**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.

1. **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.

1. **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',

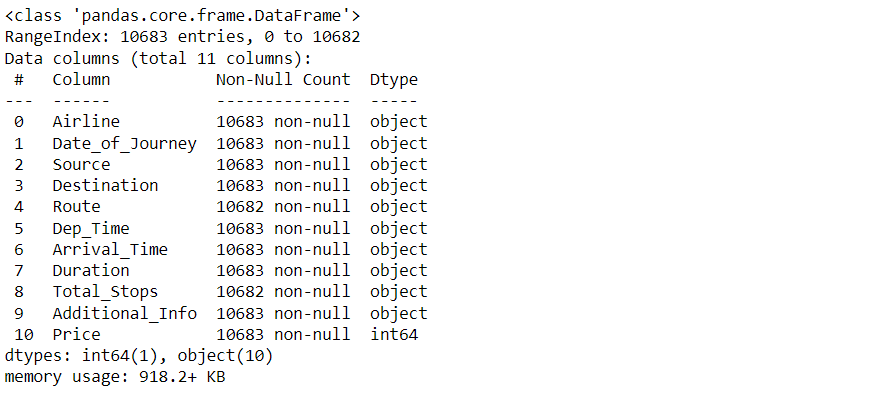
'Additional\_Info', 'Price'],

dtype='object')

**Here we can get more information about our dataset**

train\_df.info()

**Output:**



**To know more about the dataset**

train\_df.describe()

**Output:**



rain\_df.isnull().head()

**Output:**



**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

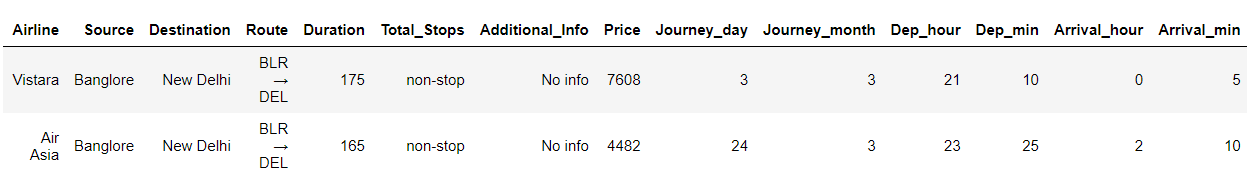
**Dropping NAN values**

train\_df.dropna(inplace = True)

**Duplicate values**

train\_df[train\_df.duplicated()].head()

**Output:**

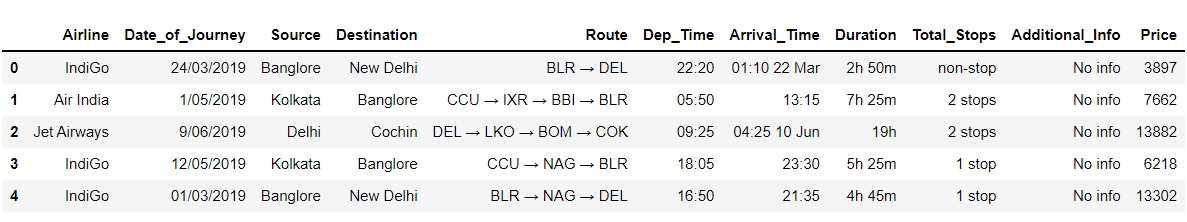


**Here we will be removing those repeated values from the dataset and keeping the in-place attribute to be true so that there will be no changes.**

train\_df.drop\_duplicates(keep='first',inplace=True)

train\_df.head()

**Output:**



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](https://datahack.analyticsvidhya.com/contest/blogathon-16/).

## 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](https://wonderfulengineering.com/wp-content/uploads/2014/05/airplane-wallpaper-2.jpg" \t "_blank)

## 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',

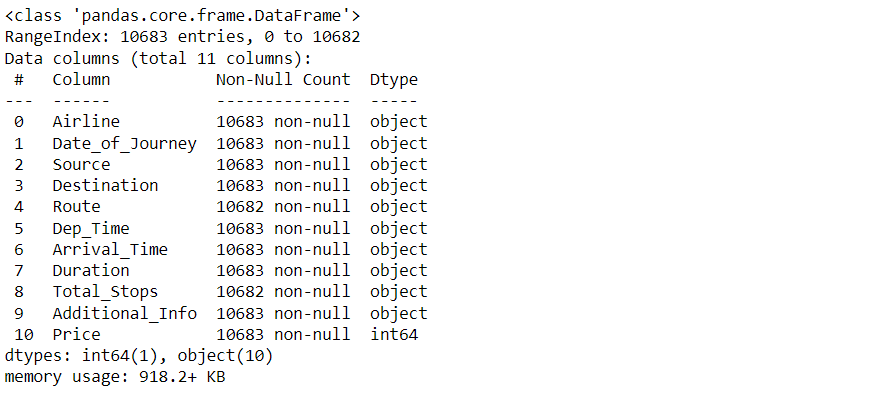
'Additional\_Info', 'Price'],

dtype='object')

