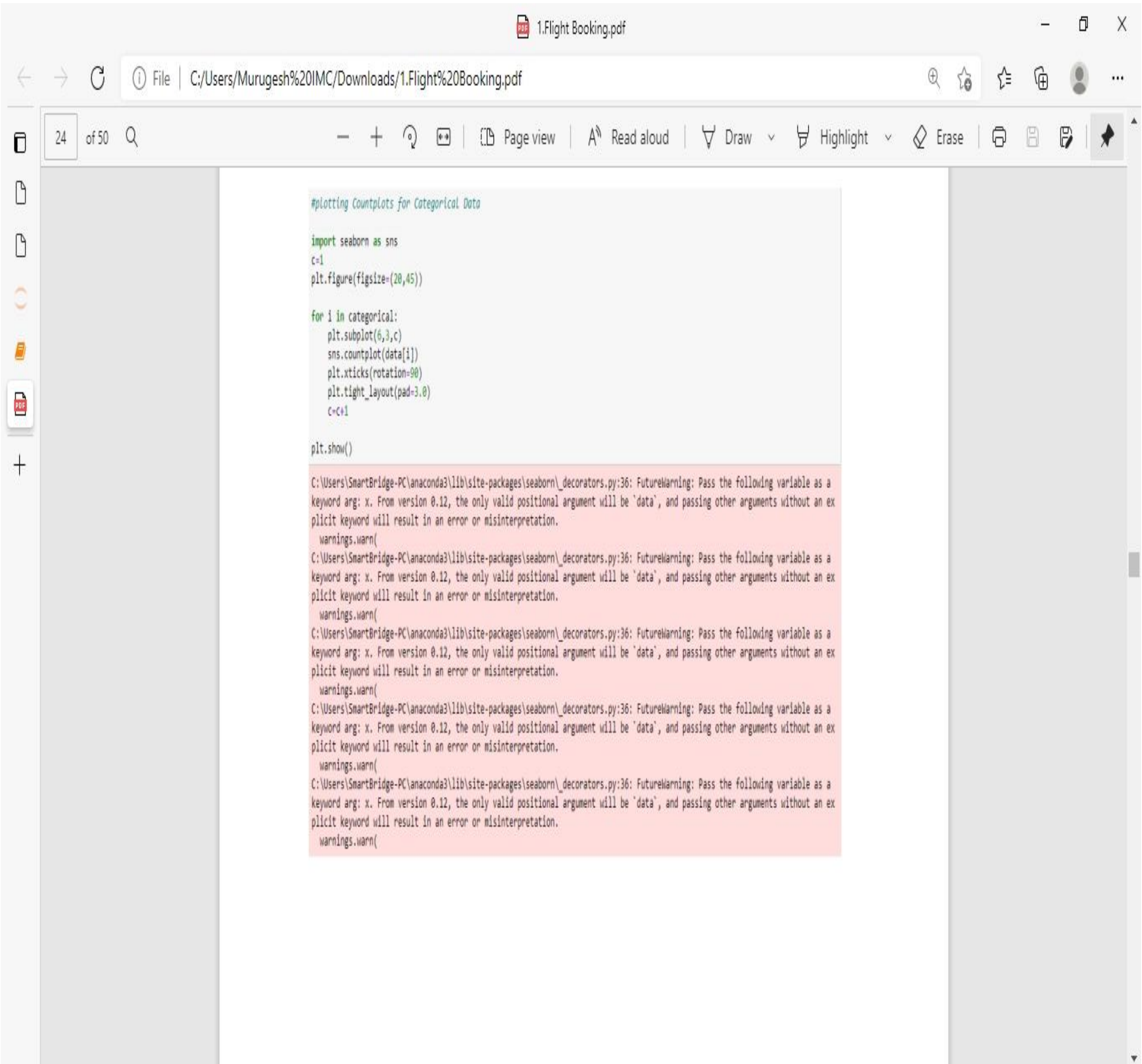
```
#filling City3 as None, the missing values are less
data['City3'].fillna('None',inplace=True)
```

```
#filling Arrival_Date as Departure_Date
data['Arrival_date'].fillna(data['Date'],inplace=True)
```

```
#filling Travel_Mins as Zero(0)
data['Travel_Mins'].fillna(0,inplace=True)
```

