Optimizing Flight Booking Decisions through Machine Learning Price Predictions

ABSTRACT

Now a day's airline corporation are using complex strategies and methods to assign airfare prices in a dynamic fashion. High complex pricing algorithms are used by airlines due to this, it becomes difficult fto a customer to buy an air ticket at the lowest cost, since the price changes dynamically. Optimal timing for airline ticket purchasing from the consumers perspective is challenging. Buyers have insufficient information for reasoning about price movements. For this reason several prediction algorithms from Machine Learning are used for air price prediction. This project will help a travelers to decide a specific airline as per their budget.

I. INTRODUCTION

The flight ticket buying system framework is to buy a ticket numerous days earlier to flight takeoff so as to remain absent from the impact of the foremost extraordinary charge. For the most part, flying courses don't concur this strategy. Plane organizations may reduce the taken a toll at the time, they ought to construct the showcase and at the time when the tickets are less open. They may maximize the costs. So the fetched may depend upon distinctive variables. To anticipate the costs this wander employments machine learning show the ways of flight tickets after a few time. All organizations have benefits and opportunity to alter its ticket costs at anytime. Pilgrim can set aside cash by booking a ticket at the slightest costs. Individual who had voyage by flight habitually aremindful of cost changes. The aircrafts utilize complex approaches of income administration for execution of particular assessing frameworks. The assessing framework as a result changes the charge depending on time, season and merry days to alter the header or footer.

II. LITERATURE SURVEY

It is very hard to find a flight ticket at reduced cost as the price of a ticket depends on various factors. The companies use very complex algorithms to fix a flight ticket price. There are various machine learning algorithms which can be used to predict a lowest price considering previous travelling data. We collected the data from kaggle website. There are two datasets namely training data set and testing dataset. The data is based on 1 year travelling data of people travelling in flights. The reference journal that we referred consists of various algorithms like linear regression, support vector machine. We improvised the project by using many feature engineering techniques like one hot encoding. We choose to use random forest regressor to train the model. We are able to improve the efficiency of the trained model.

Data Collection

The assortment of data is the very first step in machine learning projects. There are various sources of data available on numerous websites that are deployed to construct the models. These sites supply a huge variety of data regarding different airlines, routes, times, and tolls. In this part, data gathered from the various available sources are studied. For the execution of this, information is brought from a site called Kaggle. For the assortment of the data and to execute the model's Python is utilized [8-15]. The dataset collected contains information about different airlines in India. It consists of various factors which affect the price of a flight ticket including the price for a particular flight. It contains 10683 rows of data. The features present in the dataset are the name of companies, Date of travelling, Origin, terminus, path of travelling, Time of Departure, Time of Arrival, Travelling Hours, Total Stoppage, Additional Info, and Price. 3. Cleaning and Preparing of Data Cleaning and preparing data are a very important step in machine learning. The data collected can't be used raw as it may contain certain parameters which would be of no use and also certain data can't be used the way it would be present in the dataset. So, before proceeding to the actual work, the data needs to be filtered and it should be absolutely clean. For achieving this, all the duplicate and null values are removed from the dataset and specific data is converted to a usable format. 4. Machine Learning Techniques Various conventional machine learning algorithms are used for creating a model for flight fare prediction which is ANN, LR, DT, and RF. These loads of machine learning techniques are executed using the sci-kit-learn library available in python. For assessing the exhibition of these algorithms, definite boundaries are thought of. These are mentioned as follows: MAPE (Mean Absolute Percentage Error) and RMSE (Root Mean Square Error). 4.1 RMSE RMSE is a tool that helps in determining how accurately the model is making the predictions. It calculates how much error the model creates while making these predictions. It measures the standard of predictions. Mathematically, it is defined as the square root of the average of the squares of all the errors. Error is defined as the difference between the actual and predicted value. Less the RMSE.

IMPLEMENTATION

We have followed following steps in our project to get to our ultimate goal of predicting flight fare:

1. *Importing Necessary Libraries*

Importing the python libraries such as pandas, matplotlib, seaborn, NumPy for reading and visualizing the dataset.

2. Reading our Dataset

We will read out dataset using pandas. As the dataset is in the excel form, we will use “pd.read_excel()”.

3. Dropping NAN Values

We will check if there are any Null values in our dataset, if we have, we will drop it using: “dropna(inplace=TRUE)”.

4. Exploratory Data Analysis

We will pre-process our dataset. We will extract day and month from the column “Date of Journey” as the model will understand numerical value, for this we will use “pd.to_datetime” for day and month column. “dt.day” and “dt.month” will extract day and month respectively from the given column.

Same process will be doing for the “dep_time” column, “Duration” column and “arrival_time” column and extract hours and min from it. After extracting day, month, hours and min, we will drop “Date of Journey”, “Duration”, “dep_time” & “arrival_time” column from our dataset.

5. Handling Categorical Data

As we know the model understands numerical value, so we will convert all the categorical data into numerical data. For this we will perform “OneHotEncoding” method to convert it to numerical data. We will make dummies using pandas and perform “OneHotEncoding” on the “Airline”, “Source” and “Destination” columns.

We will drop “AdditionalInfo” and “Route” columns as “Route” column contains same data as “Total_Stops” columns and “AdditionalInfo” column doesn’t have any additional info. “Total_Stops” column is ordinal type data so we will perform “LabelEncoder” and label each stop as 0,1,2,3,4. As the stop increases, the value also increases.

6. Test Data: Performing EDA and Feature Engineering

For the test data, we will perform same steps followed in step (2), (3), (4) and (5).

7. Feature Selection

In this process, we will find out the best feature which will contribute to our target variable.

X = “Independent Feature”

Y = “Dependent Feature” i.e., “Price” column.

We will separate all the independent features except price in the X variable and price in Y variable. For this, we will use loc & iloc method.

Now, we have used “ExtraTreesRegressor” to find more important features from the data. Use the selection variable and do fitting the X & Y features. After this we will print “feature_importance” and will get to know the important features.

Importing Libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
from math import sqrt
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from prettytable import PrettyTable
```

Reading the training data of our dataset

Exploratory Data Analysis (EDA)

Now here we will be looking at the kind of columns our dataset has.