**Here we can get more information about our dataset**

train\_df.info()

**Output:**



**To know more about the dataset**

train\_df.describe()

**Output:**



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

train\_df.isnull().head()

**Output:**



**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

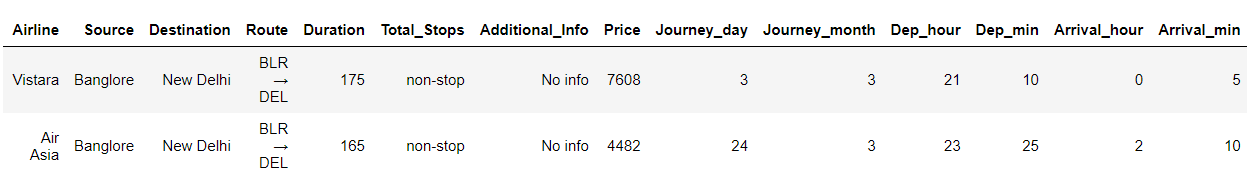
**Dropping NAN values**

train\_df.dropna(inplace = True)

**Duplicate values**

train\_df[train\_df.duplicated()].head()

**Output:**

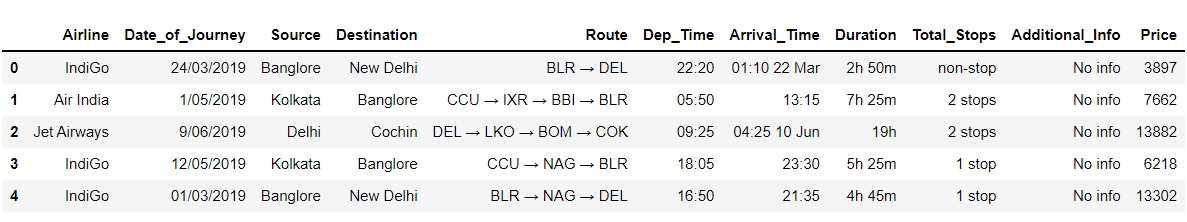


**Here we will be removing those repeated values from the dataset and keeping the in-place attribute to be true so that there will be no changes.**

train\_df.drop\_duplicates(keep='first',inplace=True)

train\_df.head()

**Output:**



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**

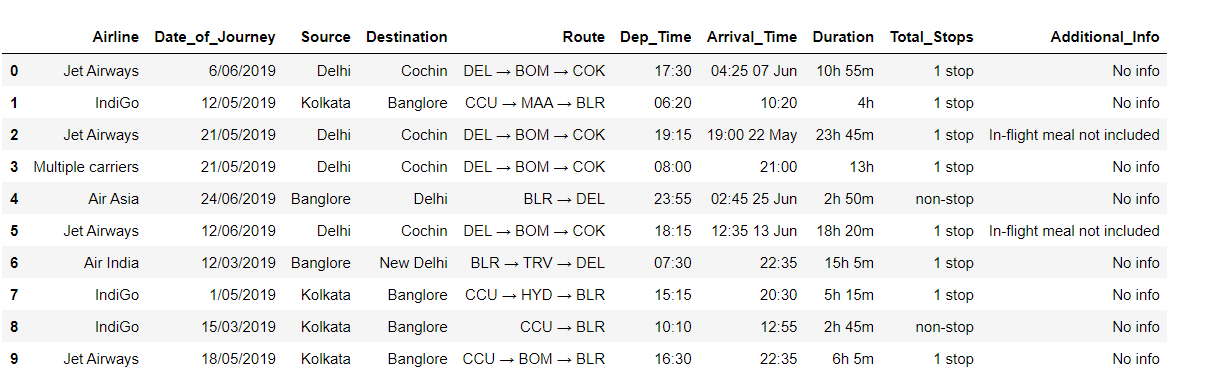
train\_df["Route"].unique()

**Now let’s look at our testing dataset**

test\_df = pd.read\_excel("Test\_set.xlsx")

test\_df.head(10)