## ➢ Categorical Data:

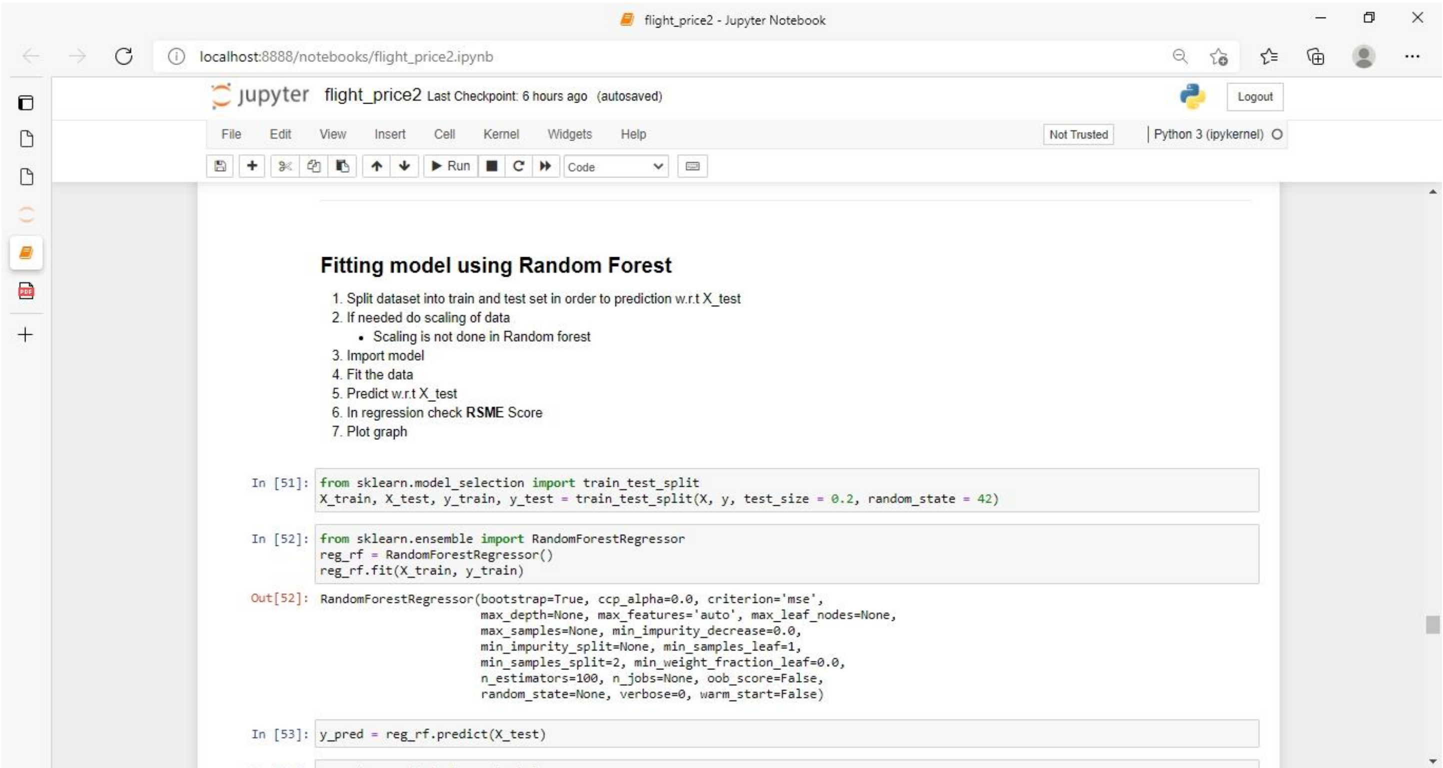
**Plotting countplots for categorical data**



```
#plotting Countplots for Categorical Data

import seaborn as sns
c=1
plt.figure(figsize=(20,45))

for i in categorical:
    plt.subplot(6,3,c)
    sns.countplot(data[i])
    plt.xticks(rotation=90)
    plt.tight_layout(pad=3.0)
    c=c+1

plt.show()
```

```
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
C:\Users\SmartBridge-PC\anaconda3\lib\site-packages\seaborn\_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.
  warnings.warn(
```

## ➤ RandomForestRegressor:




## ➤ Scaling the Data

```python
y = data['Price']
x = data.drop(columns=['Price'],axis=1)
```

```python
### Scaling the Data
```

```python
from sklearn.preprocessing import StandardScaler
ss=StandardScaler()
```

```python
x_scaled = ss.fit_transform(x)
```

```python
x_scaled = pd.DataFrame(x_scaled,columns=x.columns)
x_scaled.head()
```

| | Airline | Source | Destination | Date | Month | Year | Dep_Time_Hour | Dep_Time_Mins | Arrival_date | Arrival_Time_Hour | Arrival_Time_Mins |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.410934 | -1.658354 | 2.416648 | 1.237192 | -1.467619 | 0.0 | 1.654162 | -0.234832 | 0.955658 | -1.800328 | -0.889941 |
| 1 | -1.261305 | 0.890262 | -0.973718 | -1.475375 | 0.250165 | 0.0 | -1.303018 | 1.363790 | -1.524701 | -0.050871 | -0.586988 |
| 2 | 0.014251 | 0.040723 | -0.295645 | -0.531874 | 1.109057 | 0.0 | -0.607211 | 0.031605 | -0.461690 | -1.362964 | 0.018919 |
| 3 | -0.410934 | 0.890262 | -0.973718 | -0.178060 | 0.250165 | 0.0 | 0.958355 | -1.034142 | -0.225465 | 1.407010 | 0.321872 |
| 4 | -0.410934 | -1.658354 | 2.416648 | -1.475375 | -1.467619 | 0.0 | 0.610452 | 1.363790 | -1.524701 | 1.115434 | 0.624825 |

## ➢ Regression Model

```python
from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2_score,mean_absolute_error,mean_squared_error

knn=KNeighborsRegressor()
svr=SVR()
dt=DecisionTreeRegressor()

for i in [knn,svr,dt]:
    i.fit(x_train,y_train)
    y_pred=i.predict(x_test)
    test_score=r2_score(y_test,y_pred)
    train_score=r2_score(y_train,i.predict(x_train))
    if abs(train_score-test_score)<=0.1:
        print(i)
        print('R2 Score is',r2_score(y_test,y_pred))
        print('R2 Score for train data',r2_score(y_train,i.predict(x_train)))
        print('Mean Absolute Error is',mean_absolute_error(y_test,y_pred))
        print('Mean Squared Error is',mean_squared_error(y_test,y_pred))
        print('Root Mean Squared Error is',(mean_squared_error(y_test,y_pred,squared=False)))
```