```
train_df.columns
```

Output:

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination', 
'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

Here we can get more information about our dataset

```
train_df.info()
```

Output:

```
<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
```

To know more about the dataset

```
train_df.describe()
```

Output:

	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

```
rain_df.isnull().head()
```

Output:

Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False

Now while using the IsNull function and sum function we will gonna see the number of null values in our dataset

```
train_df.isnull().sum()
```

Output:

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

Dropping NAN values

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

Duplicate values

```
train_df[train_df.duplicated()].head()
```

Output:

Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
Vistara	Banglore	New Delhi	BLR → DEL	175	non-stop	No info	7608	3	3	21	10	0	5
Air Asia	Banglore	New Delhi	BLR → DEL	165	non-stop	No info	4482	24	3	23	25	2	10

Here we will be removing those repeated values from the dataset and keeping the in-place attribute to be true so that there will be no changes.

```
train_df.drop_duplicates(keep='first',inplace=True)
train_df.head()
```

Output:

	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

```
train_df.shape
```

Output:

```
(10462, 11)
```

Checking the Additional_info column and having the count of unique types of values.

```
train_df["Additional_Info"].value_counts()
```

Output:

```
No info                          8182
```

```
In-flight meal not included     1926
No check-in baggage included     318
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
2 Long layover                     1
Red-eye flight                     1
Name: Additional_Info, dtype: int64
```

Checking the different Airlines

```
train_df["Airline"].unique()
```

Output:

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

Checking the different Airline Routes

```
train_df["Route"].unique()
```

This article was published as a part of the Data Science Blogathon.

Overview

In this article, 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.

Image source: Kaggle

Takeaways from the blog

In this article, we do prediction using machine learning which leads to below takeaways:

1. **EDA:** Learn the complete process of EDA
2. **Data analysis:** Learn to withdraw some insights from the dataset both mathematically and visualize it.
3. **Data visualization:** Visualising the data to get better insight from it.
4. **Feature engineering:** We will also see what kind of stuff we can do in the feature engineering part.

About the dataset

1. **Airline:** So this column will have all the types of airlines like Indigo, Jet Airways, Air India, and many more.
2. **Date_of_Journey:** This column will let us know about the date on which the passenger's journey will start.
3. **Source:** This column holds the name of the place from where the passenger's journey will start.
4. **Destination:** This column holds the name of the place to where passengers wanted to travel.
5. **Route:** Here we can know about that what is the route through which passengers have opted to travel from his/her source to their destination.
6. **Arrival_Time:** Arrival time is when the passenger will reach his/her destination.
7. **Duration:** Duration is the whole period that a flight will take to complete its journey from source to destination.
8. **Total_Stops:** This will let us know in how many places flights will stop there for the flight in the whole journey.
9. **Additional_Info:** In this column, we will get information about food, kind of food, and other amenities.
10. **Price:** Price of the flight for a complete journey including all the expenses before onboarding.

Importing Libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
from math import sqrt
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import RandomizedSearchCV

from prettytable import PrettyTable
```

Exploratory Data Analysis (EDA)

Now here we will be looking at the kind of columns our dataset has.

```
train_df.columns
```

Output:

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination',
'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info', 'Price'],
      dtype='object')
```

Here we can get more information about our dataset

```
train_df.info()
```

Output:

```
<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
```

To know more about the dataset

```
train_df.describe()
```

Output:

	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

Now while using the IsNull function we will gonna see the number of null values in our dataset

```
train_df.isnull().head()
```

Output:

Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False
False	False	False	False	False	False	False	False	False	False	False	False	False	False

Now while using the IsNull function and sum function we will gonna see the number of null values in our dataset

```
train_df.isnull().sum()
```

Output:

```
Airline            0
Date_of_Journey    0
Source             0
Destination        0
Route              1
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        1
Additional_Info    0
Price              0
dtype: int64
```

Dropping NAN values

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

Duplicate values

```
train_df[train_df.duplicated()].head()
```

Output:

Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
Vistara	Banglore	New Delhi	BLR → DEL	175	non-stop	No info	7608	3	3	21	10	0	5
Air Asia	Banglore	New Delhi	BLR → DEL	165	non-stop	No info	4482	24	3	23	25	2	10

Here we will be removing those repeated values from the dataset and keeping the in-place attribute to be true so that there will be no changes.

```
train_df.drop_duplicates(keep='first',inplace=True)
train_df.head()
```

Output:

	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

```
train_df.shape
```

Output:

```
(10462, 11)
```

Checking the Additional_info column and having the count of unique types of values.

```
train_df["Additional_Info"].value_counts()
```

Output:

```
No info                          8182
In-flight meal not included      1926
No check-in baggage included      318
1 Long layover                     19
Change airports                     7
Business class                      4
No Info                             3
1 Short layover                     1
2 Long layover                      1
Red-eye flight                      1
Name: Additional_Info, dtype: int64
```

Checking the different Airlines

```
train_df["Airline"].unique()
```

Output:

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJet',
       'Multiple carriers', 'GoAir', 'Vistara', 'Air Asia',
       'Vistara Premium economy', 'Jet Airways Business',
       'Multiple carriers Premium economy', 'Trujet'], dtype=object)
```

Checking the different Airline Routes

```
train_df["Route"].unique()
```

Now let's look at our testing dataset

```
test_df = pd.read_excel("Test_set.xlsx")
test_df.head(10)
```

Output:

	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
5	Jet Airways	12/06/2019	Delhi	Cochin	DEL → BOM → COK	18:15	12:35 13 Jun	18h 20m	1 stop	In-flight meal not included
6	Air India	12/03/2019	Banglore	New Delhi	BLR → TRV → DEL	07:30	22:35	15h 5m	1 stop	No info
7	IndiGo	1/05/2019	Kolkata	Banglore	CCU → HYD → BLR	15:15	20:30	5h 15m	1 stop	No info
8	IndiGo	15/03/2019	Kolkata	Banglore	CCU → BLR	10:10	12:55	2h 45m	non-stop	No info
9	Jet Airways	18/05/2019	Kolkata	Banglore	CCU → BOM → BLR	16:30	22:35	6h 5m	1 stop	No info

Now here we will be looking at the kind of columns our testing data has.

```
test_df.columns
```

Output:

```
Index(['Airline', 'Date_of_Journey', 'Source', 'Destination',
'Route',
       'Dep_Time', 'Arrival_Time', 'Duration', 'Total_Stops',
       'Additional_Info'],
      dtype='object')
```

Information about the dataset

```
test_df.info()
```

Output:

```
<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
```

To know more about the testing dataset

```
test_df.describe()
```

Output:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
count	2671	2671	2671	2671	2671	2671	2671	2671	2671	2671
unique	11	44	5	6	100	199	704	320	5	6
top	Jet Airways	9/05/2019	Delhi	Cochin	DEL → BOM → COK	10:00	19:00	2h 50m	1 stop	No info
freq	897	144	1145	1145	624	62	113	122	1431	2148