**Output:**



**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',

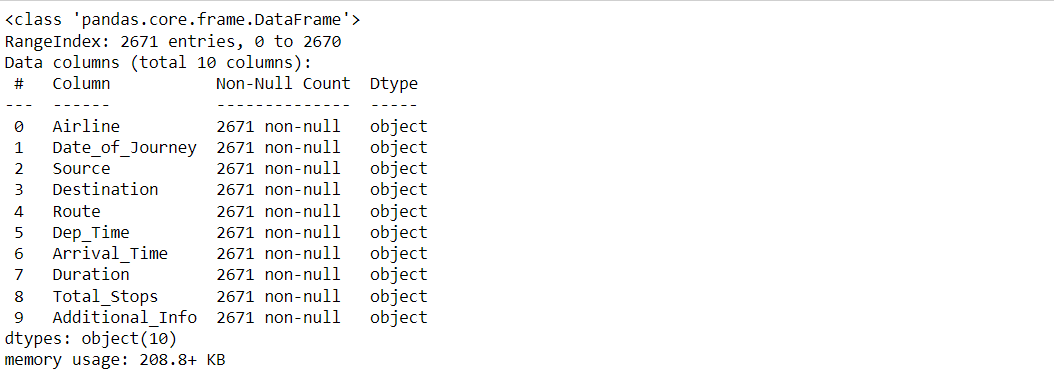
'Additional\_Info'],

dtype='object')

**Information about the dataset**

test\_df.info()

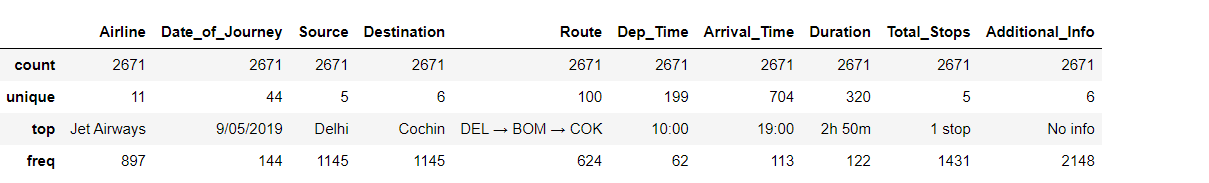
**Output:**



**To know more about the testing dataset**

test\_df.describe()