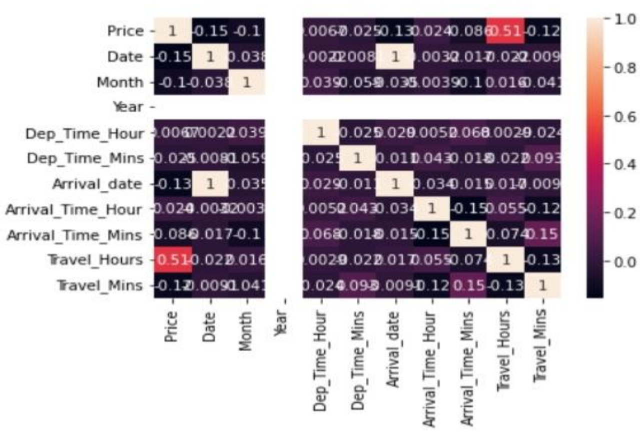
```
KNeighborsRegressor()
R2 Score is 0.7354576039734038
R2 Score for train data 0.7910150823510993
Mean Absolute Error is 1635.3106223678053
Mean Squared Error is 5584955.836743098
Root Mean Squared Error is 2363.2511158874117
SVR()
R2 Score is -0.007934481035057894
R2 Score for train data -0.012381130959185693
Mean Absolute Error is 3631.923243955232
Mean Squared Error is 21279271.857602067
Root Mean Squared Error is 4612.94611475162
```

## ➢ Data Analysts:

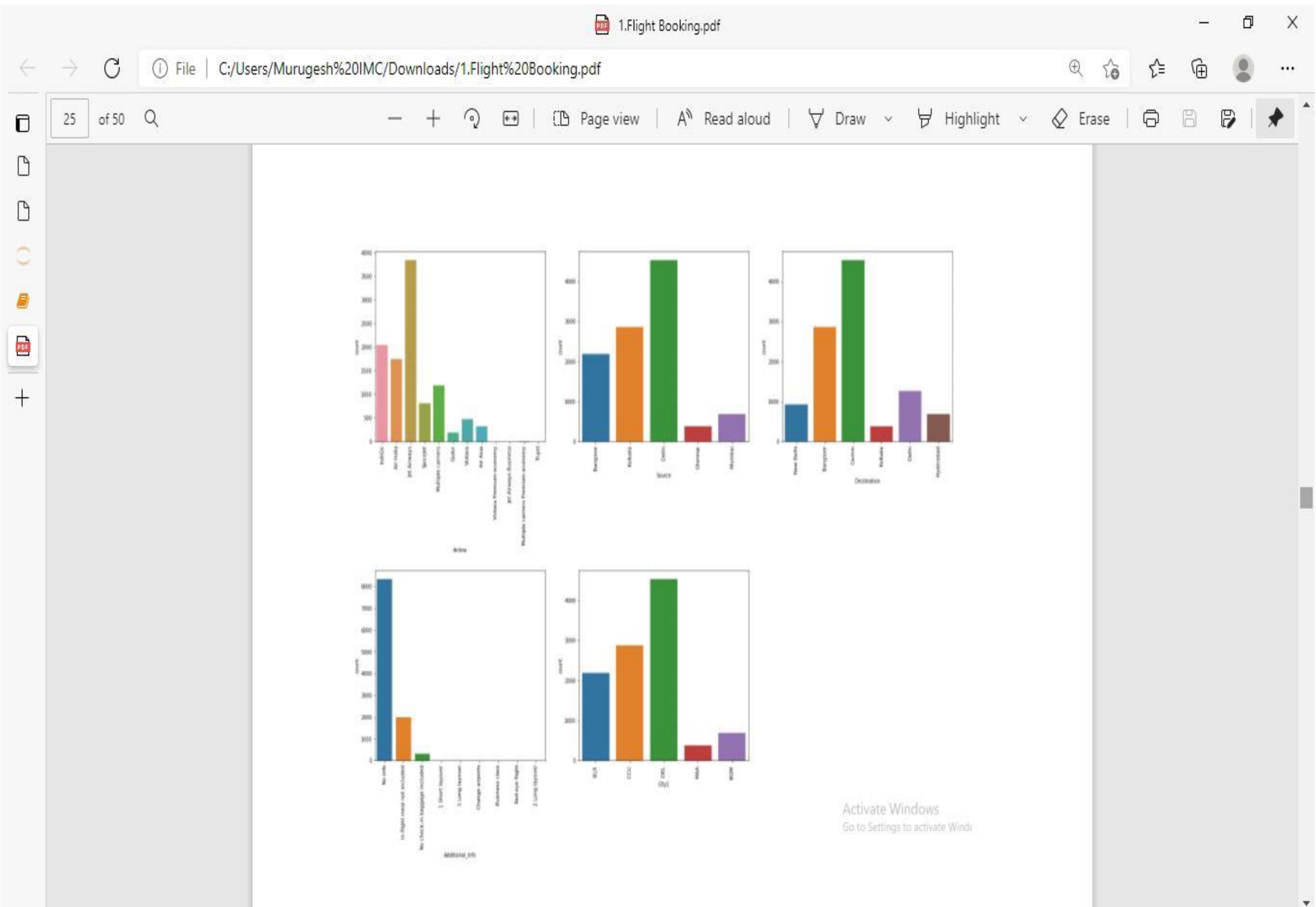
```
data.describe()
```

| | Price |
|---|---|
| count | 10683.000000 |
| mean | 9087.064121 |
| std | 4611.359167 |
| min | 1759.000000 |
| 25% | 5277.000000 |
| 50% | 8372.000000 |
| 75% | 12373.000000 |
| max | 79512.000000 |