Now while using the IsNull function and sum function we will gonna see the number of null values in our testing data

```
test_df.isnull().sum()
```

Output:

```
Airline            0
Date_of_Journey    0
Source             0
```

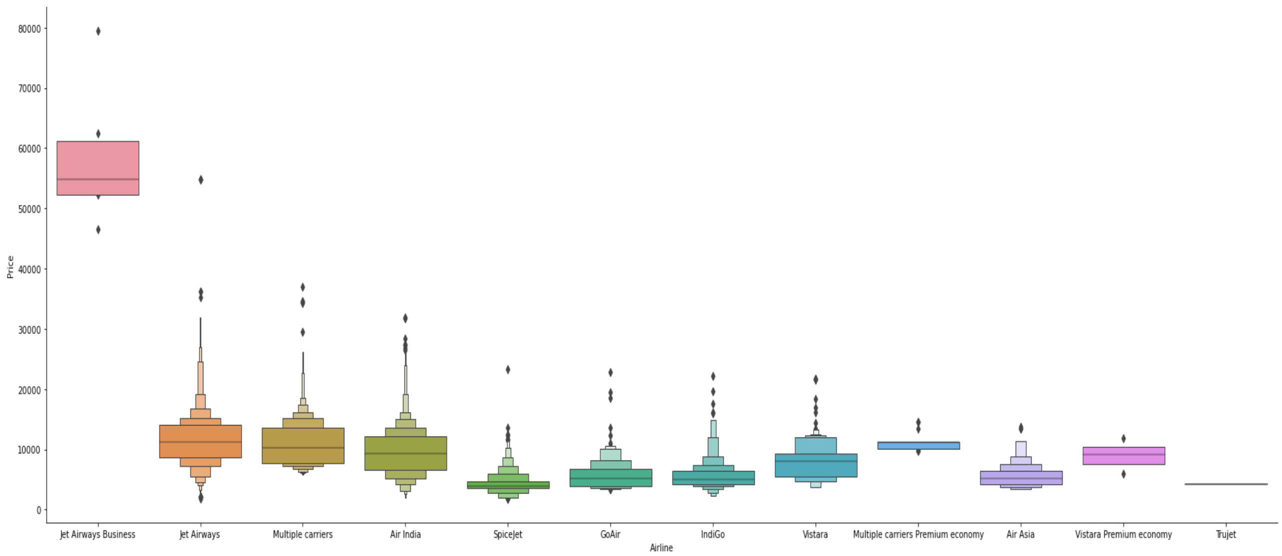
```
Destination        0
Route              0
Dep_Time           0
Arrival_Time       0
Duration           0
Total_Stops        0
Additional_Info    0
dtype: int64
```

Data Visualization

Plotting Price vs Airline plot

```
sns.catplot(y = "Price", x = "Airline", data =
train_df.sort_values("Price", ascending = False), kind="boxen",
height = 8, aspect = 3)
plt.show()
```

Output:

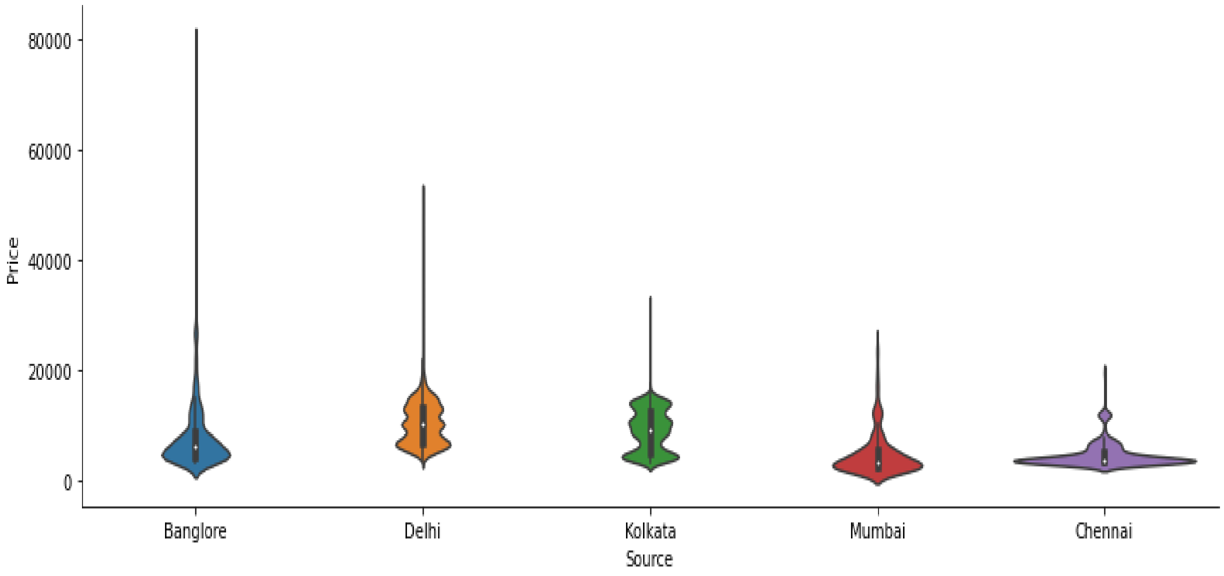


Inference: Here with the help of the cat plot we are trying to plot the boxplot between the price of the flight and airline and we can conclude that **Jet Airways has the most outliers in terms of price**.

Plotting Violin plot for Price vs Source

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

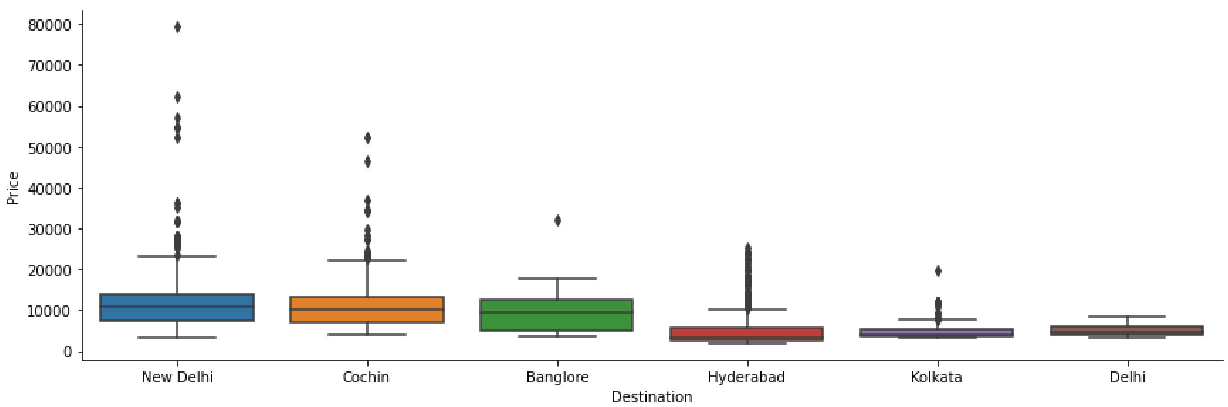
Output:



Plotting Box plot for Price vs Destination

```
sns.catplot(y = "Price", x = "Destination", data =
train_df.sort_values("Price", ascending = False), kind="box", height
= 4, aspect = 3)
plt.show()
```

Output:



Feature Engineering

Let’s see our processed data first

```
train_df.head()
```

Output:

	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

Here first we are dividing the features and labels and then converting the hours in minutes.

```
train_df['Duration'] = train_df['Duration'].str.replace("h",
'*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
test_df['Duration'] = test_df['Duration'].str.replace("h",
'*60').str.replace(' ','+').str.replace('m','*1').apply(eval)
```

Date_of_Journey: Here we are organizing the format of the date of journey in our dataset for better preprocessing in the model stage.

```
train_df["Journey_day"] =
train_df['Date_of_Journey'].str.split('/').str[0].astype(int)
train_df["Journey_month"] =
train_df['Date_of_Journey'].str.split('/').str[1].astype(int)
train_df.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

Dep_Time: Here we are converting departure time into hours and minutes

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

Arrival_Time: Similarly we are converting the arrival time into hours and minutes.

Now after final preprocessing let's see our dataset

```
train_df.head()
```

Output:

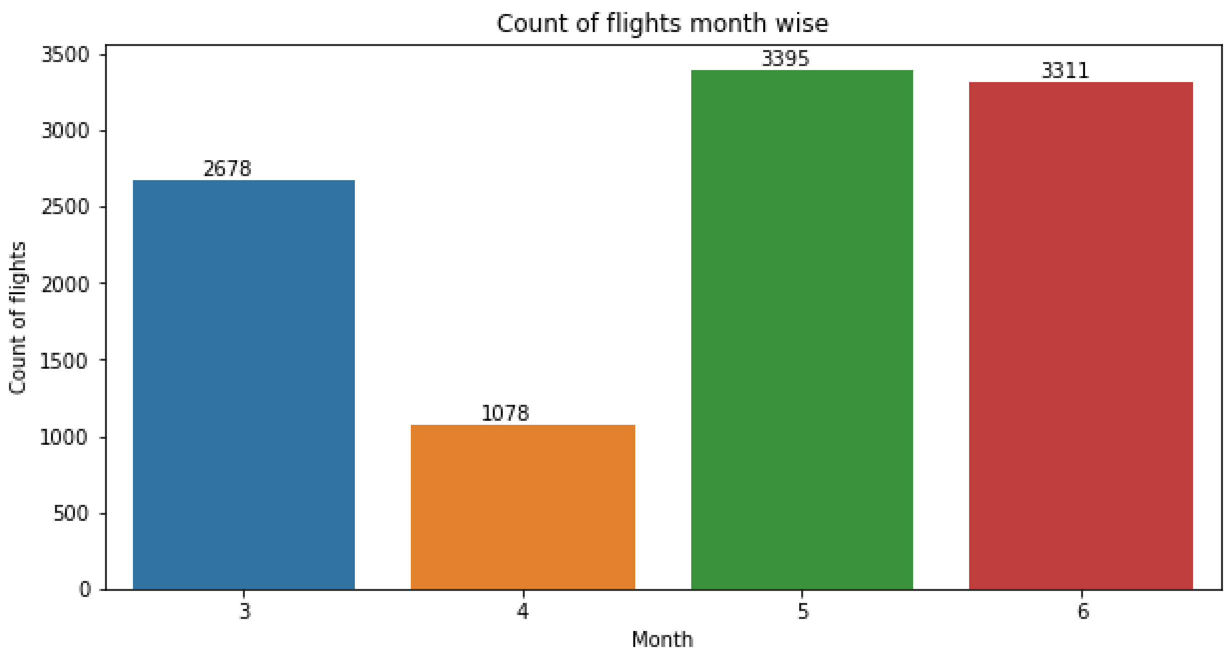
Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
IndiGo	Banglore	New Delhi	BLR → DEL	170	non-stop	No info	3897	24	3	22	20	1	10
Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	445	2 stops	No info	7662	1	5	5	50	13	15
Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	1140	2 stops	No info	13882	9	6	9	25	4	25
IndiGo	Kolkata	Banglore	CCU → NAG → BLR	325	1 stop	No info	6218	12	5	18	5	23	30
IndiGo	Banglore	New Delhi	BLR → NAG → DEL	285	1 stop	No info	13302	1	3	16	50	21	35

Plotting Bar chart for Months (Duration) vs Number of Flights