**Output:**



**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

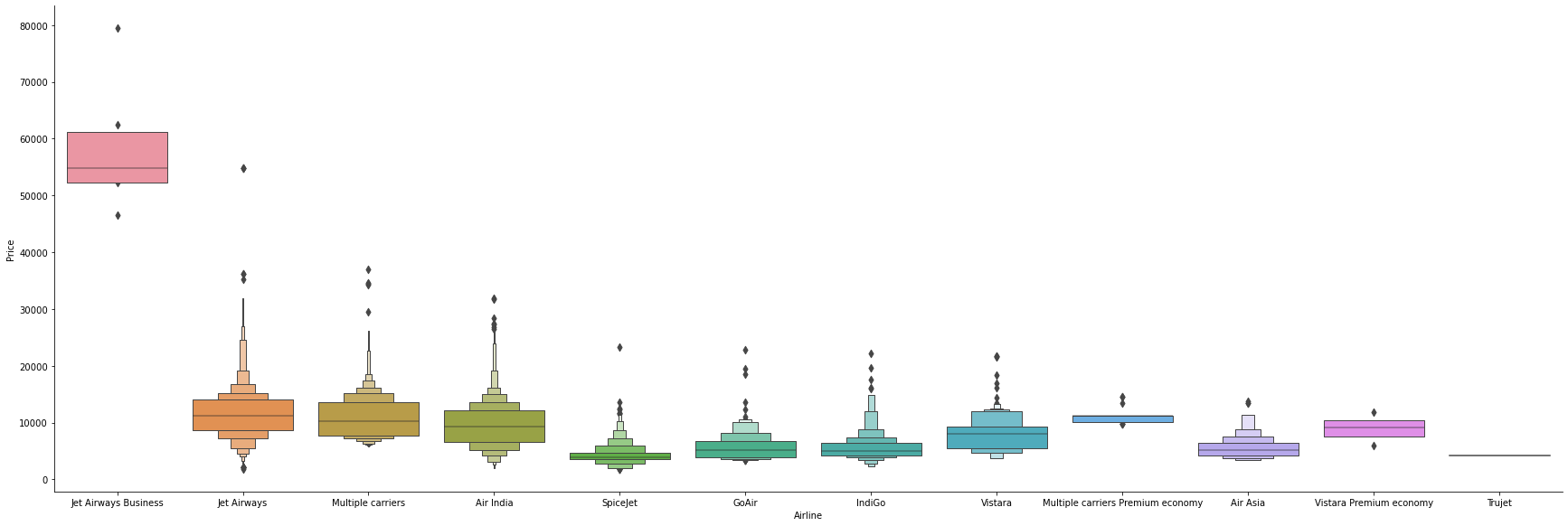
## Data Visualization

**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:**



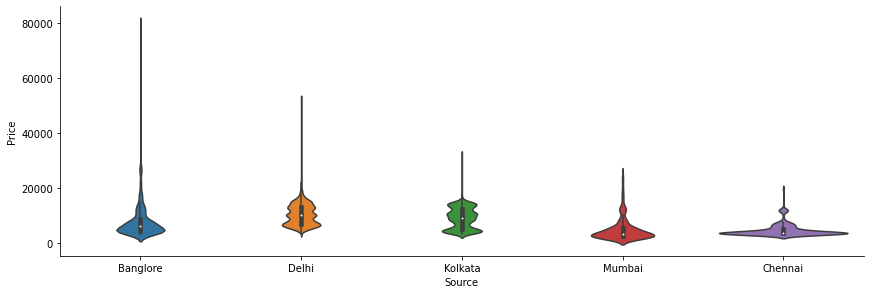
**Inference:** Here with the help of the cat plot we are trying to plot the boxplot between the price of the flight and airline and we can conclude that **Jet Airways has the most outliers in terms of price**.

**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:**

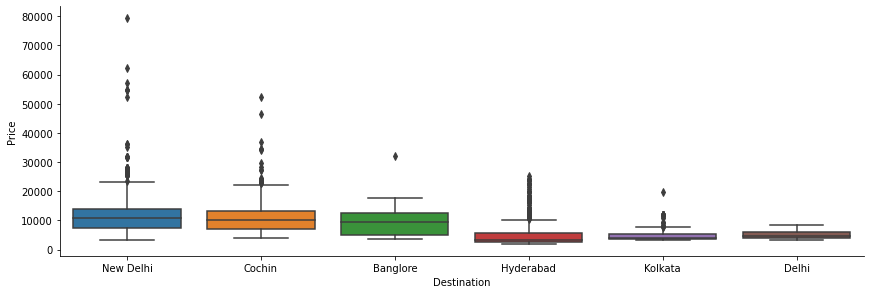


**Plotting Box plot for Price vs Destination**

sns.catplot(y = "Price", x = "Destination", data = train\_df.sort\_values("Price", ascending = False), kind="box", height = 4, aspect = 3)

plt.show()

**Output:**

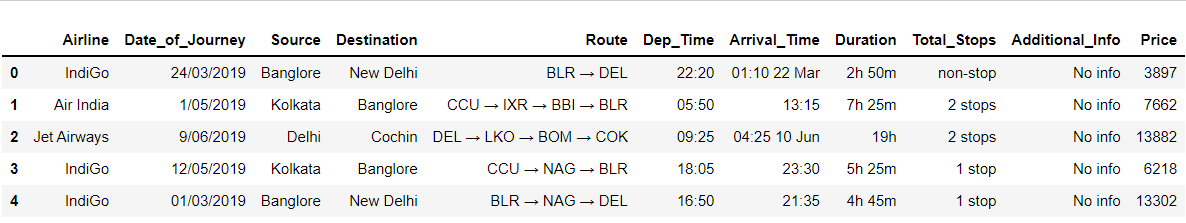


### Feature Engineering

**Let’s see our processed data first**

train\_df.head()

