SmartInternz Long Term Virtual Internship

On

"Optimizing flight Booking Decisions Through Machine Learning Price Prediction"

An Internship Report submitted in partial fulfillment of the requirements for the award of degree of

BACHELOR OF TECHNOLOGY

In

ELECTRONICS AND COMMUNICATION ENGINEERING

Submitted by:

Team Id: LTVIP2023TMID06268

Team Leader: D Bhavana Giri

Team Member: R Sushma Sree

Team Member: Chintha Anitha

Team Member: Sunkara Akhila

Team Member: N Eshwar

Under the Esteemed Guidance of

Dr P Ajay Kumar Reddy, M Tech, Ph.D

Associate Professor

DEPARTMENT OF ELECTRONICS & COMMUNICATION ENGINEERING

KUPPAM ENGINEERING COLLEGE

ANDHRA PRADESH

517425

Approved by AICTE, New Delhi, affiliated to JNTU, Anantapur

Accredited by NBA & Certified by ISO 9001:2008

Recognition of UGC under 2(f) & 12(B)

CONTENTS

Abstract

A lot of factors that affect the overall price of airline tickets, including the airline, the date of travel, source, destination, route, duration, and so on. Each provider seems to have its own unique set regulations and methods for determining pricing. Recent breakthroughs in Artificial Intelligence (AI) and Machine Learning (ML) allow for the inference of such principles as well as the modelling of price volatility. This project gives information on predicting flight prices. Utilizing two datasets for testing and training, this study analyses various machine learning methods for predicting flight prices.

CHAPTER 1

Introduction:

Fare prediction is a classic time series forecasting problem that finds trends in past observations to make future predictions. Many popular flight booking websites today, including Google Flights, Ease My Trip, Goibibo, showcase intelligent insights on flight fare trends to help user decide what's the right time to book a flight ticket.

1.1 Why is flight price prediction important?

There are two main use cases of flight price prediction in the travel industry. OTAs and other travel platforms integrate this feature to attract more visitors looking for the best rates. Airlines employ the technology to forecast rates of competitors and adjust their pricing strategies accordingly.

1.2 Machine Learning Tools:

Machine learning is one of the most revolutionary technologies that is making lives simpler. It is a subfield of Artificial Intelligence, which analyses the data, build the model, and make predictions. Due to its popularity and great applications, every tech enthusiast wants to learn and build new machine learning Apps. However, to build ML models, it is important to master machine learning tools. Mastering machine learning tools will enable you to play with the data, train your models, discover new methods, and create algorithms. There are different tools, software, and platform available for machine learning, and also new software and tools are evolving day by day. Although there are many options and availability of Machine learning tools, choosing the best tool per your model is a challenging task.

1.3 Pre-Requisites:

Our flight fare prediction dataset has 10,684 observations with booking details such as: airline, date of journey, source, destination, route, departure time, arrival time, duration, stoppages, additional info (as applicable), and lastly the price, which is our target variable. These are the important factors which is used to predict the flight price. we will be analyzing the flight fare prediction using Machine Learning dataset using essential exploratory data analysis techniques then will draw some predictions about the price of the flight based on some features such as what type of airline it is, what is the arrival time, what is the departure time, what is the duration of the flight, source, destination and more.

1.4 Prior Knowledge:

The flight price predictor estimates flight fares by comparing previous prices of flights for the dates and destinations. The flight price predictor will also look at different airlines' previous flight fares to find out when is the best time to book flights so you can secure the cheapest deal. There are so many different reasons that determine the price of a flight ticket. This includes the time of booking, the day of the week and the demand. Flight fares are likely to increase closer to the date of the departure as the aircraft has fewer seats available. Therefore since passengers want to travel, airlines will take advantage and increase flight fares.

1.5 Project Objectives:

This project aims to predict flight prices for different flights using the machine learning model. The user receives the expected values, and using these as a guide, they can choose whether to purchase tickets. This model helps its users by advising them whether to buy tickets or wait for a suitable time to get the optimal deal. It uses data mining techniques like Rule Learning, Reinforcement Learning, time-series methods, and their combinations to achieve greater accuracy in predicting the fare of flights. Features considered for the study include flight number, hours till departure, the current price of a ticket, airline, and its route.

1.6 Technology Stack:

Software:

1) Jupyter Notebook

2) Visual studio Code

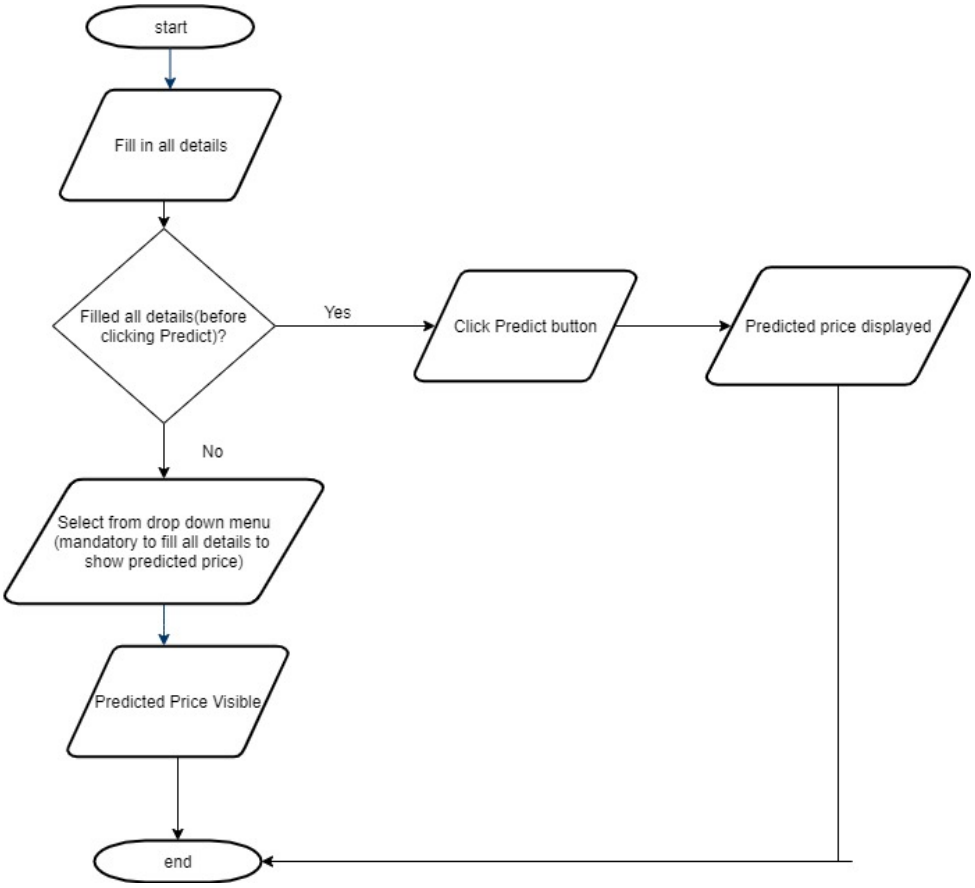
Technology:

1) Machine Learning Using Python

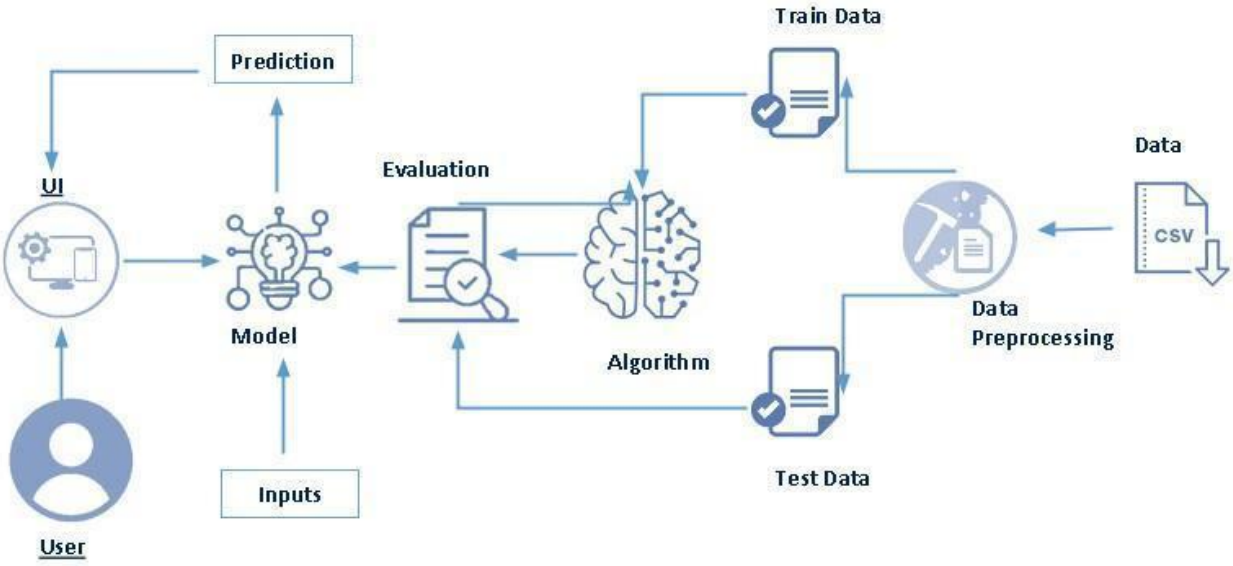
2)HTML

3) CSS

1.7 Project Flow:



1.8 Project Structure:



CHAPTER 2

Problem Understanding:

2.1 Specify The Business Problem:

Optimizing flight booking decisions through machine learning price prediction can help address several business problems in the airline industry. Here are some specific challenges that can be tackled:

1. Price volatility: Flight ticket prices are subject to frequent changes due to various factors such as demand, competition, fuel costs, and seasonality. Predicting these price fluctuations accurately can help airlines optimize their revenue management strategies and offer competitive prices to customers.

2. Demand forecasting: Accurately predicting future demand for flights is crucial for airlines to optimize their capacity planning, flight scheduling, and resource allocation. Machine learning models can analyze historical data and other relevant factors to forecast demand, enabling airlines to make informed decisions about flight availability and pricing.

3. Customer segmentation: Understanding customer preferences and behavior is essential for airlines to tailor their offerings and marketing strategies. Machine learning algorithms can analyze customer data to segment travelers based on factors like travel patterns, demographics, and past purchase behavior. This segmentation can help airlines personalize their pricing strategies and promotional campaigns to target specific customer segments effectively.