```
plt.figure(figsize = (10, 5))
```

```
plt.title('Count of flights month wise')
ax=sns.countplot(x = 'Journey_month', data = train_df)
plt.xlabel('Month')
plt.ylabel('Count of flights')
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, 
p.get_height()+1), va='bottom', color= 'black')
```

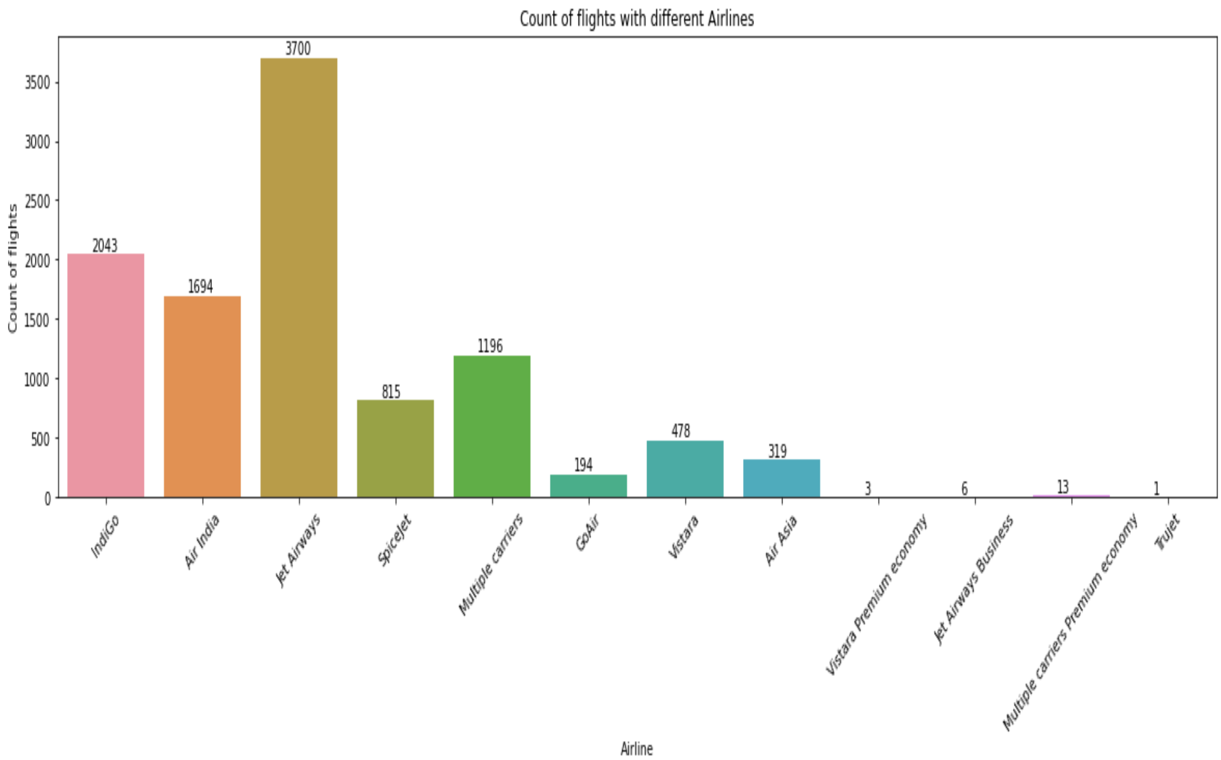
Output:



Plotting Bar chart for Types of Airline vs Number of Flights

```
plt.figure(figsize = (20,5))
plt.title('Count of flights with different Airlines')
ax=sns.countplot(x = 'Airline', data =train_df)
plt.xlabel('Airline')
plt.ylabel('Count of flights')
plt.xticks(rotation = 45)
for p in ax.patches:
    ax.annotate(int(p.get_height()), (p.get_x()+0.25, 
p.get_height()+1), va='bottom', color= 'black')
```

Output:



Plotting Ticket Prices VS Airlines

```
plt.figure(figsize = (15,4))
plt.title('Price VS Airlines')
plt.scatter(train_df['Airline'], train_df['Price'])
plt.xticks
plt.xlabel('Airline')
plt.ylabel('Price of ticket')
plt.xticks(rotation = 90)
```

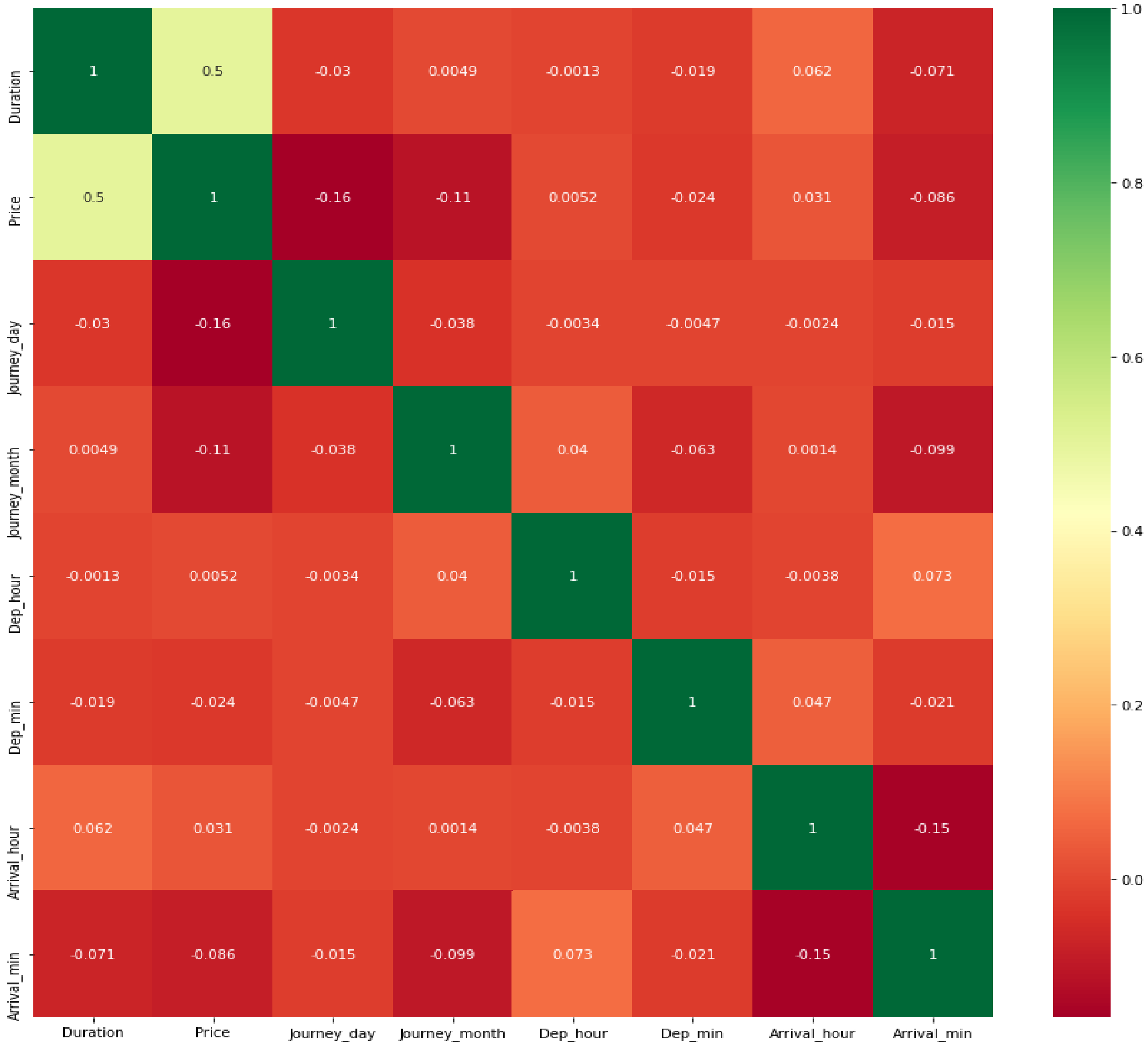
Output:

Correlation between all Features

Plotting Correlation