**Output:**



**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

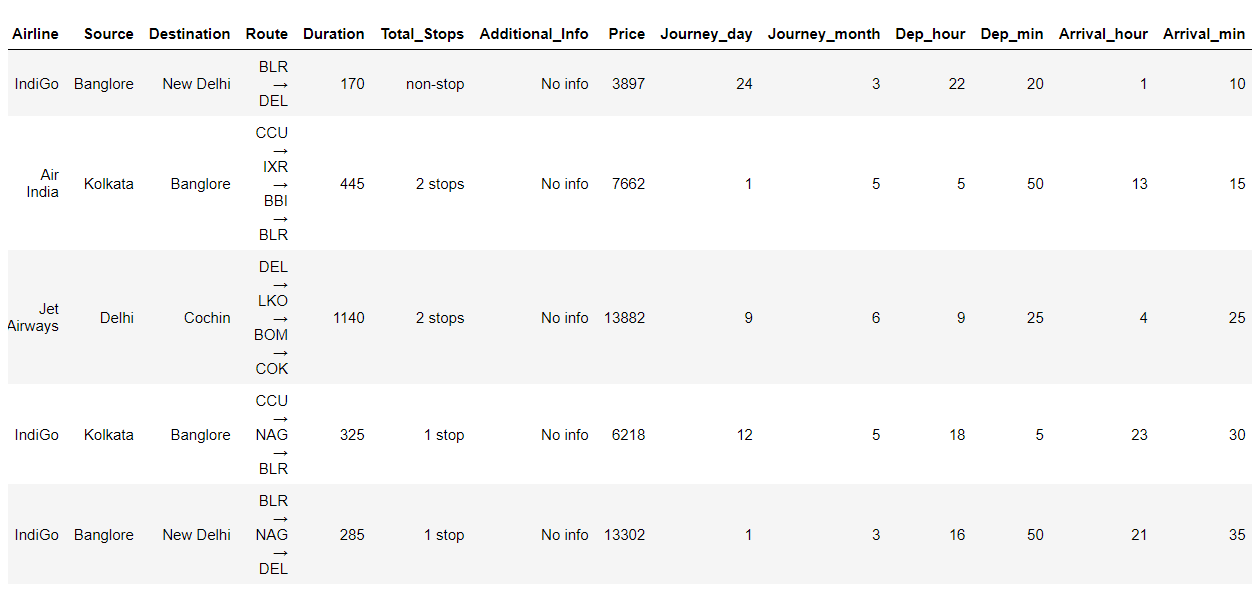
train\_df.drop(["Dep\_Time"], axis = 1, inplace = True)

**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:**



**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)

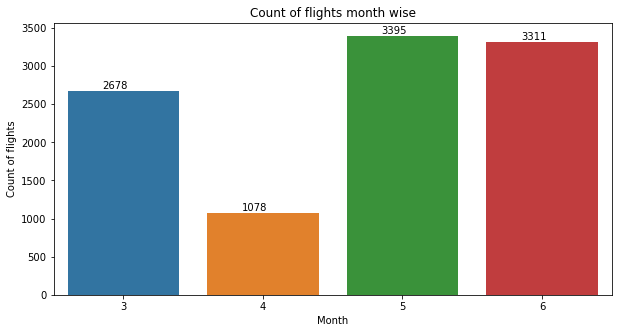
plt.xlabel('Month')

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')

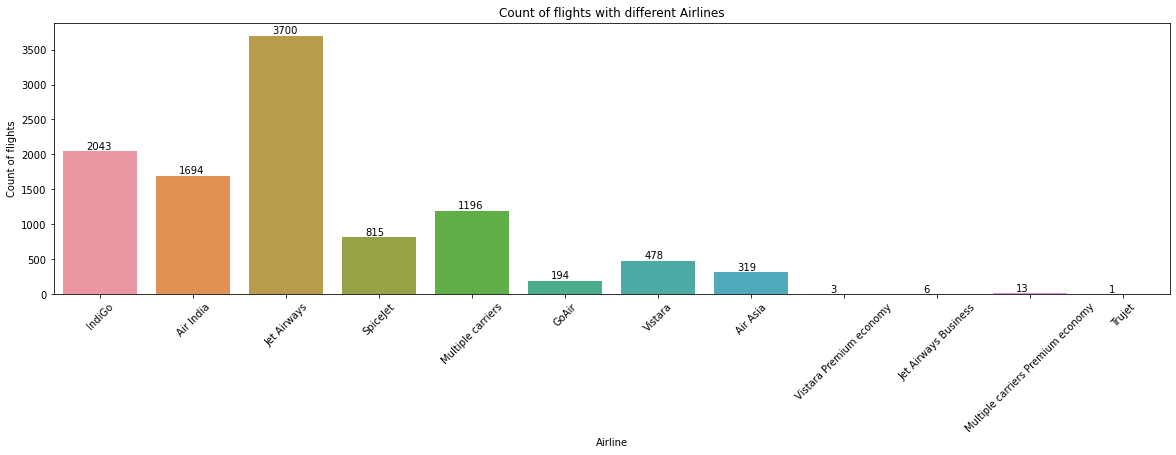
plt.ylabel('Count of flights')

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'])