4. Competitive pricing: Airlines operate in a highly competitive market, and offering the right price at the right time is crucial for attracting customers. Machine learning models can analyze competitor pricing data and market trends to provide insights on optimal pricing strategies. This can help airlines adjust their prices dynamically, stay competitive, and maximize their revenue.

5. Overbooking Issues: If prices are predicted too low and flights are overbooked, airlines could face logistical challenges and potentially to need to compensate for denied boarding.

2.2 Business Requirements:

1. Data sources: The system should be able to access and collect data from various sources such as airline reservation systems, third-party travel websites, market data, and social media. This data should be cleaned, preprocessed, and integrated into a single database for analysis.

2. Machine learning algorithms: The system should be able to apply various machine learning algorithms such as regression, time-series analysis, and clustering to predict flight prices accurately. The algorithms should be selected based on the specific business problem and the available data.

3. Real-time prediction: The system should be able to provide real-time predictions of flight prices to enable airlines to adjust their pricing strategies quickly. This requires a scalable and responsive system architecture that can handle large volumes of data and user requests.

4. Accuracy and reliability: The system should provide accurate and reliable predictions to ensure that airlines can make informed decisions. The accuracy of the predictions should be continuously monitored and evaluated to improve the performance of the system.

5. Integration with existing systems: The system should be integrated with existing airline systems such as revenue management, inventory management, and customer relationship management to enable seamless data exchange and decision-making.

6. Security and privacy: The system should ensure the security and privacy of customer data.

2.3 Literature Survey:

1) K. Tziridis T. Kalampokas G.Papakostas and K.Diamantaras(2009) titled "Airfare Price Prediction Using Machine Learning Techniques", the researchers focused on predicting airfare prices by employing machine learning methods. They gathered a dataset consisting of 1814 flight records from Aegean Airlines, which was used to train their machine learning models. To explore the impact of feature selection on model accuracy, they experimented with different combinations of features. Several machine learning algorithms were utilized in their study, Random Forest Regression Tree, Regression Tree and Linear Regression (LR). [1]

2) William Groves and Maria Gini "An agent for optimizing airline ticket purchasing" in proceedings of the 2013 international conference on autonomous agents and multi-agent systems.

In the case study conducted by William Groves [2], an agent is introduced with the capability to optimize the timing of ticket purchases on behalf of customers. The study utilizes the technique of Partial Least Square (PLS) regression to build a predictive model. Initially, various techniques for feature selection are employed, including Feature Extraction, Lagged Feature Computation, Regression Model Construction, and Optimal Model Selection. The experiments conducted in the study aim to estimate the real-world costs associated with using the prediction.

3) Supriya Rajankar, Neha sakhrakar and Omprakash rajankar “Flight fare prediction using machine learning algorithms” International journal of Engineering Research and Technology (IJERT) June 2019.

In the survey conducted by Supriya Rajankar, the focus was on flight fare prediction using machine learning algorithms. The dataset used in the study consisted of flights between Delhi and Bombay. Various machine learning algorithms were applied, including K-nearest neighbors (KNN), linear regression, and support vector machine (SVM), to obtain different outcomes and analyze their performance. To predict flight ticket prices, several machine learning algorithms were implemented, such as Support Vector Machine (SVM), Linear Regression, K-Nearest Neighbors, Decision Tree, Multilayer Perceptron, Gradient Boosting, and Random Forest.

2.4 Social Or Business Impact:

Optimizing flight booking decisions through machine learning price prediction can have both social and business impacts. Here are some key points:

1. Social Impact:

Cost Savings: Machine learning price prediction can help travelers find the best deals and save money on their flight bookings. This can make air travel more affordable and accessible to a wider range of people.

Improved Planning: By accurately predicting flight prices, travelers can plan their trips in advance, reducing last-minute rushes and stress.

Enhanced Travel Experience: With cost savings, travelers may have more budget for other aspects of their trip, such as accommodation, activities, and dining, leading to an overall improved travel experience.

2. Business Impact:

Increased Revenue: Airlines and travel agencies can benefit from machine learning price prediction by optimizing their pricing strategies. By offering competitive prices based on accurate predictions, they can attract more customers and increase revenue.

Demand Forecasting: Machine learning can help airlines anticipate demand patterns and adjust flight capacities accordingly. This can lead to better resource allocation, reduced operational costs, and improved efficiency.

Customer Satisfaction: By providing customers with accurate price predictions and personalized offers, airlines and travel agencies can enhance customer satisfaction and loyalty.

CHAPTER 3

Data Collection and Preparation:

3.1 Data Collection:

Data is collected from different sources namely Kaggle. We will be using two datasets, train data and test data. The train data comprises of 10684 rows and 11 columns whereas the test data has 2672 rows and 10 columns. Following is the description of features available in the dataset

1. Airline: The name of the airline.

2. Date_of_Journey: The date of the journey

3. Source: The source from which the service begins.

4. Destination: The destination where the service ends.

5. Route: The route taken by the flight to reach the destination.

6. Dep_Time: The time when the journey starts from the source.

7. Arrival_Time: Time of arrival at the destination.

8. Duration: Total duration of the flight.

9. Total_Stops: Total stops between the source and destination.

10. Additional_Info: Additional information about the flight.

11. Price: The price of the ticket.

3.2 Data Preparation:

Remove duplicates, as having duplicate entries can skew your analysis. Handle missing values in a meaningful way. Depending on the feature and the amount of missing data, you can choose to impute values (replace missing values with estimated values) or consider excluding incomplete data. Divide your dataset into training, validation, and test sets. This helps you train your model on one subset, tune hyperparameters on another, and assess its performance on unseen data.

Convert categorical variables into numerical format using techniques like one-hot encoding or label encoding. For categorical variables like airlines or departure cities, use appropriate techniques like target encoding or embedding to represent them numerically.

CHAPTER 4

Exploratory Data Analysis:

Exploratory data analysis(EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

4.1 Descriptive Statistical Analysis: Descriptive statistical analysis helps you to understand your data and is a very important part of machine learning. This is due to machine learning being all about making predictions. On the other hand, statistics is all about drawing conclusions from data, which is a necessary initial step.

1) Concatenating categorical dataframe with all the dataframes that we have defined earlier. For this we will be using concat() from pandas library.

```
data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)
```

2) Concatenating the dataframes:

This code concatenates the modified test dataset with the dummy variable dataframes for "Airline", "Source", and "Destination" along the columns (axis=1).

```
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)
```

3) Printing the shape of the final test dataset:

This code prints the shape (number of rows and columns) of the final test dataset after preprocessing.

```
print("Shape of test data : ", data_test.shape)
```

```
Test data Info
-------------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Airline          2671 non-null   object
 1   Date_of_Journey  2671 non-null   object
 2   Source           2671 non-null   object
 3   Destination      2671 non-null   object
 4   Route            2671 non-null   object
 5   Dep_Time         2671 non-null   object
 6   Arrival_Time     2671 non-null   object
 7   Duration         2671 non-null   object
 8   Total_Stops      2671 non-null   object
 9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
None
```

```
Null values :
------------------------------------------
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
dtype: int64
```

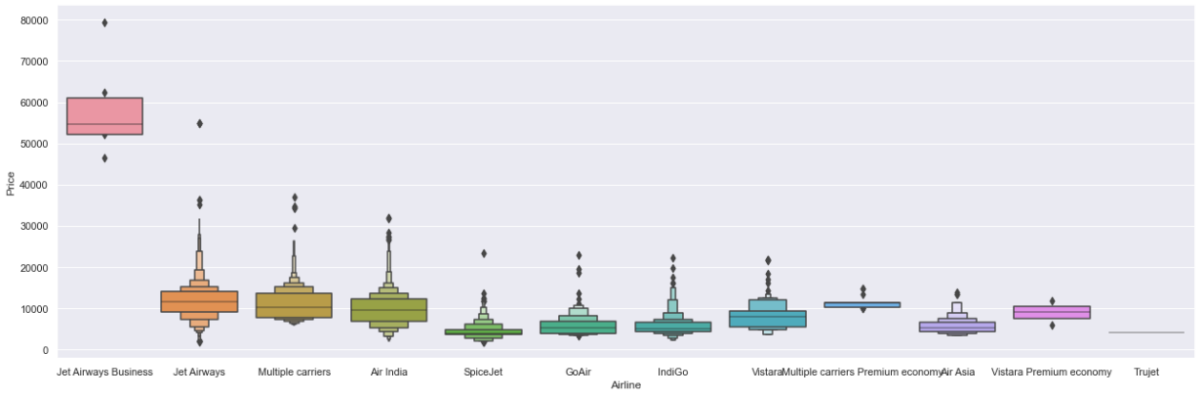
```
Airline
------------------------------------------------------
Airline
Jet Airways                          897
IndiGo                               511
Air India                            440
Multiple carriers                    347
SpiceJet                             208
Vistara                              129
Air Asia                              86
GoAir                                 46
Multiple carriers Premium economy      3
Vistara Premium economy                2
Jet Airways Business                   2
Name: count, dtype: int64
```

```
Source
------------------------------------------------
Source
Delhi       1145
Kolkata      710
Banglore     555
Mumbai       186
Chennai       75
Name: count, dtype: int64

Destination
------------------------------------------------
Destination
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: count, dtype: int64


Shape of test data :  (2671, 28)
```

4.2 Visual Analysis: Visual analytics is the use of sophisticated tools and processes to analyze datasets using visual representations of the data. Visualizing the data in graphs, charts, and maps helps users identify patterns and thereby develop actionable insights. These insights help organizations make better, data-driven decisions.

1) Now we will handle the categorical data and basically perform Feature Encoding because machine learning works only on numerical data.