```
plt.figure(figsize = (15,15))
sns.heatmap(train_df.corr(), annot = True, cmap = "RdYlGn")
plt.show()
```

Output:



Dropping the Price column as it is of no use

```
data = train_df.drop(["Price"], axis=1)
```

Dealing with Categorical Data and Numerical Data

```
train_categorical_data = data.select_dtypes(exclude=['int64',
'float','int32'])
train_numerical_data = data.select_dtypes(include=['int64',
'float','int32'])

test_categorical_data = test_df.select_dtypes(exclude=['int64',
'float','int32','int32'])
test_numerical_data  = test_df.select_dtypes(include=['int64',
'float','int32'])
train_categorical_data.head()
```

Output:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info

Label Encode and Hot Encode for Categorical Columns

Output:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info
0	3	0	5	18	4	8
1	1	3	0	84	1	8
2	4	2	1	118	1	8
3	3	3	0	91	0	8
4	3	0	5	29	0	8

Concatenating both Categorical Data and Numerical Data

```
X = pd.concat([train_categorical_data, train_numerical_data],
axis=1)
y = train_df['Price']
test_set = pd.concat([test_categorical_data, test_numerical_data],
axis=1)
X.head()
```

Output:

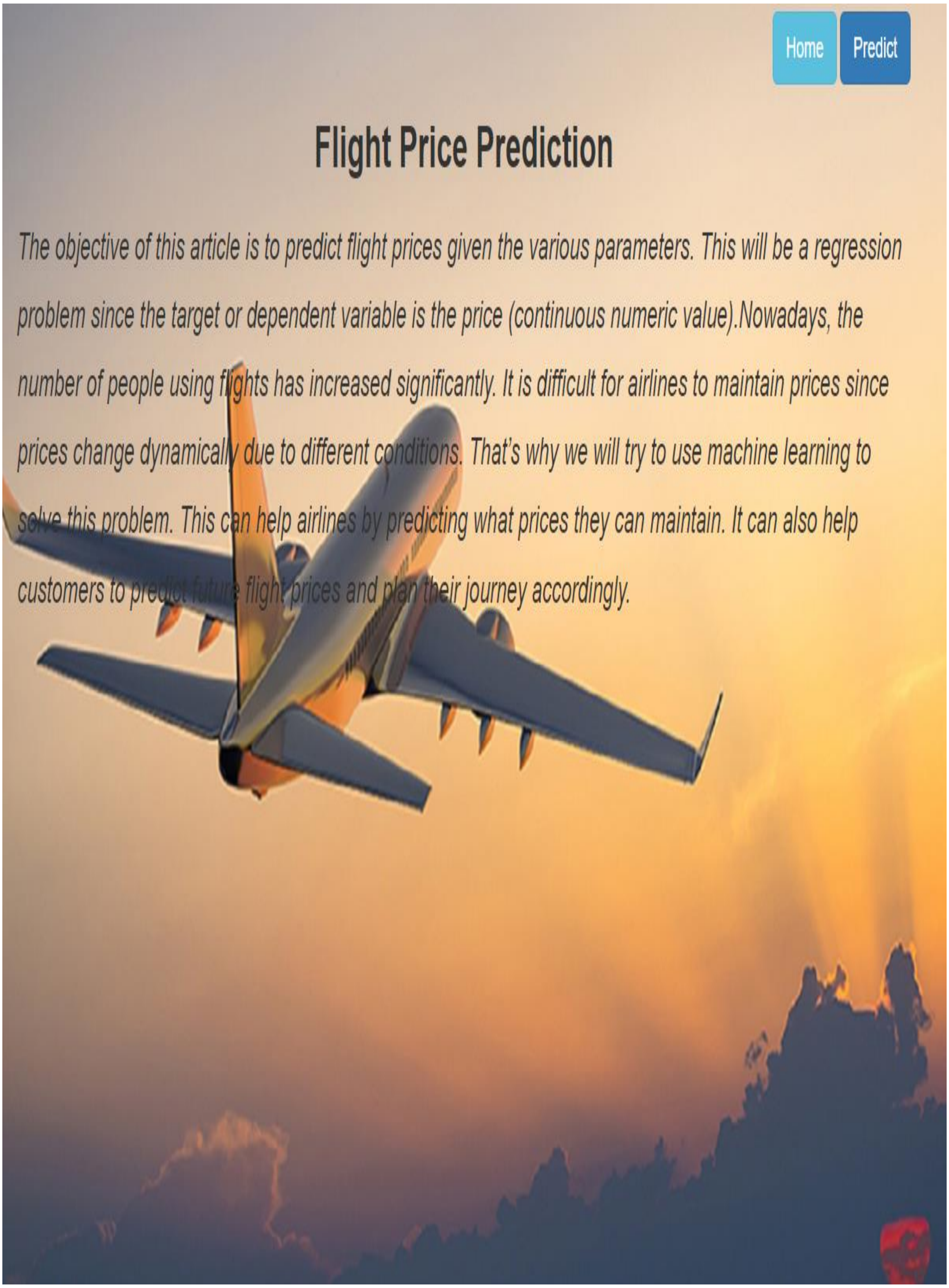
	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Duration	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
0	3	0	5	18	4	8	170	24	3	22	20	1	10
1	1	3	0	84	1	8	445	1	5	5	50	13	15
2	4	2	1	118	1	8	1140	9	6	9	25	4	25
3	3	3	0	91	0	8	325	12	5	18	5	23	30
4	3	0	5	29	0	8	285	1	3	16	50	21	35

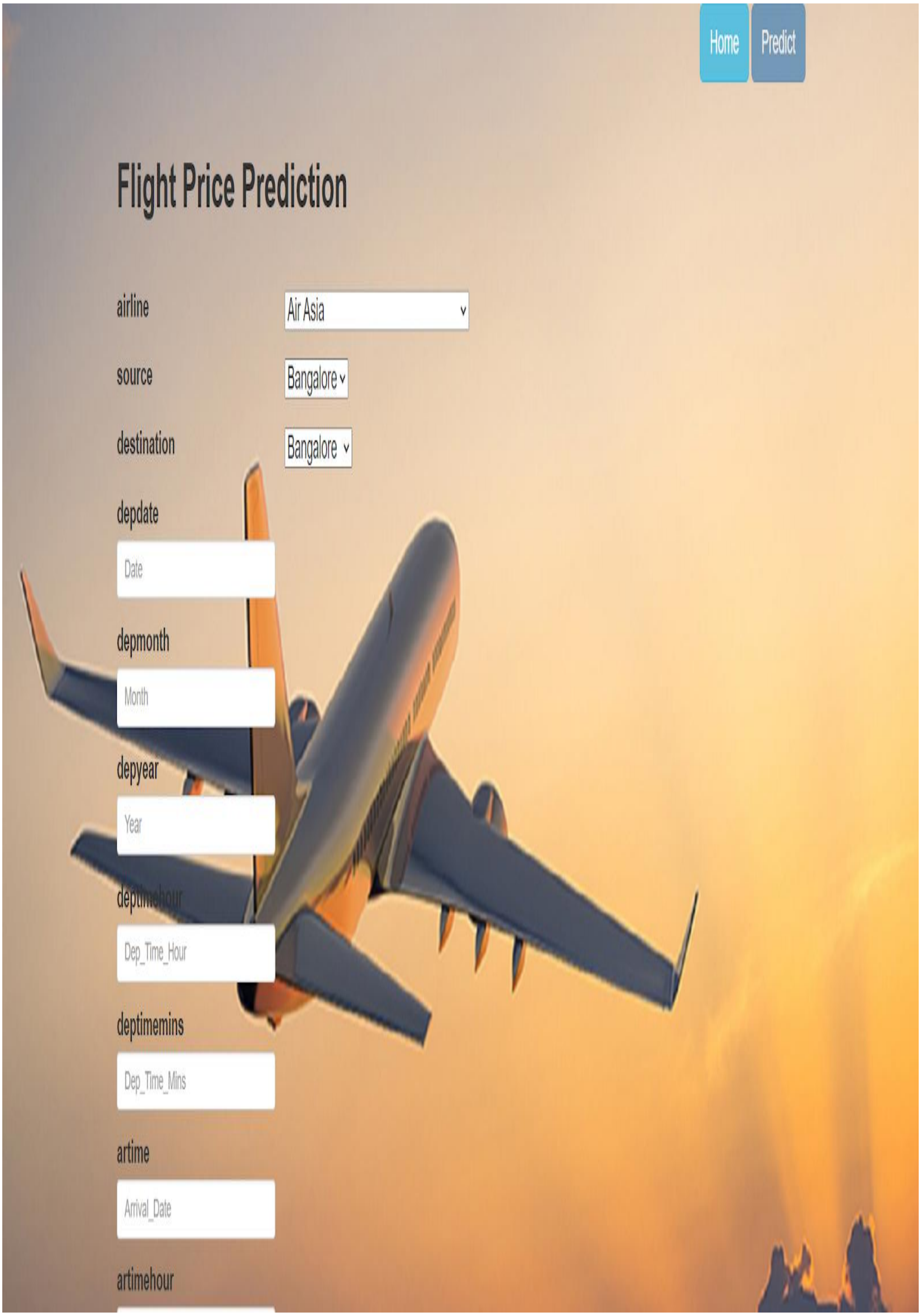
```
y.head()
```

Output:

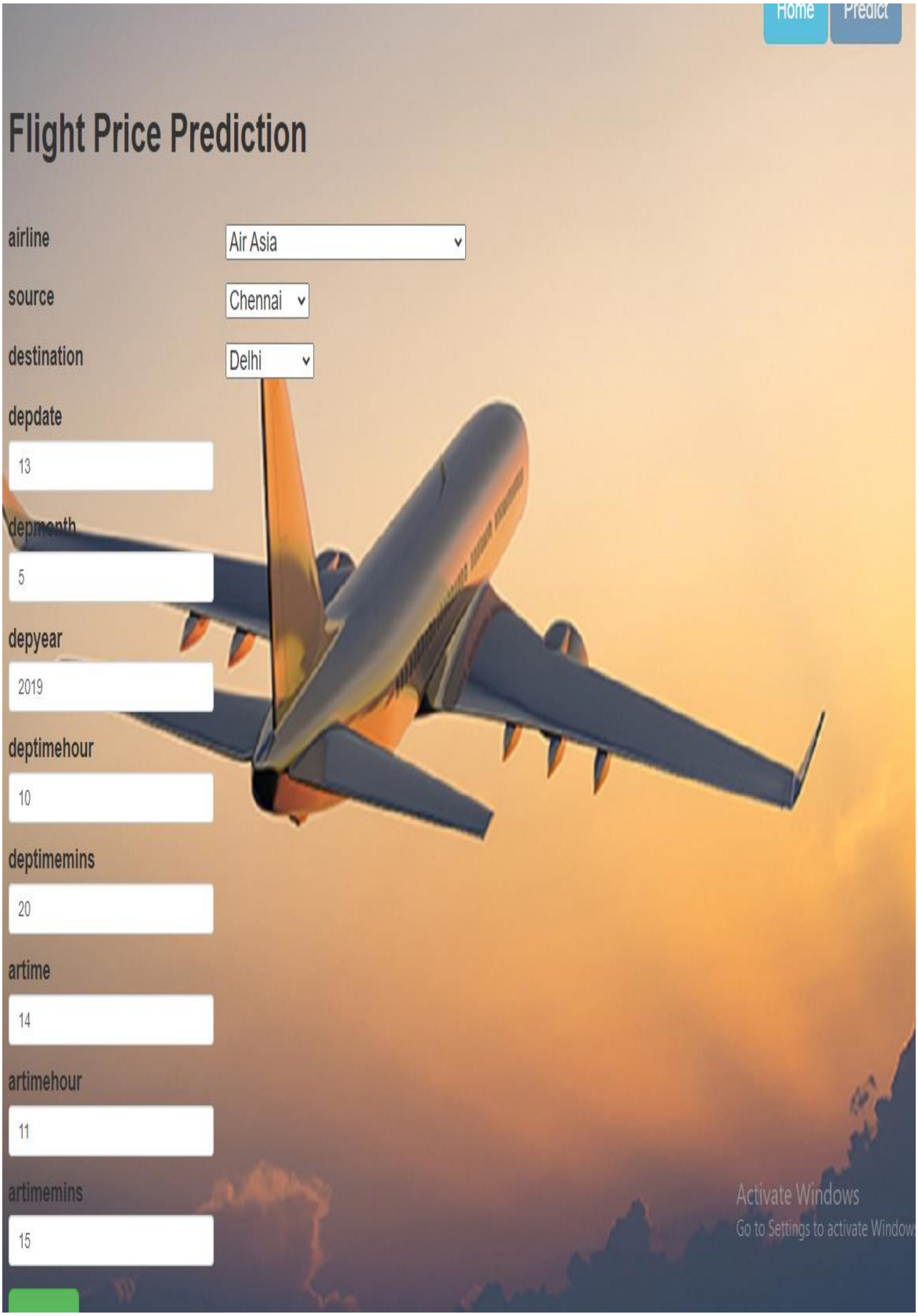
```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