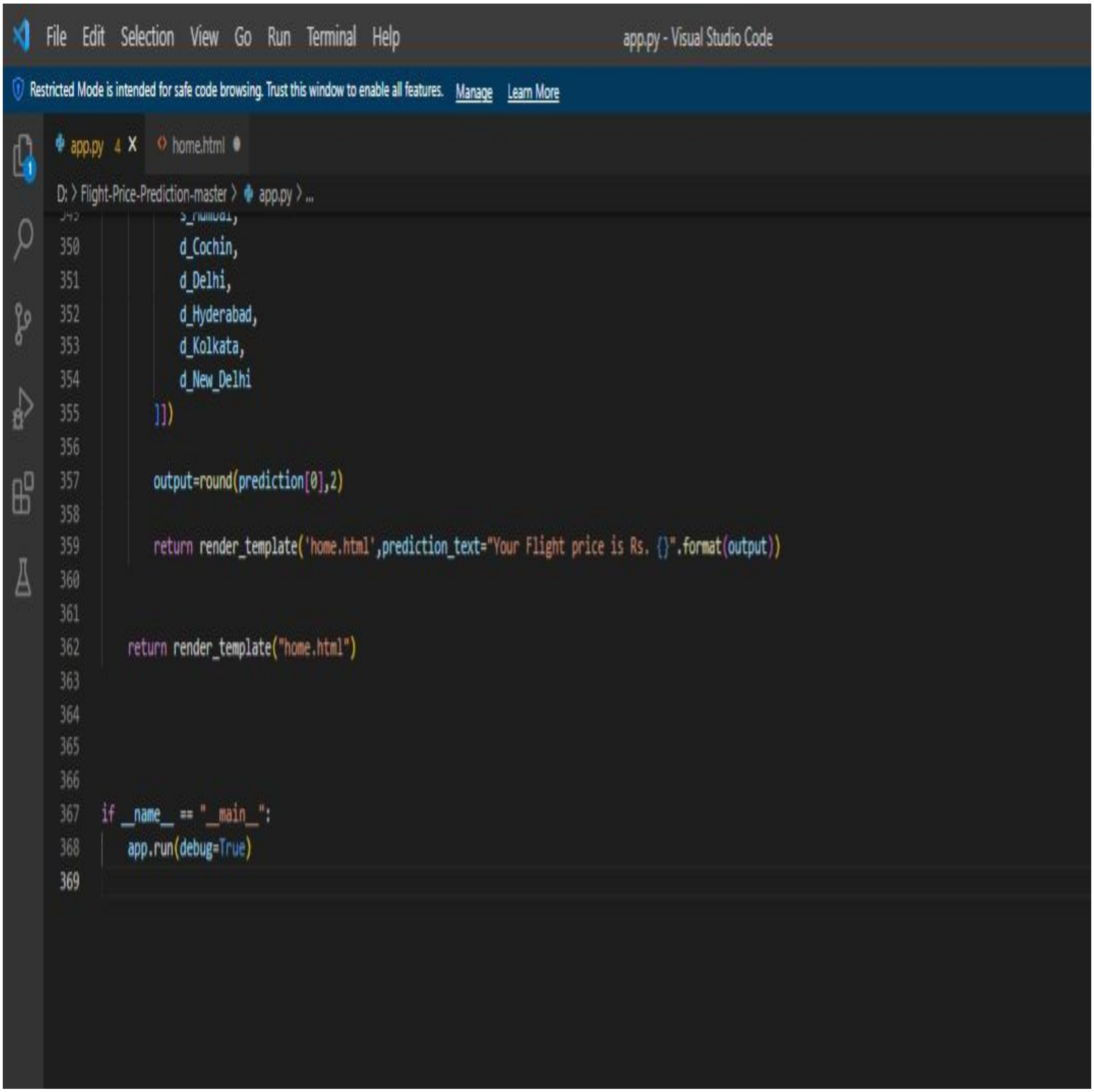
## ➢ Heat map:



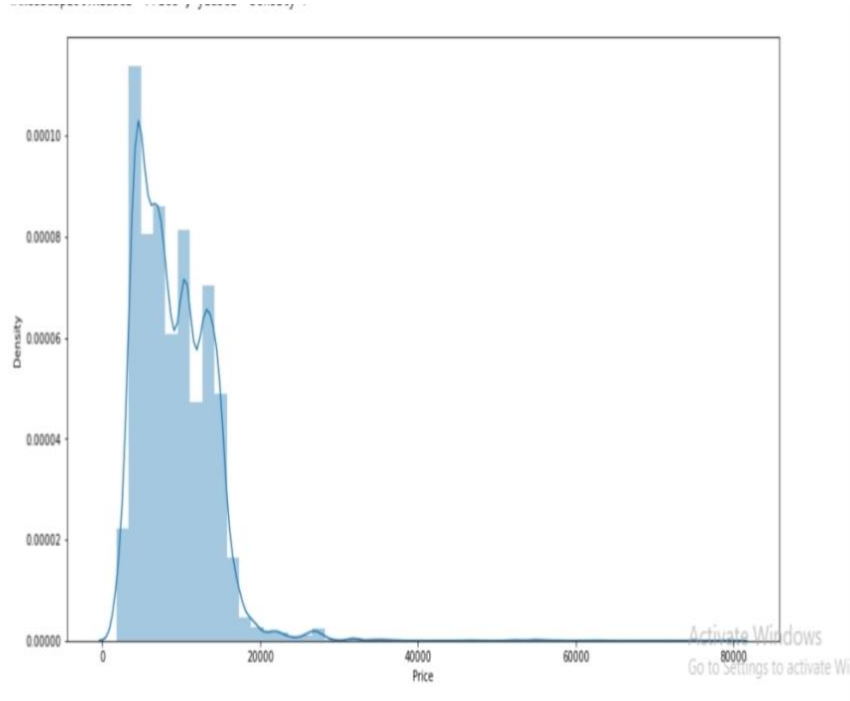
## ➢ Visual Analysis:



- **Visual studio code:**



- **Distribution in numerical DATA:**




- **Splitting data into train and test:**

```
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)
```

```
x_train.head()
```

| | Airline | Source | Destination | Date | Month | Year | Dep_Time_Hour | Dep_Time_Mins | Arrival_date | Arrival_Time_Hour | Arrival_Time_Mins |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **10611** | 4 | 4 | 3 | 18 | 5 | 2019 | 7 | 5 | 18 | 8 | 30 |
| **1034** | 8 | 2 | 1 | 24 | 4 | 2019 | 15 | 45 | 24 | 22 | 5 |
| **8123** | 4 | 2 | 1 | 27 | 6 | 2019 | 2 | 15 | 27 | 12 | 35 |
| **4779** | 4 | 3 | 0 | 1 | 4 | 2019 | 6 | 30 | 1 | 18 | 15 |
| **3207** | 3 | 3 | 0 | 24 | 5 | 2019 | 18 | 5 | 24 | 23 | 30 |

### Model Deployment

➢ **Integrate with Web Framework**

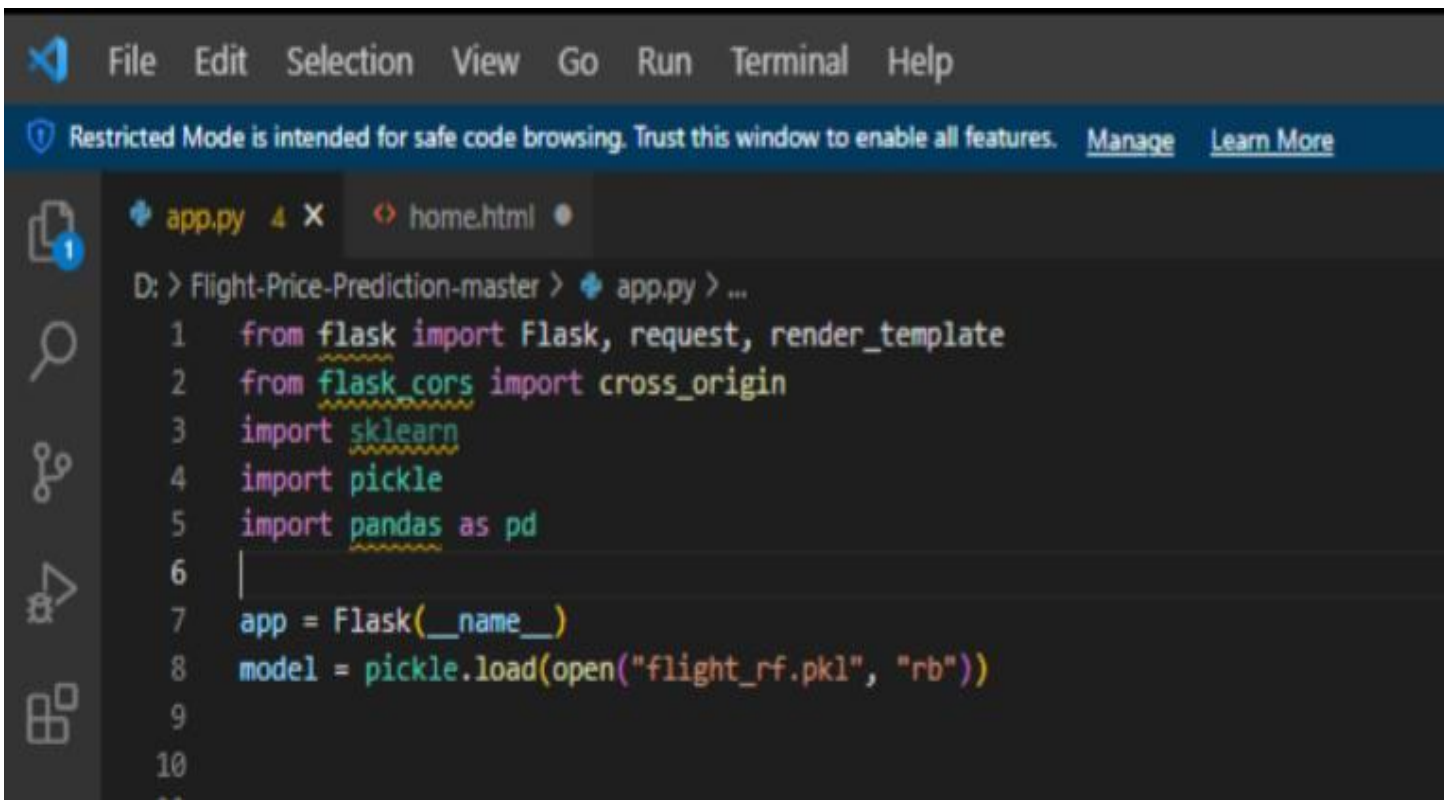
- **Building HTML Pages**
- **Building server side script**
- **Run the web application**

➢ **Building Html Pages:**

- **home.html**
- **predict.html**
- **submit.html**

**And save them in the templates folder.**

➢ **Python code libraries:**
**Import the libraries**



- **Render HTML page:**

```
@app.route("/home")
def home():
    return render_template('home.html')
```