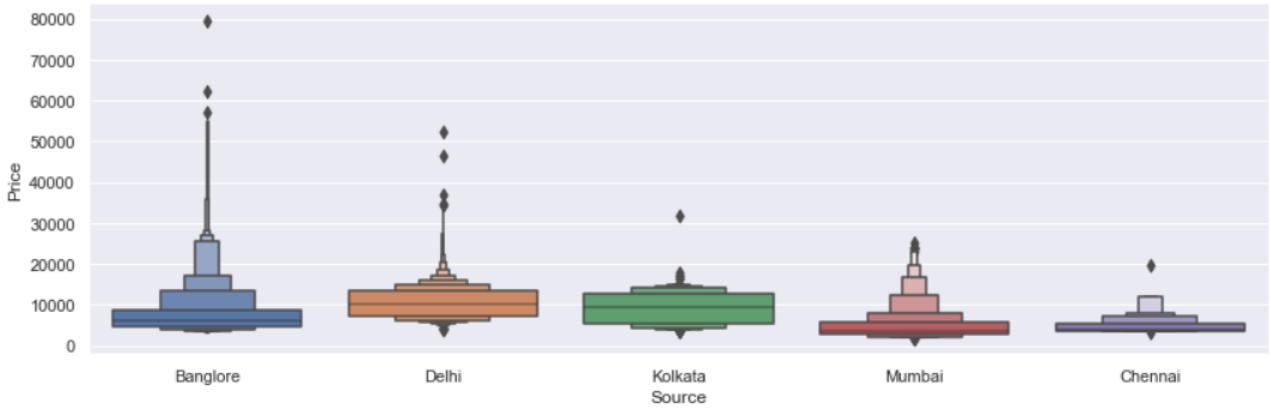
```
sns.catplot(y = "Price", x = "Airline", data = train_data.sort_values("Price", ascending = False), kind="boxen",
            height = 6, aspect = 3)
plt.show()
```



We can come up with a conclusion that Jet airways has the highest price whereas other airlines had almost similar median with minimal fluctuations.

2)

```
sns.catplot(y = "Price", x = "Source", data = train_data.sort_values("Price", ascending = False), kind="boxen",
            height = 4, aspect = 3)
plt.show()
```



Airlines with 1 or 2 stops has many outliers and hence their price varies. On the contrary price for airline with 4 stops is not fluctuating.

3)

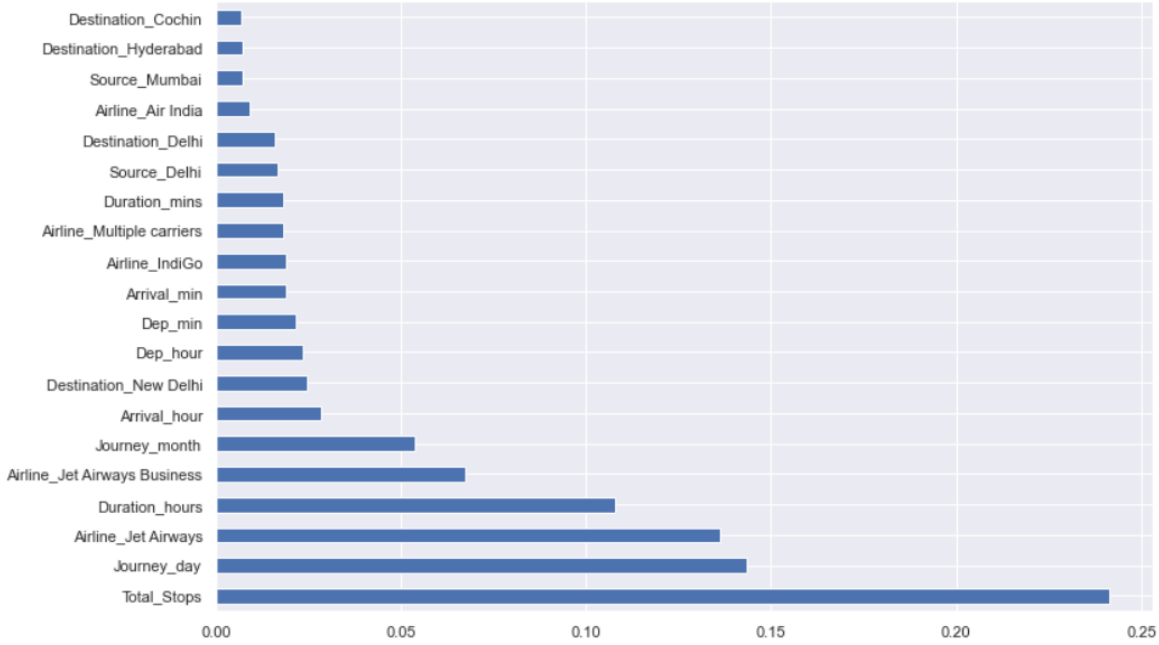
```
plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```

(i) plt.figure(figsize=(12, 8)): This line of code sets the figure size of the plot to be created. The figsize parameter takes a tuple of width and height values in inches. In this case, the figure size is set to 12 inches in width and 8 inches in height.

(ii) feat_importances = pd.Series(selection.feature_importances_, index=X.columns): This line of code creates a pandas Series object called feat_importances using the feature importances calculated by the ExtraTreesRegressor model. The selection.feature_importances_ contains the importance scores, and X.columns represents the column names of the input features.

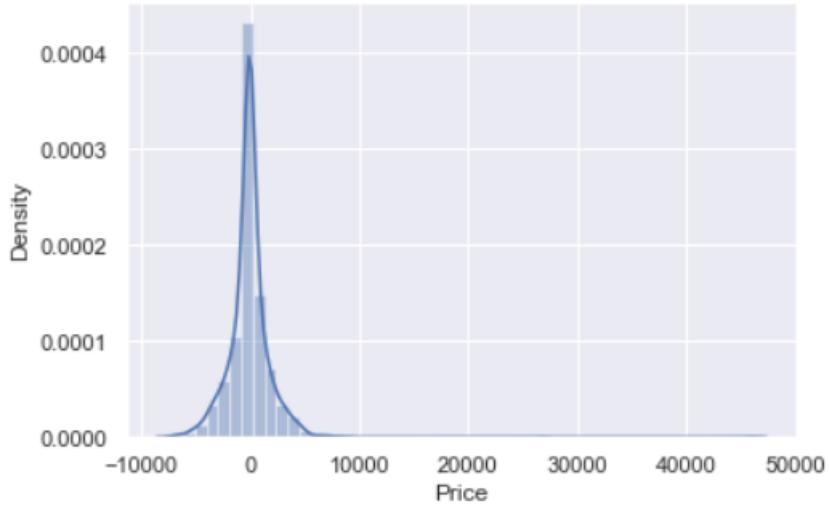
(iii) feat_importances.nlargest(20).plot(kind='barh'): This line of code selects the top 20 feature importances from the feat_importances Series using the nlargest method. It then creates a horizontal bar plot using the plot method with kind='barh'. This means that the feature importances will be plotted as horizontal bars.

(iv) plt.show(): This line of code displays the plot on the screen. The plt.show() function is used to render the plot and show it in a separate window.



4) The code snippet sns.distplot(y_test-y_pred) and plt.show() is used to create and display a distribution plot of the differences between the true target values (y_test) and the predicted target values (y_pred) obtained from a regression model.

```
sns.distplot(y_test-y_pred)
plt.show()
```

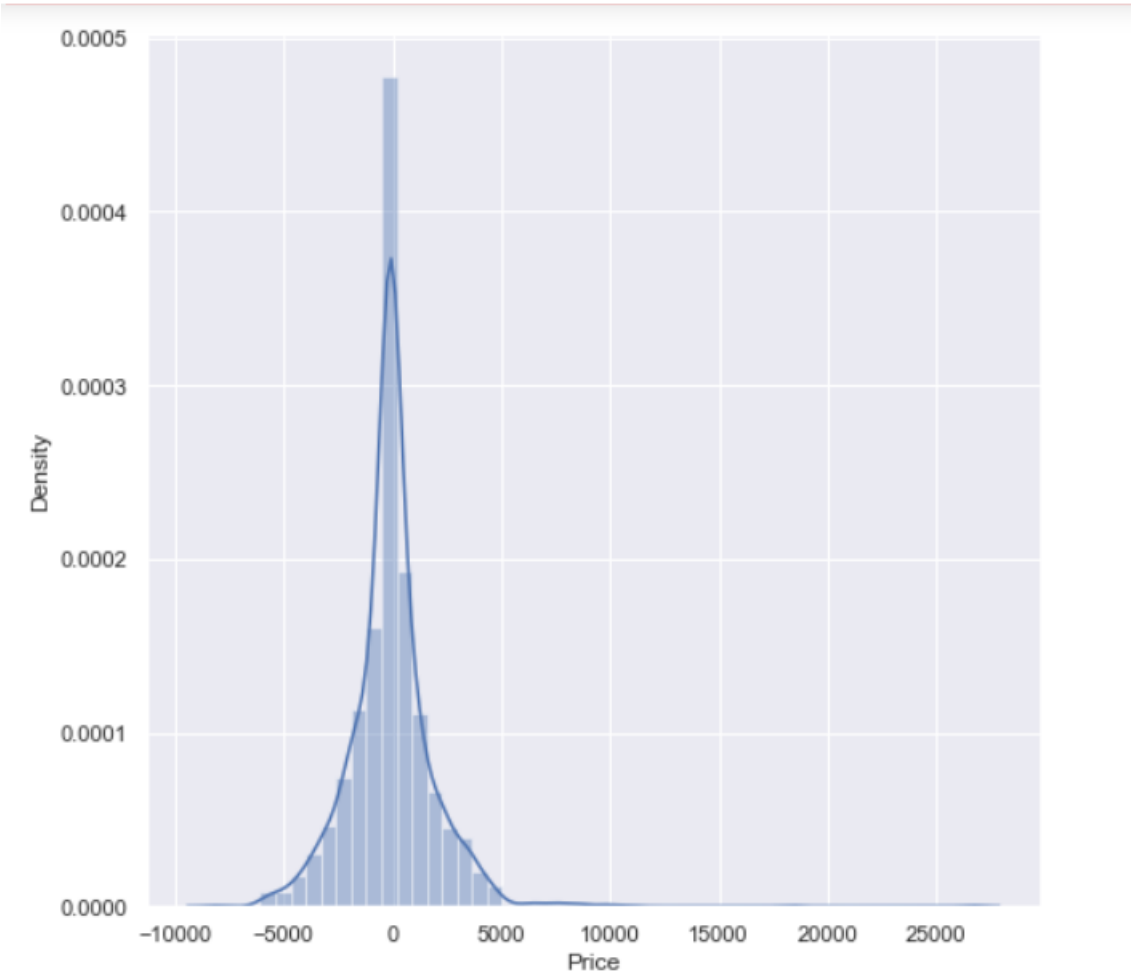


5) The code snippet sns.distplot(y_test-y_pred) and plt.show() is used to create and display a distribution plot of the differences between the true target values (y_test) and the predicted target values (y_pred) obtained from a regression model. By subtracting y_pred from y_test, we obtain the differences between the true and predicted values. After creating the distribution plot, plt.show() is called to display the plot in a separate window.