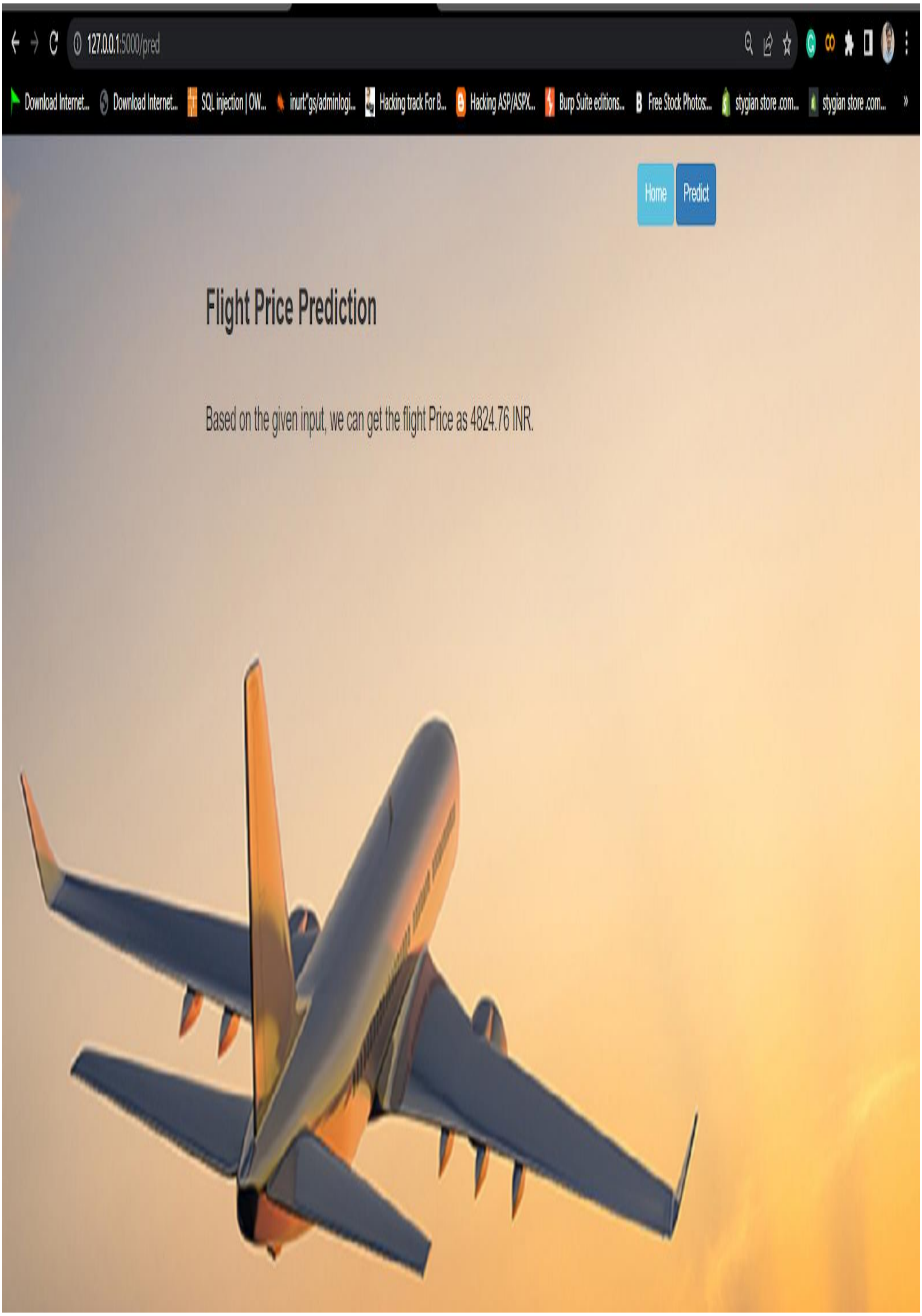
SOURCE CODE


Home
Predict
Flight Price Prediction
The objective of this article is to predict flight prices given the various parameters. This will be a regression problem since the target or dependent variable is the price (continuous numeric value).Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.


Home
Predict
Flight Price Prediction
airline
Air Asia
source
Bangalore
destination
Bangalore
depdate
Date
depmonth
Month
depyear
Year
deptimehour
Dep_Time_Hour
deptimemins
Dep_Time_Mins
artime
Arrival_Date
artimehour


source
Bangalore
destination
Bangalore
depdate
Date
depmonth
Month
depyear
Year
deptimehour
Dep_Time_Hour
deptimemins
Dep_Time_Mins
artime
Arrival_Date
artimehour
Arrival_Time_Hour
artimemins
Arrival_Time_Mins
Submit


Home
Predict
Flight Price Prediction
airline
Air Asia
source
Chennai
destination
Delhi
depdate
13
depmonth
5
depyear
2019
deptimehour
10
deptimemins
20
artime
14
artimehour
11
artimemins
15
Activate Windows
Go to Settings to activate Window


127.0.0.1:5000/pred
Download Internet...
Download Internet...
SQL injection | OW...
inurl:"gs/adminlogi...
Hacking track For B...
Hacking ASP/ASPX...
Burp Suite editions...
Free Stock Photos:...
stygian store .com...
stygian store .com...
Home
Predict
Flight Price Prediction
Based on the given input, we can get the flight Price as 4824.76 INR.

CONCLUSION

Machine Learning algorithms are applied on the dataset to predict the dynamic fare of flights. This gives the predicted values of flight fare to get a flight ticket at minimum cost. The values of R-squared obtained from the algorithm give the accuracy of the model. In the future, if more data could be accessed such as the current availability of seats, the predicted results will be more accurate. Finally, we conclude that this methodology is not preferred for performing this project. We can add more methods, more data for more accurate results

The data is collected from kaggle website and done feature engineering techniques, used random forest algorithm to predict the price of a flight ticket. The accuracy obtained is 0.81 which is good. The more feature engineering techniques can be used and data of more than 1 year can be used to improve the accuracy.