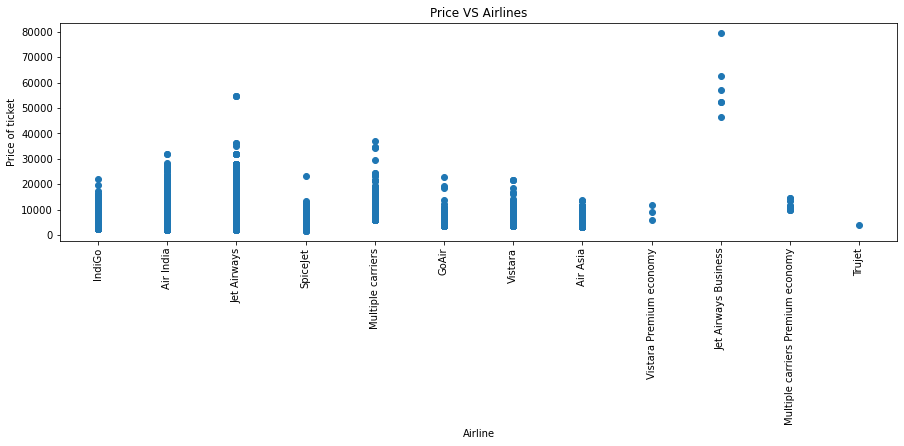
plt.xticks

plt.xlabel('Airline')

plt.ylabel('Price of ticket')

plt.xticks(rotation = 90)

**Output:**



## Correlation between all Features

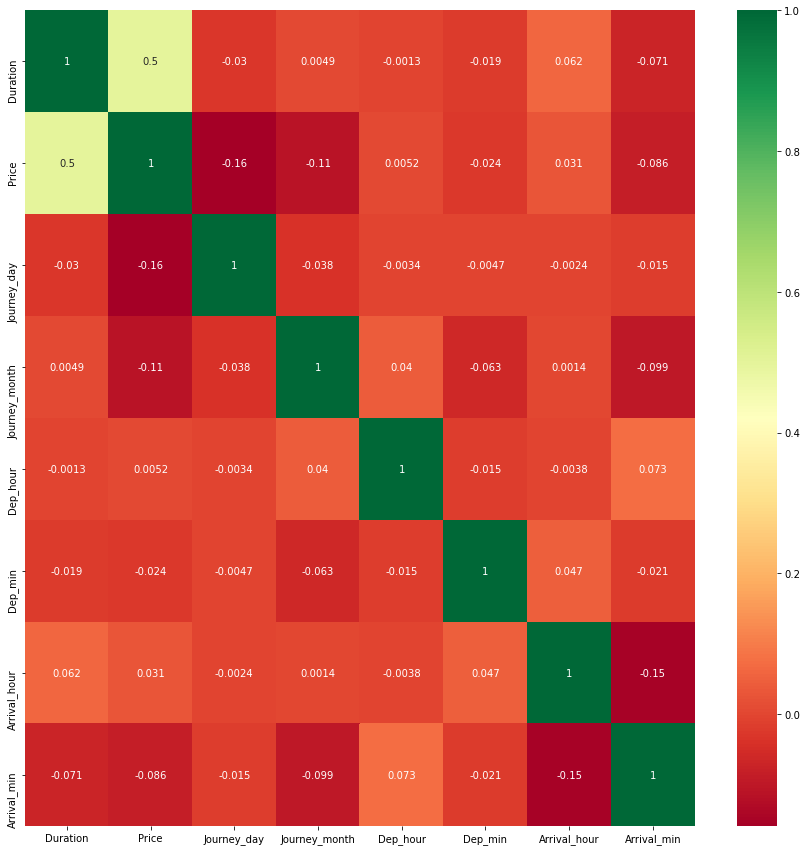
**Plotting Correlation**

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'])

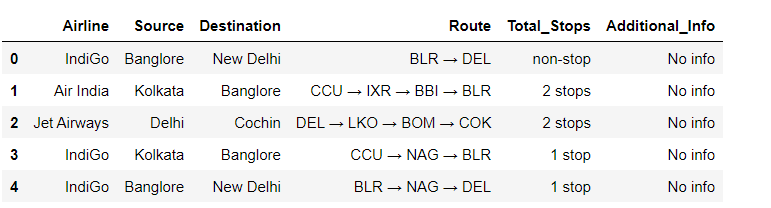
train\_numerical\_data = data.select\_dtypes(include=['int64', 'float','int32'])

test\_categorical\_data = test\_df.select\_dtypes(exclude=['int64', 'float','int32','int32'])

test\_numerical\_data = test\_df.select\_dtypes(include=['int64', 'float','int32'])