```
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

6) plt.figure(figsize = (8,8)): This line sets the figure size of the plot to 8 inches by 8 inches.sns.distplot(y_test-prediction) This line creates a histogram plot using the distplot function from the seaborn library. The y_test-prediction expression calculates the residuals by subtracting the predicted values from the actual values. The distplot function then visualizes the distribution of these residuals. plt.show() This line displays the plot on the screen.

```
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
```

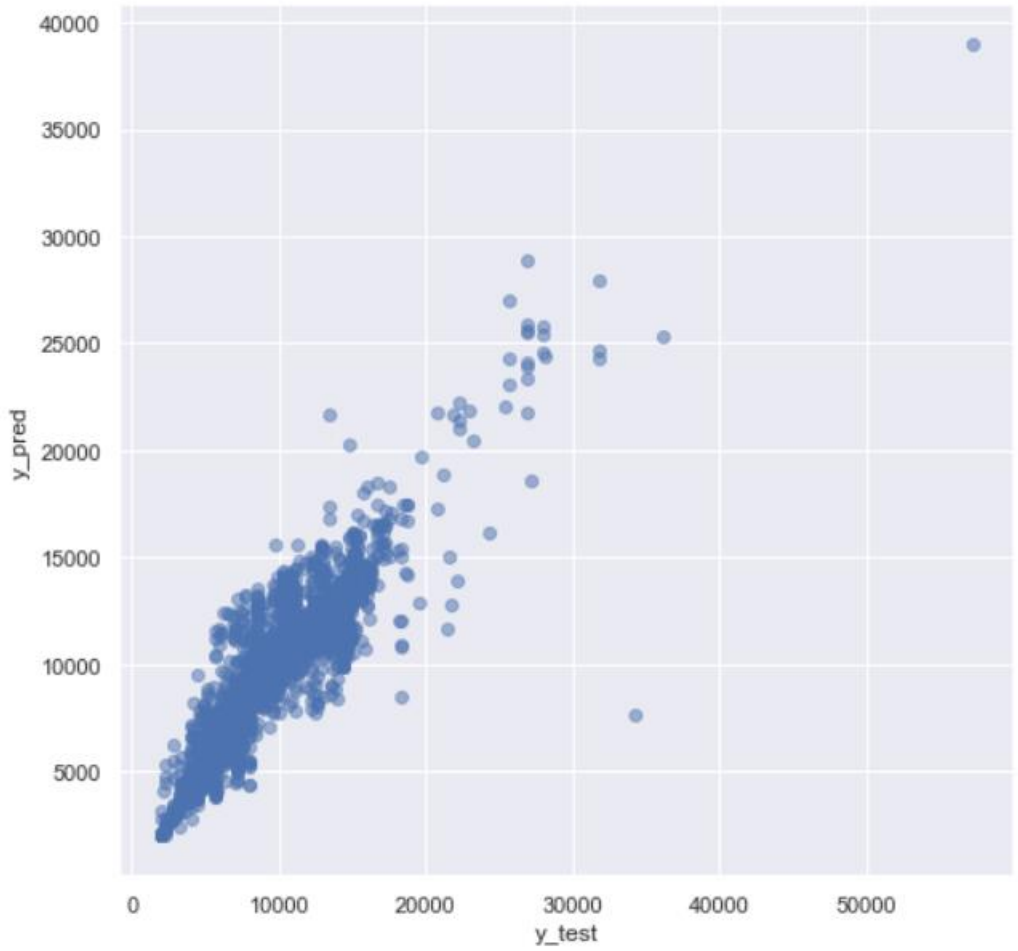


7) plt.figure(figsize = (8,8)): This line sets the figure size of the plot to 8 inches by 8 inches. plt.scatter(y_test, prediction, alpha = 0.5): This line creates a scatter plot using the scatter function from matplotlib. The y_test values are plotted on the x-axis, while the prediction values are plotted on the y-axis. Each point on the scatter plot represents a data point from the test set. The alpha parameter sets the transparency level of the points, with a value of 0.5 indicating a semi-transparent plot.

plt.xlabel("y_test") and plt.ylabel("y_pred"): These lines set the labels for the x-axis and y-axis, respectively.

plt.show(): This line displays the plot on the screen.

```
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



4.3 Training the Model In Multiple Algorithms:

1) RandomForestRegressor is instantiated as an object named reg_rf without specifying any hyperparameters. By default, the RandomForestRegressor class uses 100 decision trees and other default settings, such as the mean squared error as the criterion for splitting and the mean absolute error as the criterion for evaluating the quality of a split.

```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

2) **Number of trees in random forest (n_estimators):** This parameter determines the number of decision trees that will be built in the random forest. In the code, it is set to a list of 12 evenly spaced values between 100 and 1200 (inclusive).

Number of features to consider at every split (max_features): n the code, it is set to a list containing the values 'auto' and 'sqrt'. 'auto' means that all features will be considered, while 'sqrt' means that the square root of the total number of features will be considered.

Maximum number of levels in tree (max_depth): This parameter limits the maximum depth of each decision tree in the random forest. In the code, it is set to a list of 6 evenly spaced values between 5 and 30 (inclusive).

Minimum number of samples required to split a node (min_samples_split): This parameter specifies the minimum number of samples required to split an internal node in a decision tree. In the code, it is set to a list of values [2, 5, 10, 15, 100].

Minimum number of samples required at each leaf node (min_samples_leaf): This parameter specifies the minimum number of samples required to be at a leaf node in a decision tree. In the code, it is set to a list of values [1, 2, 5, 10].

```
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

3) The random_grid variable in your code snippet is a Python dictionary that defines a search space for hyperparameters in a random forest model. The keys in the dictionary correspond to the hyperparameter names ('n_estimators', 'max_features', 'max_depth', 'min_samples_split', 'min_samples_leaf'), and the values are the lists of possible values for each hyperparameter that

we defined earlier in the code.

```
# Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}
```

4) It seems like you are fitting a random forest model using the rf_random object that was created with the RandomizedSearchCV function. Assuming that X_train represents the training data and y_train represents the corresponding target labels, the fit() method is used to train the random forest model on the given training data.

```
rf_random.fit(X_train,y_train)
```

5) Importing the ExtraTreesRegressor class from the sklearn.ensemble module in the scikit-learn library. ExtraTreesRegressor is a machine learning algorithm that belongs to the ensemble methods family, specifically the Random Forest algorithm. It is used for regression tasks, where the goal is to predict continuous numerical values.

The fit method takes two arguments, X and y. X represents the input features or independent variables, and y represents the target variable or dependent variable.

```
from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)
```

```
ExtraTreesRegressor()
```

6) The code print(selection.feature_importances_) is used to print the feature importances calculated by the ExtraTreesRegressor model.

```
print(selection.feature_importances_)
```

```
[2.41217840e-01 1.43358703e-01 5.37086823e-02 2.34083467e-02
 2.15230248e-02 2.82871704e-02 1.90783388e-02 1.08048795e-01
 1.81975140e-02 9.25849126e-03 1.48833966e-03 1.88138801e-02
 1.36121069e-01 6.74289298e-02 1.82992672e-02 9.04279500e-04
 2.74460963e-03 1.10326065e-04 5.20800031e-03 8.41530365e-05
 5.64185146e-04 1.65500190e-02 3.19207177e-03 7.19271793e-03
 6.81011653e-03 1.59977089e-02 7.09641716e-03 4.68935980e-04
 2.48380666e-02]
```

4.4 Testing The Model:

model testing is referred to as the process where the performance of a fully trained model is evaluated on a testing set.

Data Train

1) Lets import the necessary libraries first.

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

sns.set()
```

2) Import training data and display the first five rows to take the overall view of the data.

```
train_data = pd.read_excel(r'C:\Users\Dineshgiri\Desktop\Flight_price_prediction-main\Data_Train.xlsx')
```

```
pd.set_option('display.max_columns', None)
```

```
train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

3) We will check whether the data contains any null values using info() and sum() method and then drop the NaN values using dropna() method so that there are no discrepancies in our data by which we can predict precisely.

```
train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          10683 non-null  object
 1   Date_of_Journey  10683 non-null  object
 2   Source           10683 non-null  object
 3   Destination      10683 non-null  object
 4   Route            10682 non-null  object
 5   Dep_Time         10683 non-null  object
 6   Arrival_Time     10683 non-null  object
 7   Duration         10683 non-null  object
 8   Total_Stops      10682 non-null  object
 9   Additional_Info  10683 non-null  object
 10  Price            10683 non-null  int64 
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

```
train_data.dropna(inplace = True)
```

```
train_data.isnull().sum()
```

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
Price              0
dtype: int64
```

4) Separate the column “Date_of_Journey” into “journey_day” and “journey_month” to help our machine learning model understand and use the column for prediction. After doing so we will delete the “Date_of_Journey” column which will no longer be useful to us.

```
train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey, format="%d/%m/%Y").dt.day
```

```
train_data["Journey_month"] = pd.to_datetime(train_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
```

```
train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	6
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12	5
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1	3

5) Now we will be dealing with the “Dep_time” and Arrival_Time” feature because the machine learning model won’t be able to understand what the time is or what day it is. We will be fetching minutes and hours from both these columns as follows:

```
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour

train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute

train_data.drop(["Dep_Time"], axis = 1, inplace = True)
```

```
train_data.head()
```

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR → DEL	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3	22	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	13:15	7h 25m	2 stops	No info	7662	1	5	5	50
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	04:25 10 Jun	19h	2 stops	No info	13882	9	6	9	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	23:30	5h 25m	1 stop	No info	6218	12	5	18	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	21:35	4h 45m	1 stop	No info	13302	1	3	16	50

```
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour

# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
```

```
train_data.head()
```

Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20	1	10
Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5	50	13	15
Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9	25	4	25
IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18	5	23	30
IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16	50	21	35

6) Now we will be processing the “Duration” column as in some cases there is no Hour(hr) term or minutes (m) term in the colum. We will be using the split function and then append ‘0h’ or ‘0m’ wherever needed.

```
duration = list(train_data["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
        else:
            duration[i] = "0h " + duration[i]           # Adds 0 hour
```

7) Let’s separate the ‘Duration’ attribute into ‘Duration_hours’ and ‘Duration_mins’ using the apply function.

```
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
```

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

8) Here we will be changing the ‘Airline’ feature into integer format using one hot encoding

```
Airline = pd.get_dummies(Airline, drop_first= True)

Airline.head()
```

Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airline_Trujet	Airline_Vistara	Airline_Vistara Premium economy
False	False	True	False	False	False	False	False	False	False	False
True	False	False	False	False	False	False	False	False	False	False
False	False	False	True	False	False	False	False	False	False	False
False	False	True	False	False	False	False	False	False	False	False
False	False	True	False	False	False	False	False	False	False	False

All the features are converted to integer values using get_dummies function. We will apply the get_dummies function on the ‘Source’ and ‘Destination’ column as well.

```
Source = train_data[["Source"]]

Source = pd.get_dummies(Source, drop_first= True)

Source.head()
```

	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	False	False	False	False
1	False	False	True	False
2	False	True	False	False
3	False	False	True	False
4	False	False	False	False

```
Destination = train_data[["Destination"]]

Destination = pd.get_dummies(Destination, drop_first = True)

Destination.head()
```

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	False	False	False	False	True
1	False	False	False	False	False
2	True	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	True

9) The ‘Route’ columns mainly tell us that how many cities they have taken to reach from source to destination

```
train_data["Route"]
```

```
0                  BLR → DEL
1          CCU → IXR → BBI → BLR
2          DEL → LKO → BOM → COK
3                CCU → NAG → BLR
4                BLR → NAG → DEL
                   ...
10678              CCU → BLR
10679              CCU → BLR
10680              BLR → DEL
10681              BLR → DEL
10682      DEL → GOI → BOM → COK
Name: Route, Length: 10682, dtype: object
```

10) Dealing with ‘Total_Stops’ attribute and assign 1 for ‘1 stop’, 2 for ‘2 Stops’ and so on.