**Here we will be using a declared constructor to route to the HTML page which we have created earlier.**

**In the above example, '/' URL is bound with the home.html function. Hence, when the home page of the Web server is opened in the browser, the html page will be rendered.**

```
@app.route("/predict")
def home1():
    return render_template('predict.html')



@app.route("/pred", methods=['POST','GET'])
def predict():
    x = [[int(x) for x in request.form.values()]]
    print(x)

    x = np.array(x)
    print(x.shape)


    print(x)
    pred = model.predict(x)
    print(pred)
    return render_template('submit.html', prediction_text=pred)
```

**Here we are routing our app to predict() function.**

## ➢ Run The Web Application:

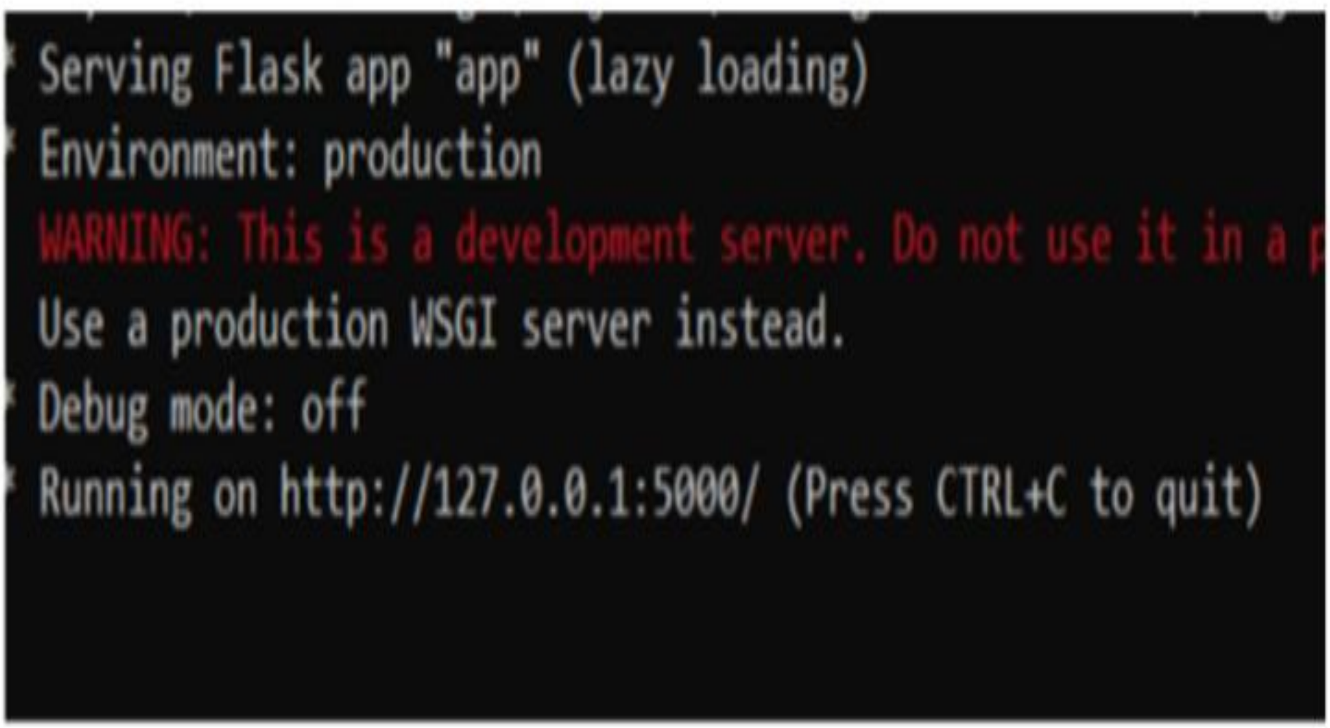
●Open anaconda prompt from the start menu

● Navigate to the folder where your python script is.