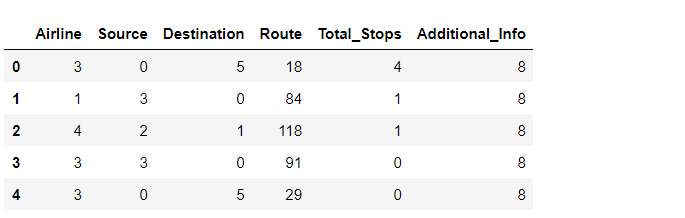
train\_categorical\_data.head()

**Output:**



**Label Encode and Hot Encode for Categorical Columns**

**Output:**



**Concatenating both Categorical Data and Numerical Data**

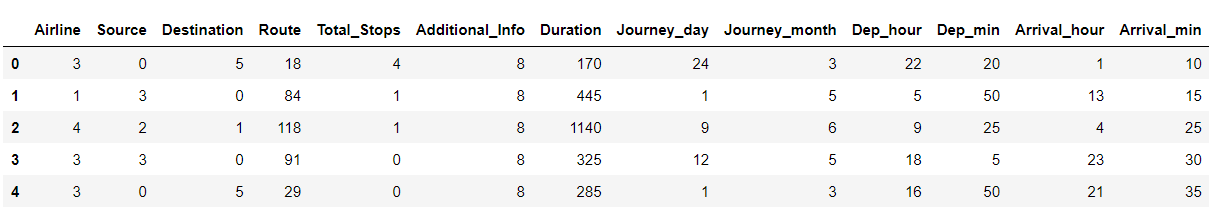
X = pd.concat([train\_categorical\_data, train\_numerical\_data], axis=1)

y = train\_df['Price']

test\_set = pd.concat([test\_categorical\_data, test\_numerical\_data], axis=1)

X.head()

**Output:**



y.head()