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

11) Dropping the Airline, Source and destination.

```
data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
```

Test set

1) Import test data and display the first five rows to take the overall view of the data.

```
test_data = pd.read_excel(r"C:\Users\Dineshgiri\Desktop\Flight_price_prediction-main\Test_set.xlsx")
```

```
test_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info

2) Printing test data information and dropping null values

```
print("Test data Info")
print("-"*75)
print(test_data.info())
```

```
print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())
```

3) Extracting journey day and month from the "Date_of_Journey" column:

This code converts the "Date_of_Journey" column to datetime format and extracts the day and month information. Then, it drops the "Date_of_Journey" column from the dataset.

```
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey, format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

4) Extracting departure hour and minute from the "Dep_Time" column:

This code converts the "Dep_Time" column to datetime format and extracts the hour and minute information. Then, it drops the "Dep_Time" column from the dataset.

```
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)
```

5) Extracting arrival hour and minute from the "Arrival_Time" column:

This code converts the "Arrival_Time" column to datetime format and extracts the hour and minute information. Then, it drops the "Arrival_Time" column from the dataset.

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

6) Processing duration column:

```
duration = list(test_data["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
        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

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)
```

7) Handling “Airline”, “Source”, “Destination” column:

This code prints the count of each unique value in the "Airline" ”, “Source”, “Destination” column. Then, it uses the pd.get_dummies() function to convert the "Airline"”, “Source”, “Destination” column into dummy variables, creating separate columns for each unique value. The drop_first=True parameter is set to drop the first dummy variable to avoid multicollinearity.

```
print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)
```

```
print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)
```

```
print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)
```

8) Dropping unnecessary columns:

This code drops the "Route" and "Additional_Info" columns from the test dataset as they are not needed for further analysis.

```
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

9) Replacing "Total_Stops" values:

This code replaces the categorical values in the "Total_Stops" column with numerical values for easier analysis.

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

10) Dropping redundant columns:

This code drops the original "Airline", "Source", and "Destination" columns from the concatenated dataframe, as they have been converted into dummy variable.

```
data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)
```

11) train_test_split is a function used to split a dataset into training and testing sets.In the code snippet, train_test_split is called with four arguments: X, y, test_size, and random_state. X represents the input features or independent variables, and y represents the target variable or dependent variable. . The test_size parameter specifies the proportion of the dataset that should be used for testing, while the random_state parameter sets the seed for the random

number generator used to split the data. After calling train_test_split, the function returns four arrays: X_train, X_test, y_train, and y_test. X_train and y_train represent the training data, while X_test and y_test represent the testing data.

```
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)
```

CHAPTER 5

Performance Testing & Evaluate The Results:

5.1 Testing Model With Multiple Evaluation Metrics:

1) Thecode snippet print('MAE:', metrics. mean_absolute_error(y_test, y_pred)), print('MSE:', metrics. mean_squared_error(y_test, y_pred)), and print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred))) is used to calculate and print the mean absolute error (MAE), mean squared error (MSE), and root mean squared error (RMSE) metrics between the true target values (y_test) and the predicted target values (y_pred) obtained from a regression model.

In this code, metrics.mean_absolute_error() is a function from the scikit-learn library that calculates the MAE between y_test and y_pred. The MAE represents the average absolute difference between the true and predicted values. Similarly, metrics.mean_squared_error() calculates the MSE, which represents the average squared difference between the true and predicted values. To calculate the RMSE, we need to take the square root of the MSE. This is done using np.sqrt() from the NumPy library.

```
from sklearn import metrics
```

```
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 1179.306846772255
MSE: 4380940.713427991
RMSE: 2093.0696867108823
```

2) The expression 2090.5509/(max(y)-min(y)) calculates the normalized value of 2090.5509 based on the range of values in the variable y.

```
# RMSE/(max(DV)-min(DV))

2090.5509/(max(y)-min(y))
```

```
0.026887077025966846
```

3) The code metrics.r2_score(y_test, y_pred) is used to calculate the R-squared score between the true target values (y_test) and the predicted target values (y_pred) obtained from a regression model. The R-squared score represents the proportion of the variance in the target variable that can be explained by the regression model.

```
metrics.r2_score(y_test, y_pred)
```

```
0.7968217572774429
```

4) To Print MAE, MSE, RMSE.

```
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 1264.780648178373
MSE: 3898219.5594936693
RMSE: 1974.3909338055798
```

5.2 Evaluate The Results;

The accuracy of the model is determined by the R-squared values obtained from the algorithm.

1) The code y_pred = reg_rf.predict(X_test) is used to make predictions on the test data using the trained RandomForestRegressor model. After training the random forest model using the fit method, you can use the predict method to generate predictions on new, unseen data. In this case, the test data represented by X_test is passed as an argument to the predict method.

```
y_pred = reg_rf.predict(X_test)
```

2) The code reg_rf.score(X_train, y_train) and reg_rf.score(X_test, y_test) is used to calculate the R-squared score of the RandomForestRegressor model on the training data. The R-squared score ranges from 0 to 1, with a higher score indicating a better fit between the predicted values and the actual values.

In this code snippet, X_train and X_test represents the input features of the training data, and y_train and y_test represents the corresponding target values. By calling reg_rf.score(X_train, y_train) and reg_rf.score(X_test , y_train)the R-squared score of the random forest model on the training data is calculated and returned as output.

```
reg_rf.score(X_train, y_train)
```

```
0.953673858354006
```

```
reg_rf.score(X_test, y_test)
```

```
0.7968217572774429
```

5.3 Saving The Model:

(i) import pickle: This line imports the pickle module, which provides functionality for serializing and deserializing Python objects.

(ii) file = open('flight_rf.pkl', 'wb'): This line opens a file named flight_rf.pkl in write binary mode ('wb'). This is the file where you want to store the data.

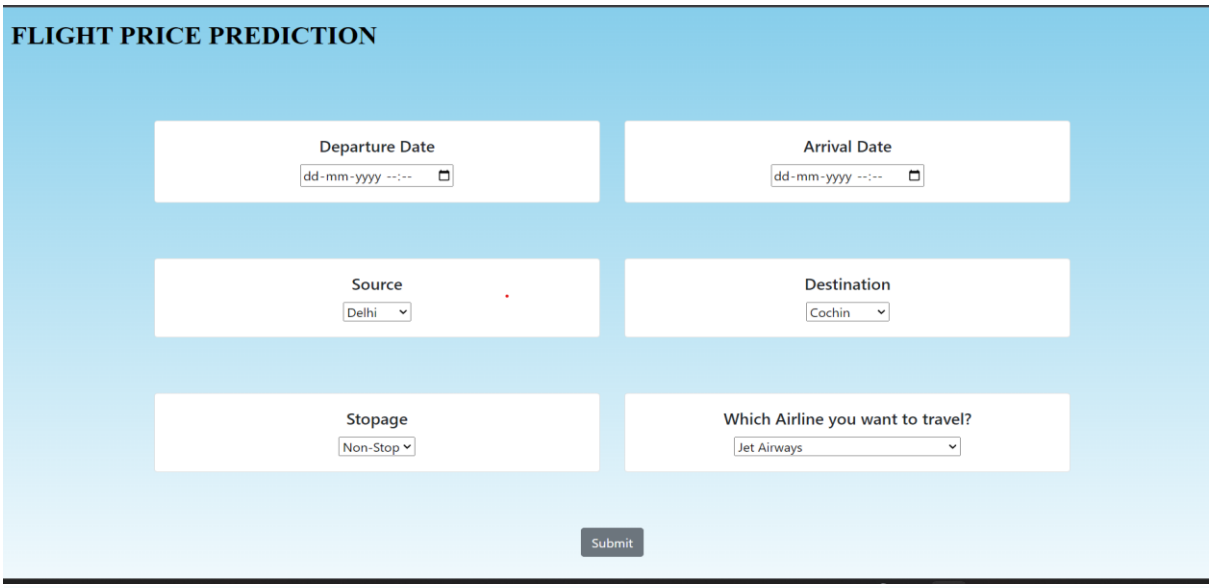
(iii) pickle.dump(reg_rf, file): This line uses the pickle.dump() function to serialize and save the reg_rf object (which represents your machine learning model) to the file flight_rf.pkl. The pickle.dump() function writes the serialized object to the file.

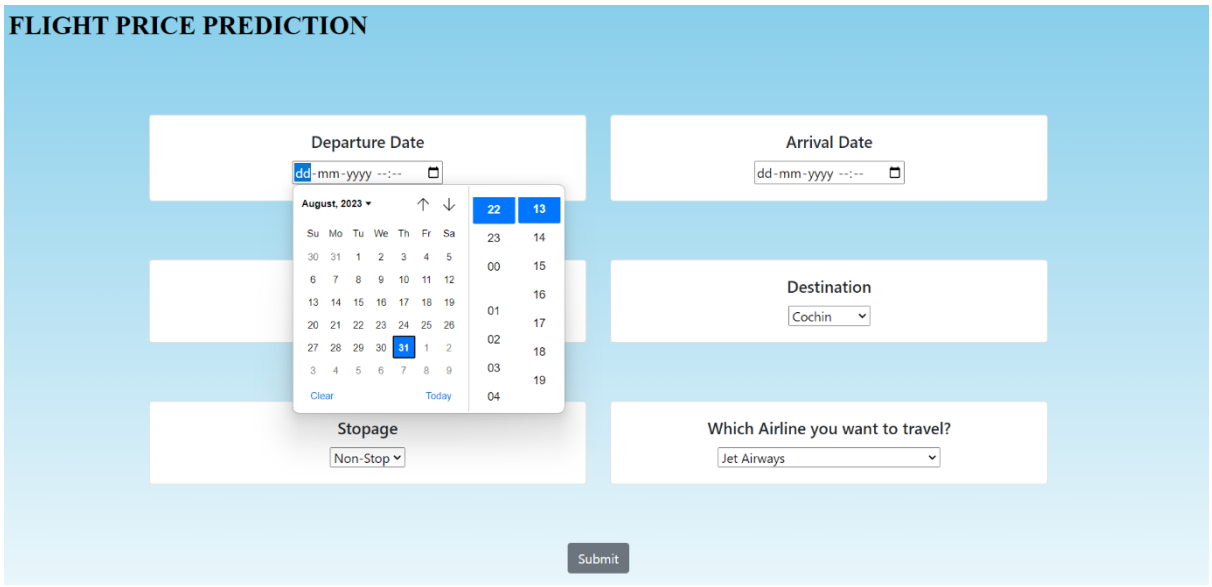
```
import pickle
# open a file, where you ant to store the data
file = open('flight_rf.pkl', 'wb')