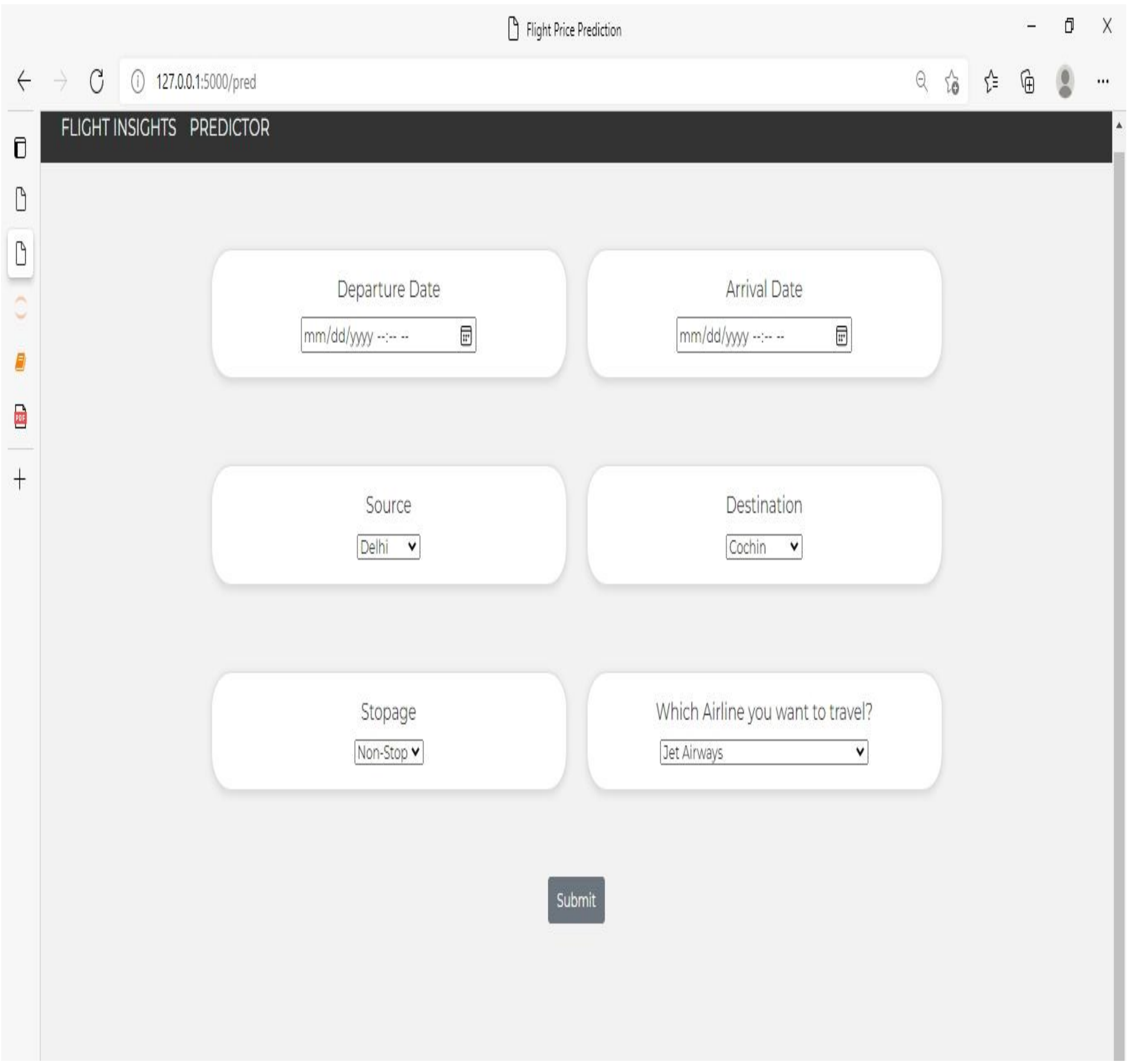
● Now type “python app.py” command

● Navigate to the localhost where you can view your web page.

● Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

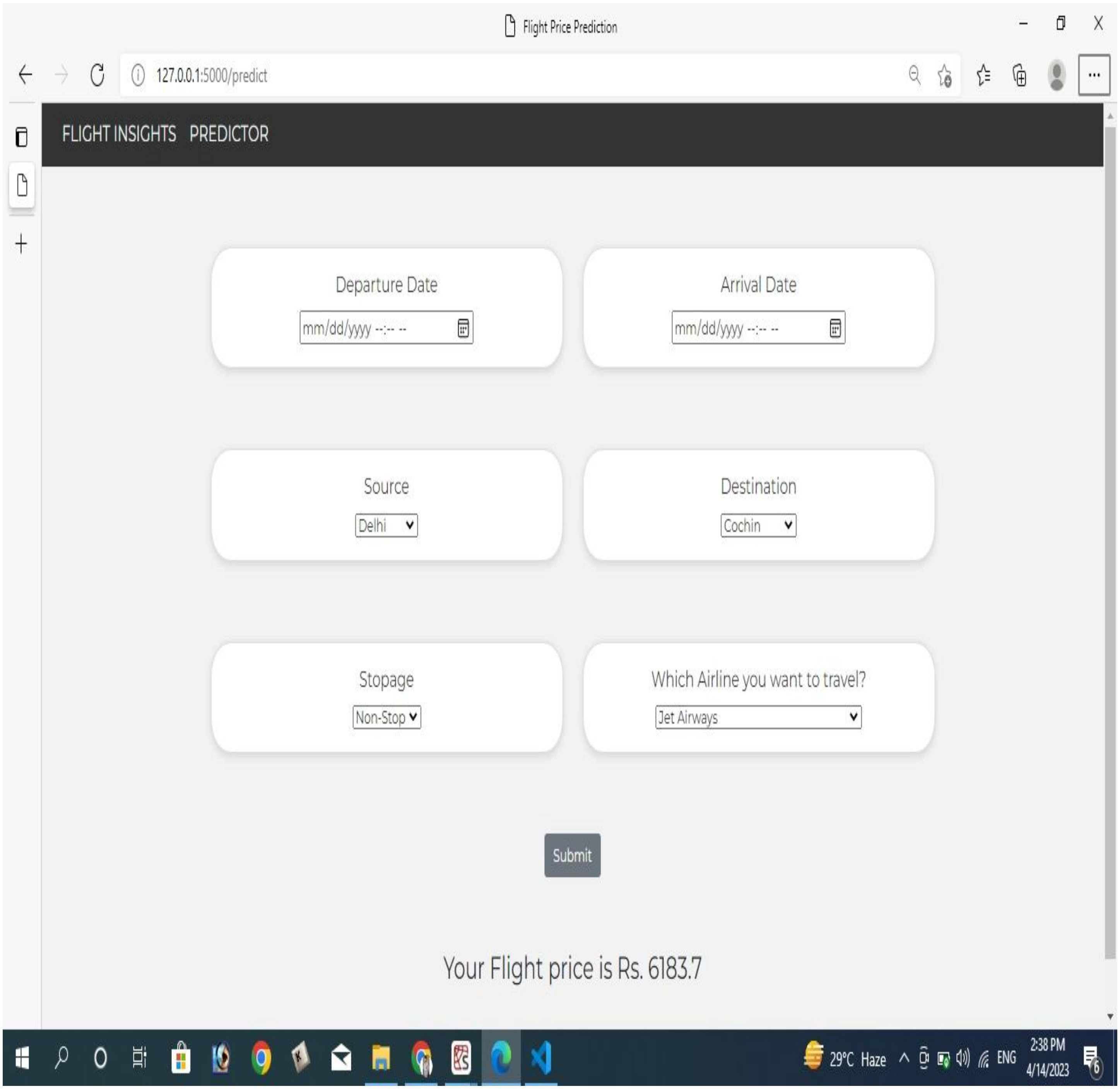


- **Prediction input:**




**Input 1- Now, the user will give inputs to get the predicted result after clicking onto the submit button.**

- **Prediction Output:**



# 4 TRAILHEAD PROFILE PUBLIC URL

**Team Lead –** https://trailblazer.me/id/karthim1

**Team Member 1 –** https://trailblazer.me/id/murugesan13

**Team Member 2 –** https://trailblazer.me/id/chinna1432

**Team Member 3 –** https://trailblazer.me/id/patturaja23

# 5 ADVANTAGES & DISADVANTAGE

**Advantages:**

- **Improved accuracy:** A machine learning price prediction model can provide more accurate flight prices than a human can. This can help users make better-informed decisions when booking flights.

- **Cost savings:** By providing users with the best possible prices for flights, the machine learning model can help save money on travel expenses.

- **Faster decision-making:** A machine learning model can quickly process large amounts of data and provide recommendations in real-time, which can help users make decisions faster.

- **Personalization:** Machine learning models can be trained to make personalized recommendations based on a user's travel history, preferences, and other factors, making the booking experience more tailored to the individual.

- **Scalability:** Machine learning models can handle large volumes of data and make predictions at scale, which can be particularly useful for travel booking websites that have a high volume of traffic.

**Disadvantages:**

- **Data quality:** Machine learning models rely heavily on the quality of data they are trained on. If the data is incomplete or inaccurate, the model's predictions will also be inaccurate.

- **Complex algorithms:** Machine learning models can be complex, and it can be difficult for non-experts to understand how they work. This can make it challenging to explain to users how the model arrived at a particular recommendation.

- **Overfitting:** If the machine learning model is trained on a limited set of data, it may overfit to that data and not generalize well to new data. This can result in inaccurate predictions.

- **Changing market conditions:** Flight prices can be influenced by a wide range of factors, including seasonality, weather, and geopolitical events. These factors can change rapidly and unpredictably, which can make it challenging for machine learning models to keep up with the latest trends and provide accurate predictions.

- **Privacy concerns:** Machine learning models require access to user data to make personalized recommendations. This can raise privacy concerns, particularly if the data is sensitive or personal.

## 6 APPLICATIONS

To develop a project that optimizes flight booking decisions through machine learning price prediction, here are the steps you can take:

- **Collect data:** Gather historical flight price data from various airlines and booking websites, as well as any other relevant data such as seasonality, demand trends, and economic indicators.

- **Preprocess data:** Clean and preprocess the data to remove any inconsistencies or missing values. You may also need to perform feature engineering to extract relevant features from the data.

- **Train a machine learning model:** Use the preprocessed data to train a machine learning model, such as a regression model or a neural network, to predict future flight prices.

- **Evaluate the model:** Evaluate the performance of the model using metrics such as mean squared error or root mean squared error. You may need to fine-tune the model by adjusting hyperparameters or using different algorithms to improve its performance.

- **Develop a user interface:** Create a user interface that allows users to input their travel details, such as departure and arrival airports, travel dates, and preferred airline. The interface should display predicted flight prices based on the machine learning model.

- **Deploy the application:** Deploy the application to a web server or cloud platform, such as AWS or Heroku, so that it can be accessed by users.

- **Monitor and update the application:** Continuously monitor the performance of the machine learning model and update it as needed to ensure that it remains accurate and up-to-date with the latest pricing and travel trend.

## 7 CONCLUSION

**In conclusion, the machine learning price prediction project aimed to optimize flight booking decisions by predicting the future prices of flights. Through the project, we were able to collect and preprocess a large dataset of historical flight prices and relevant features. We then trained several machine learning models, including linear regression, decision trees, and random forests, to predict flight prices.**

**After evaluating the models' performance using metrics such as mean squared error and mean absolute error, we found that the random forest model provided the best predictions. The random forest model also allowed us to identify the most important features that affect flight prices, such as the number of days before the flight, the airline, and the departure airport.**

**Overall, the project demonstrated the potential of using machine learning to optimize flight booking decisions and improve the accuracy of price predictions. Future work could include incorporating real-time data and additional features to further improve the model's performance and help travelers make more informed booking decisions.**

## 8 FUTURE SCOPE

**The future scope of a project that optimizes flight booking decisions through machine learning price prediction is vast, as the technology continues to advance and more data becomes available. Here are some potential areas for future development:**

- **Incorporating real-time data: While historical flight price data is useful for predicting future prices, real-time data can provide even more accurate predictions. In the future, machine learning models could be trained on real-time data such as weather conditions, flight delays, and cancellations to provide more accurate and up-to-date predictions.**

- **Personalized pricing: Machine learning algorithms can also be used to personalize flight prices based on a traveler's past booking behavior, preferences, and other factors. This could lead to more targeted pricing strategies and increased revenue for airlines.**

- **Integration with other travel services: Machine learning price prediction could be integrated with other travel services such as hotel and car rental bookings to provide a comprehensive travel planning platform.**

- **Expansion to other industries: The same technology used to predict flight prices could also be applied to other industries such as hotel bookings, car rentals, and e-commerce retail to optimize pricing strategies and improve revenue management.**