# dump information to that file
pickle.dump(reg_rf, file)
```

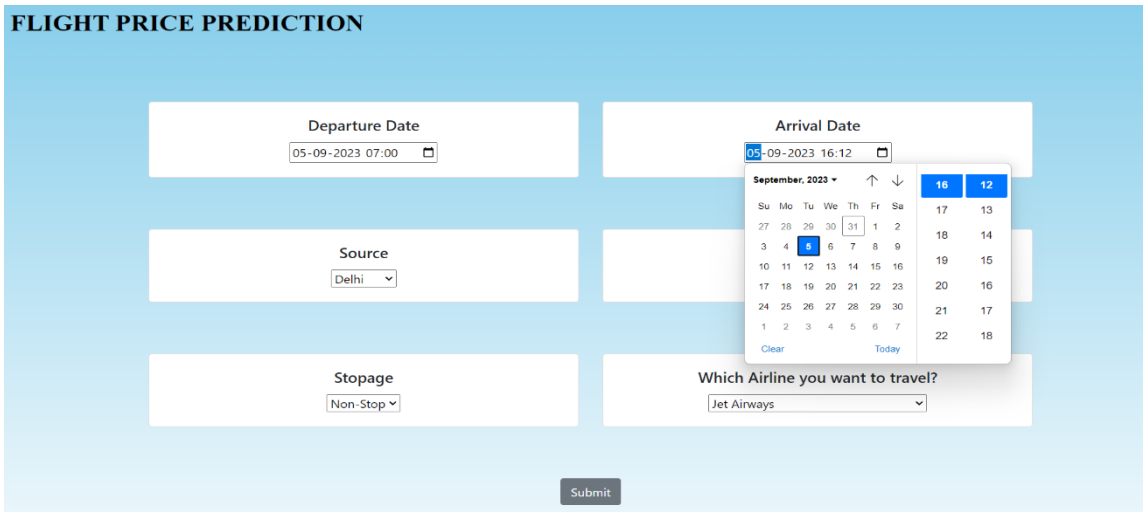
CHAPTER 6

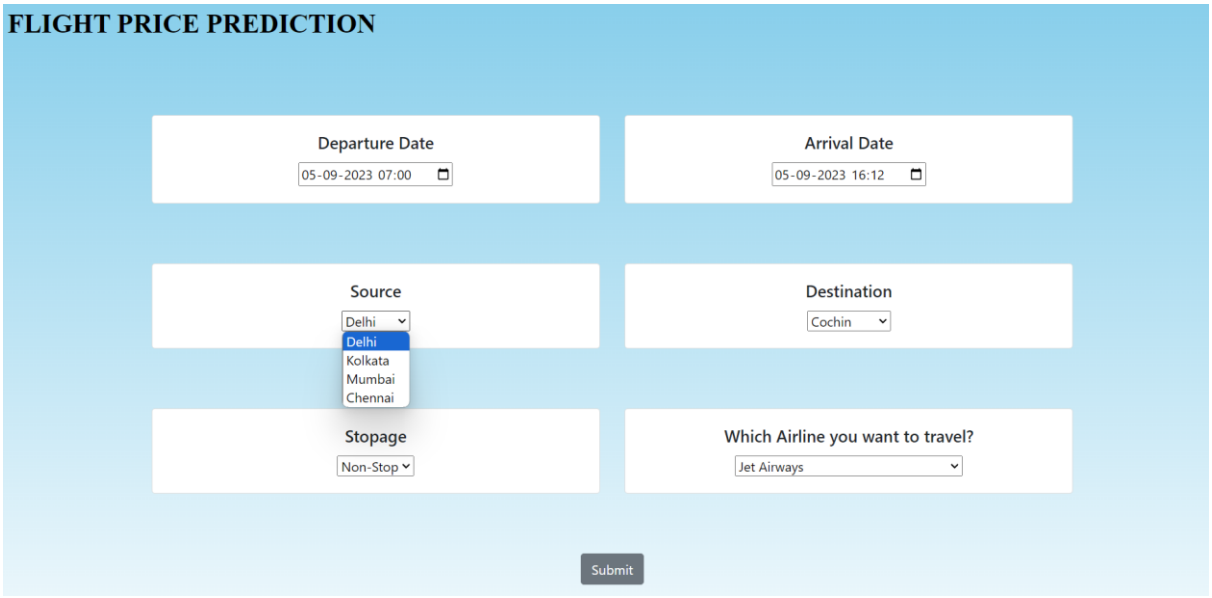
Application Building:

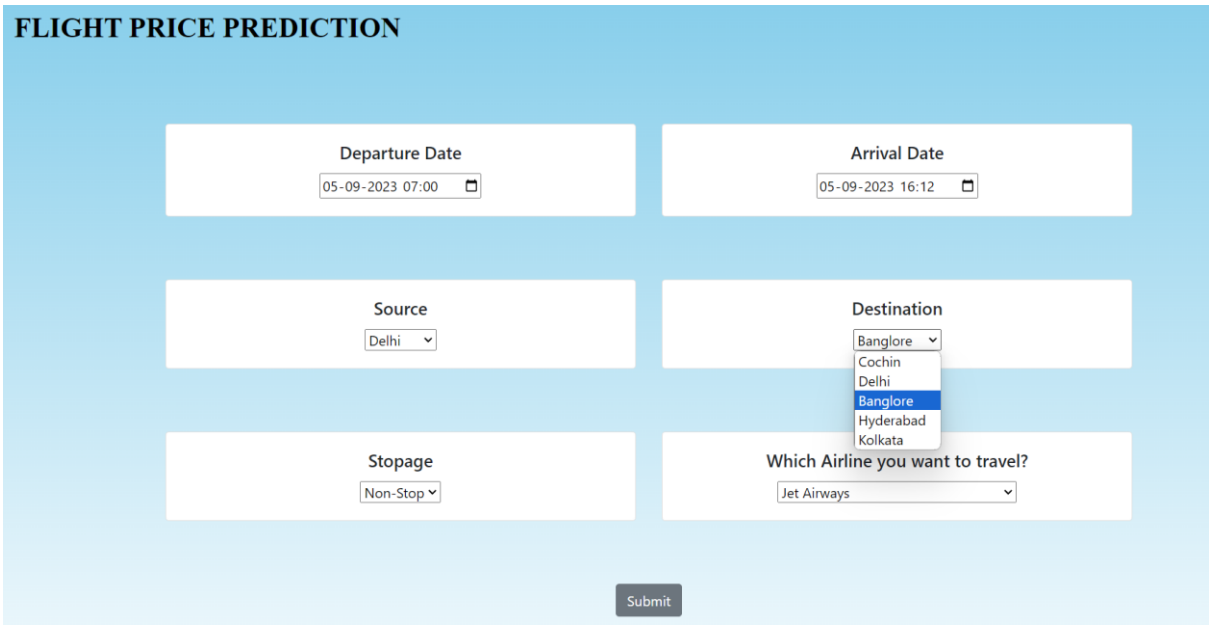
6.1 Build HTML Pages:

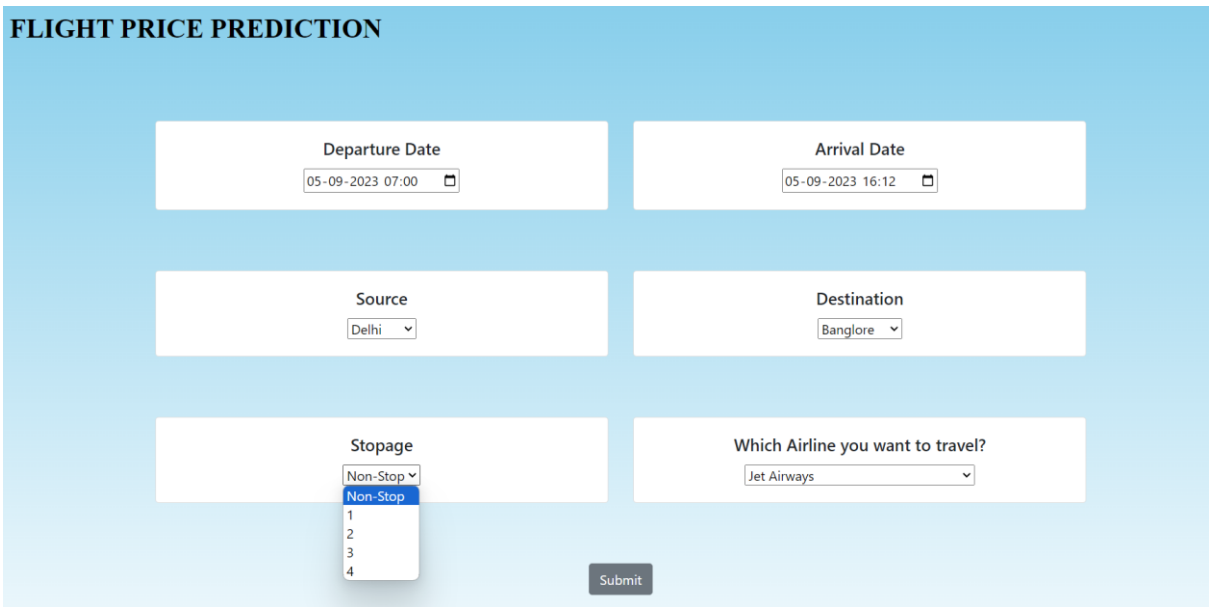


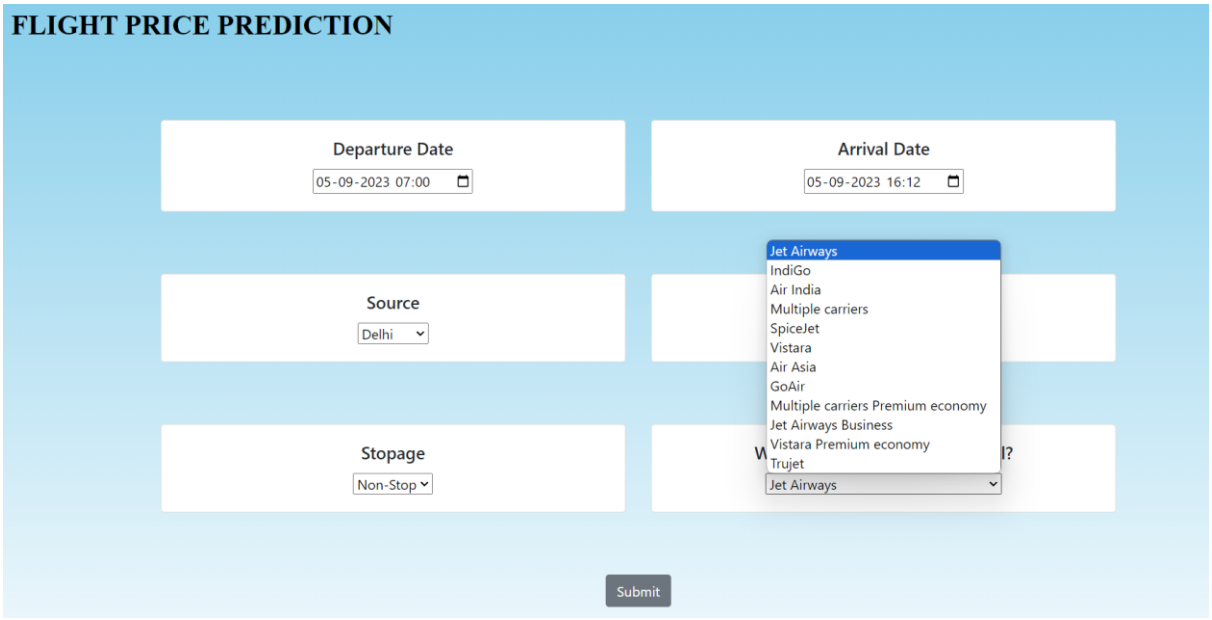





FLIGHT PRICE PREDICTION
Departure Date
05-09-2023 07:00
Arrival Date
05-09-2023 16:12
Source
Delhi
Delhi
Kolkata
Mumbai
Chennai
Destination
Cochin
Stopage
Non-Stop
Which Airline you want to travel?
Jet Airways
Submit


FLIGHT PRICE PREDICTION
Departure Date
05-09-2023 07:00
Arrival Date
05-09-2023 16:12
Source
Delhi
Destination
Banglore
Cochin
Delhi
Banglore
Hyderabad
Kolkata
Stopage
Non-Stop
Which Airline you want to travel?
Jet Airways
Submit


FLIGHT PRICE PREDICTION
Departure Date
05-09-2023 07:00
Arrival Date
05-09-2023 16:12
Source
Delhi
Destination
Banglore
Stopage
Non-Stop
Non-Stop
1
2
3
4
Which Airline you want to travel?
Jet Airways
Submit



6.2 Build Python Code:

```
from flask import Flask, request, render_template
from flask_cors import cross_origin
import sklearn
import pickle
import pandas as pd

app = Flask(__name__)
model = pickle.load(open("flight_rf.pkl", "rb"))



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




@app.route("/predict", methods = ["GET", "POST"])
@cross_origin()
def predict():
    if request.method == "POST":

        # Date_of_Journey
        date_dep = request.form["Dep_Time"]
        Journey_day = int(pd.to_datetime(date_dep, format="%Y-%m-
%dT%H:%M").day)
        Journey_month = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").month)
```

```
        # print("Journey Date : ",Journey_day, Journey_month)

        # Departure
        Dep_hour = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").hour)
        Dep_min = int(pd.to_datetime(date_dep, format ="%Y-%m-
%dT%H:%M").minute)
        # print("Departure : ",Dep_hour, Dep_min)

        # Arrival
        date_arr = request.form["Arrival_Time"]
        Arrival_hour = int(pd.to_datetime(date_arr, format ="%Y-%m-
%dT%H:%M").hour)
        Arrival_min = int(pd.to_datetime(date_arr, format ="%Y-%m-
%dT%H:%M").minute)
        # print("Arrival : ", Arrival_hour, Arrival_min)

        # Duration
        dur_hour = abs(Arrival_hour - Dep_hour)
        dur_min = abs(Arrival_min - Dep_min)
        # print("Duration : ", dur_hour, dur_min)

        # Total Stops
        Total_stops = int(request.form["stops"])
        # print(Total_stops)

        # Airline
        # AIR ASIA = 0 (not in column)
        airline=request.form['airline']
        if(airline=='Jet Airways'):
            Jet_Airways = 1
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='IndiGo'):
            Jet_Airways = 0
            IndiGo = 1
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
```

```
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Air India'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 1
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Multiple carriers'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 1
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='SpiceJet'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 1
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Vistara'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
```

```
            Vistara = 1
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='GoAir'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 1
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Multiple carriers Premium economy'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 1
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Jet Airways Business'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 1
            Vistara_Premium_economy = 0
            Trujet = 0

        elif (airline=='Vistara Premium economy'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
```

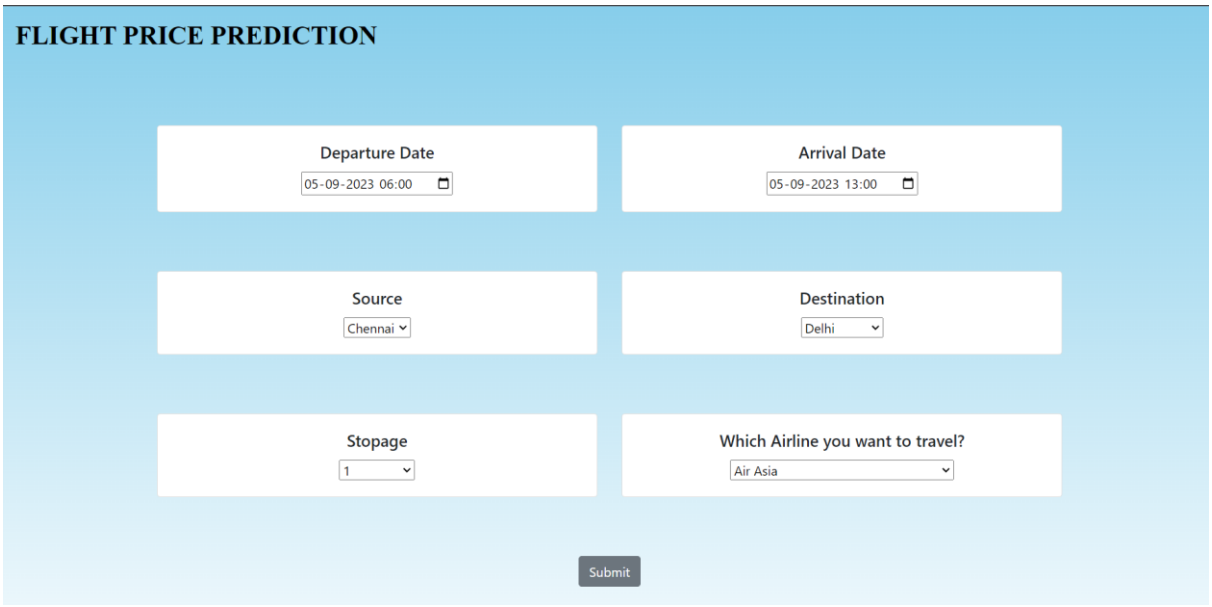
```
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 1
            Trujet = 0

        elif (airline=='Trujet'):
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 1

        else:
            Jet_Airways = 0
            IndiGo = 0
            Air_India = 0
            Multiple_carriers = 0
            SpiceJet = 0
            Vistara = 0
            GoAir = 0
            Multiple_carriers_Premium_economy = 0
            Jet_Airways_Business = 0
            Vistara_Premium_economy = 0
            Trujet = 0

        # print(Jet_Airways,
        #     IndiGo,
        #     Air_India,
        #     Multiple_carriers,
        #     SpiceJet,
        #     Vistara,
        #     GoAir,
        #     Multiple_carriers_Premium_economy,
        #     Jet_Airways_Business,
        #     Vistara_Premium_economy,
        #     Trujet)

        # Source
        # Banglore = 0 (not in column)
        Source = request.form["Source"]
```

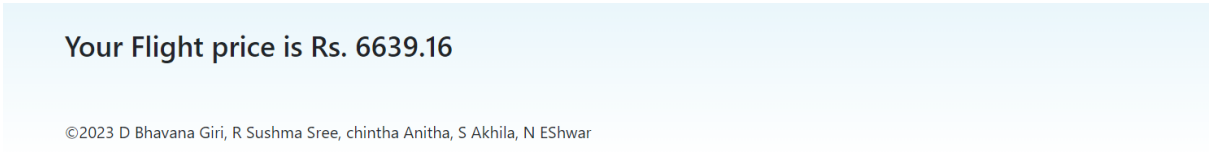
```
        if (Source == 'Delhi'):
            s_Delhi = 1
            s_Kolkata = 0
            s_Mumbai = 0
            s_Chennai = 0

        elif (Source == 'Kolkata'):
            s_Delhi = 0
            s_Kolkata = 1
            s_Mumbai = 0
            s_Chennai = 0

        elif (Source == 'Mumbai'):
            s_Delhi = 0
            s_Kolkata = 0
            s_Mumbai = 1
            s_Chennai = 0

        elif (Source == 'Chennai'):
            s_Delhi = 0
            s_Kolkata = 0
            s_Mumbai = 0
            s_Chennai = 1

        else:
            s_Delhi = 0
            s_Kolkata = 0
            s_Mumbai = 0
            s_Chennai = 0

        # print(s_Delhi,
        #     s_Kolkata,
        #     s_Mumbai,
        #     s_Chennai)

        # Destination
        # Banglore = 0 (not in column)
        Source = request.form["Destination"]
        if (Source == 'Cochin'):
            d_Cochin = 1
            d_Delhi = 0
            d_New_Delhi = 0
            d_Hyderabad = 0
            d_Kolkata = 0

        elif (Source == 'Delhi'):
            d_Cochin = 0
            d_Delhi = 1
            d_New_Delhi = 0
            d_Hyderabad = 0
```

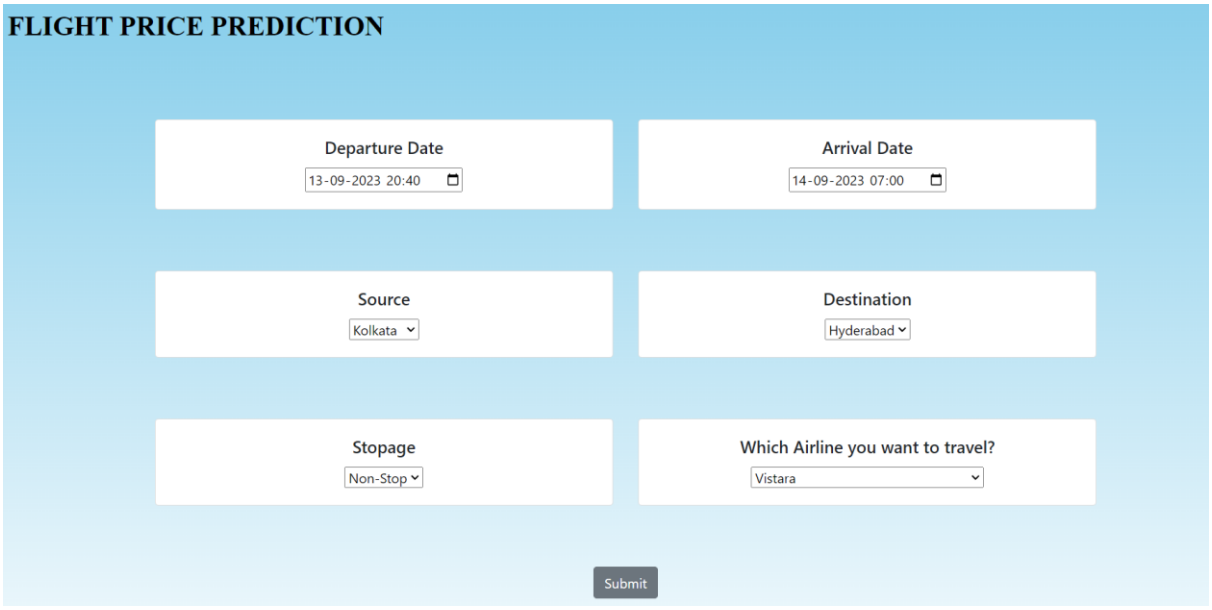
```
            d_Kolkata = 0

        elif (Source == 'New_Delhi'):
            d_Cochin = 0
            d_Delhi = 0
            d_New_Delhi = 1
            d_Hyderabad = 0
            d_Kolkata = 0

        elif (Source == 'Hyderabad'):
            d_Cochin = 0
            d_Delhi = 0
            d_New_Delhi = 0
            d_Hyderabad = 1
            d_Kolkata = 0

        elif (Source == 'Kolkata'):
            d_Cochin = 0
            d_Delhi = 0
            d_New_Delhi = 0
            d_Hyderabad = 0
            d_Kolkata = 1

        else:
            d_Cochin = 0
            d_Delhi = 0
            d_New_Delhi = 0
            d_Hyderabad = 0
            d_Kolkata = 0

        # print(
        #     d_Cochin,
        #     d_Delhi,
        #     d_New_Delhi,
        #     d_Hyderabad,
        #     d_Kolkata
        # )


    #     ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
    #    'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
    #    'Duration_mins', 'Airline_Air India', 'Airline_GoAir',
'Airline_IndiGo',
    #    'Airline_Jet Airways', 'Airline_Jet Airways Business',
    #    'Airline_Multiple carriers',
    #    'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
    #    'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium
economy',
    #    'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
    #    'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
```

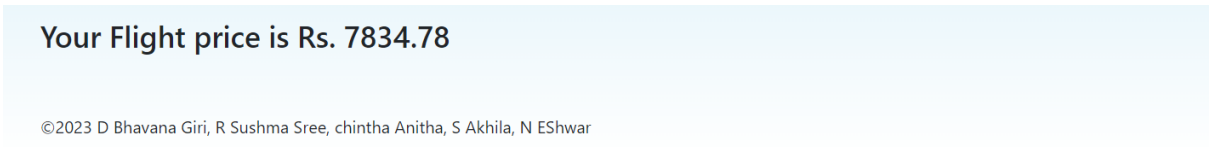
```
    #    'Destination_Kolkata', 'Destination_New Delhi']

        prediction=model.predict([[
            Total_stops,
            Journey_day,
            Journey_month,
            Dep_hour,
            Dep_min,
            Arrival_hour,
            Arrival_min,
            dur_hour,
            dur_min,
            Air_India,
            GoAir,
            IndiGo,
            Jet_Airways,
            Jet_Airways_Business,
            Multiple_carriers,
            Multiple_carriers_Premium_economy,
            SpiceJet,
            Trujet,
            Vistara,
            Vistara_Premium_economy,
            s_Chennai,
            s_Delhi,
            s_Kolkata,
            s_Mumbai,
            d_Cochin,
            d_Delhi,
            d_Hyderabad,
            d_Kolkata,
            d_New_Delhi
        ]])

        output=round(prediction[0],2)

        return render_template('home.html',prediction_text="Your Flight price
is Rs. {}".format(output))


    return render_template("home.html")



if __name__ == "__main__":
    app.run(debug=True)
```

6.3 Run The Web Application:









CHAPTER 7

Conclusion:

In conclusion, optimizing flight booking decisions through machine learning price prediction can greatly benefit travelers by providing them with valuable insights and helping them make informed choices. By leveraging historical data, machine learning algorithms can analyze various factors such as flight routes, dates, and demand patterns to predict future prices accurately. This empowers travelers to identify the best time to book flights, potentially saving them money and improving their overall travel experience.

Machine learning price prediction models can also consider external factors like holidays, events, and market trends, providing a comprehensive analysis of flight prices. By utilizing these predictions, travelers can take advantage of price fluctuations, secure the most affordable fares, and avoid overpaying for their flights.

Furthermore, machine learning algorithms continuously learn and adapt, improving their accuracy over time. This means that as more data is collected and analyzed, the predictions become more reliable, enabling travelers to make even better decisions.

Overall, optimizing flight booking decisions through machine learning price prediction offers a powerful tool for travelers to make cost-effective and efficient choices, enhancing their travel planning process and ultimately leading to a more enjoyable travel experience.

Future Scope:

The future scope for optimizing flight booking decisions through machine learning price prediction is promising. Here are a few potential areas of development:

1. Enhanced Personalization: Machine learning algorithms can be further refined to provide personalized flight recommendations based on individual preferences, travel history, and budget constraints. By understanding each traveler's unique needs, the algorithms can suggest tailored flight options, taking into account factors such as preferred airlines, layovers, and seat preferences.

2. Real-time Price Updates: Integrating real-time data feeds into machine learning models can enable travelers to receive up-to-the-minute price updates and notifications. This can help them take advantage of sudden price drops or limited-time offers, allowing for more flexible and dynamic flight booking decisions.

3. Multi-modal Travel Optimization: Machine learning algorithms can be expanded to optimize not only flights but also other modes of transportation, such as trains, buses, and rental cars. By considering the entire travel itinerary, these algorithms can suggest the most cost-effective and efficient combination of transportation options, taking into account factors like travel time, cost, and convenience.

4. Integration with Travel Planning Platforms: Machine learning price prediction models can be integrated with existing travel planning platforms and mobile applications. This would allow travelers to access price predictions and make bookings directly within these platforms, streamlining the entire travel planning process and providing a seamless user experience.

Overall, the future scope for optimizing flight booking decisions through machine learning price prediction is vast. As technology advances and more data becomes available, these models have the potential to revolutionize the way we plan and book our flights, making travel more affordable, convenient, and personalized for everyone.

References:

[1] K. Tziridis T. Kalampokas G.Papakostas and K.Diamantaras "Airfare price prediction using machine learning techniques" in European Signal Processing Conference (EUSIPCO), DOI: 10.23919/EUSIPCO .2017.8081365L. Li Y. Chen and Z. Li" Yawning detection for monitoring driver fatigue based on two cameras" Proc. 12th Int. IEEE Conf. Intel. Transp. Syst. pp. 1-6 Oct. 2009.

[2] Ren, Ruixuan, Yunzhe Yang, and Shenli Yuan. "Prediction of airline ticket price." University of Stanford (2014).

[3] Groves, William, and Maria Gini. "An agent for optimizing airline ticket purchasing." Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems. 2013.

[4] Rajankar, Supriya, and Neha Sakharkar. "A Survey on Flight Pricing Prediction using MachineLearning." Internatıonal Journal Of Engıneerıng Research & Technology (Ijert) 8.6 (2019): 1281-1284.

[5] Boruah, Abhijit, et al. "A Bayesian Approach for Flight Fare Prediction Based on Kalman Filter." Progress in Advanced Computing and Intelligent Engineering. Springer, Singapore, 2019. 191-203.

[6] Jaywrat Singh Champawat, Udhhav Arora, Dr. K. Vijaya, "INDIAN FLIGHT FARE PREDICTION: A PROPOSAL", International Journal of Advanced Technology in Engineering and Science, Vol. No.09, Issue No.03, ISSN 2348-7550, March 2021.

[7] Pavithra Maria K, Anitha K L, "Flight Price Prediction for Users by Machine Learning Techniques", International Advanced Research Journal in Science, Engineering and Technology, Vol.8, Issue 3, DOI: 10.17148/IARJSET.2021.8321, March 2021.

[8] Rutuja Konde, Rutuja Somvanshi, Pratiksha Khaire, Prachi Zende, Kamlesh Patil, "Airfare Price Prediction System", International Research Journal of Engineering and Technology (IRJET), Volume: 09 Issue: 05, May 2022.

[9] Viet Hong Vu, Phu Phung,An airfare Prediction model for developing Markets,IEEE 2018.

[10] T. Janssen, A linear quantile mixed regression model for prediction of airline ticket prices, Bachelor Thesis, Radboud University, 2014.