**Output:**

0 3897

1 7662

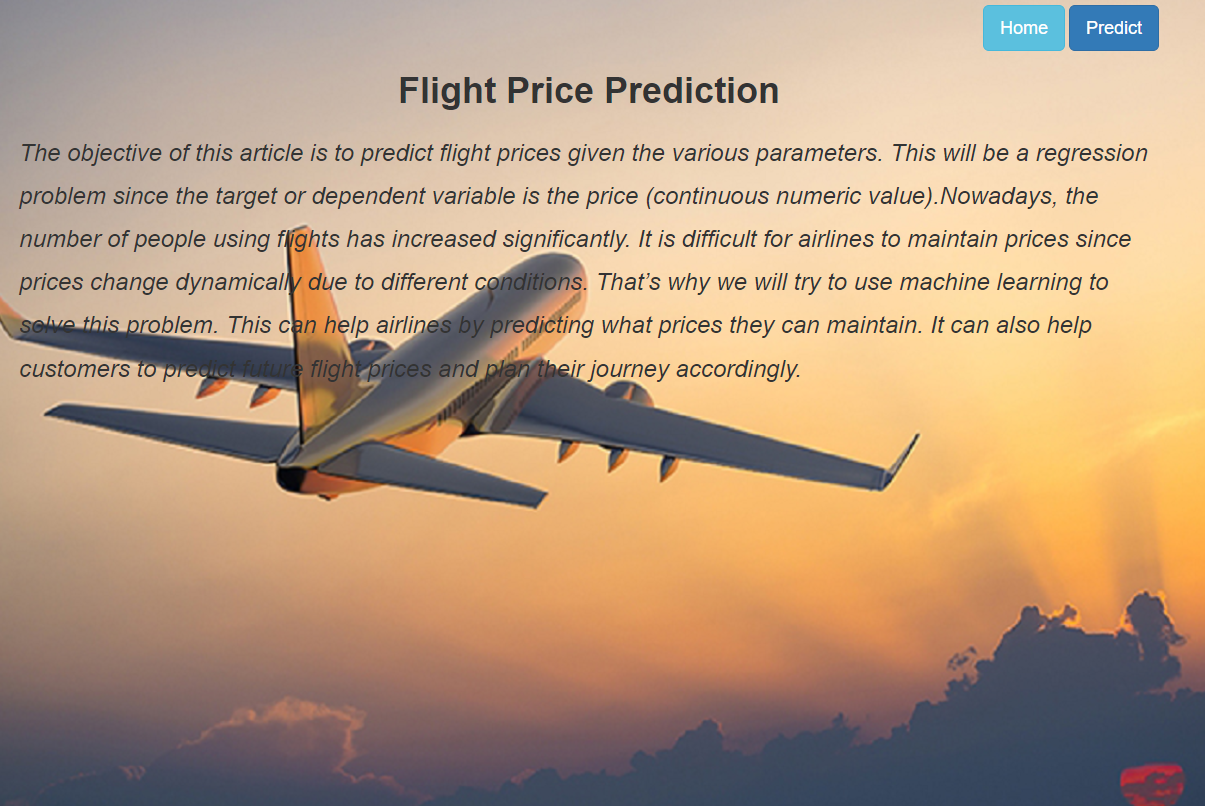
2 13882

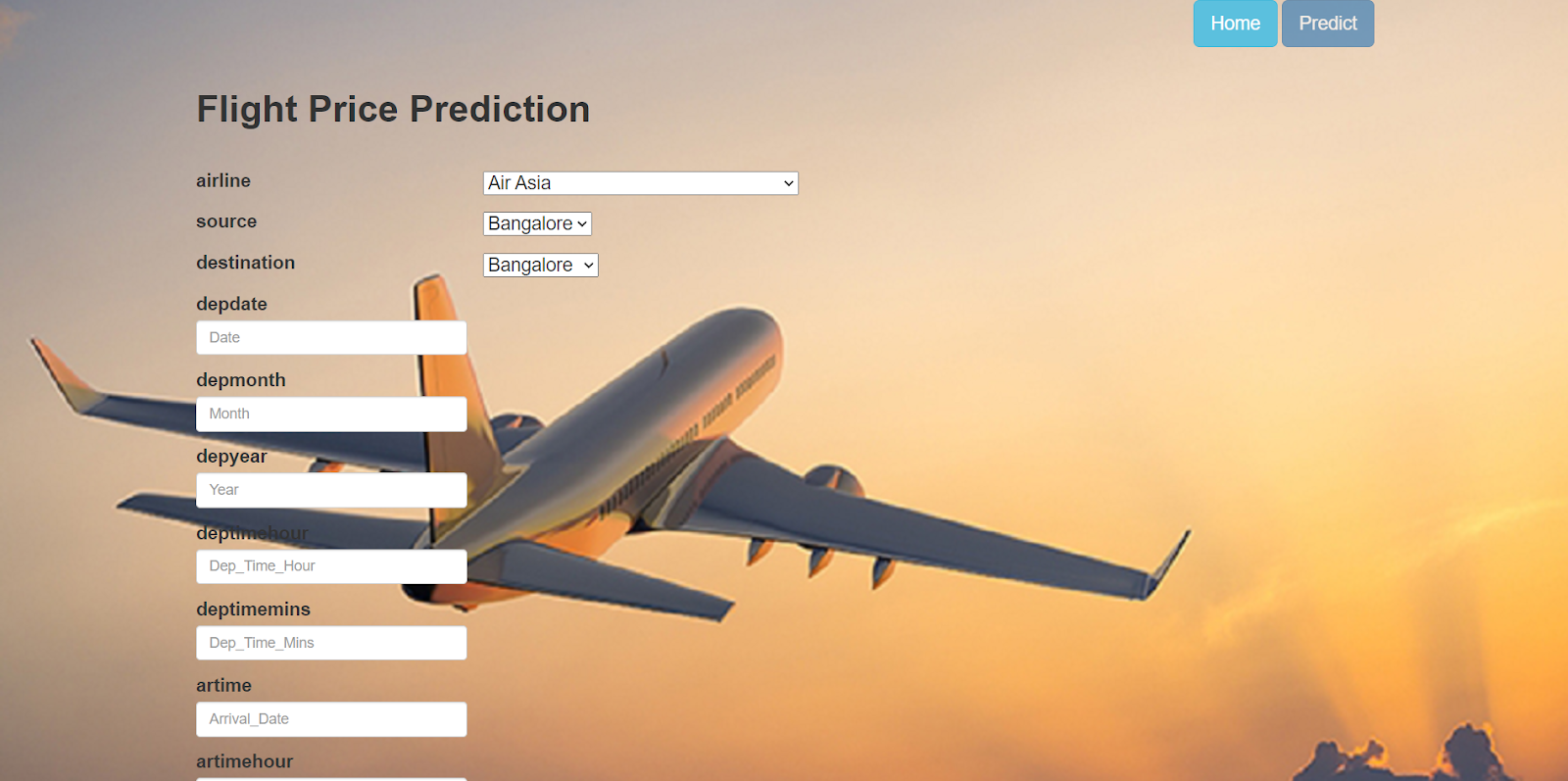
3 6218

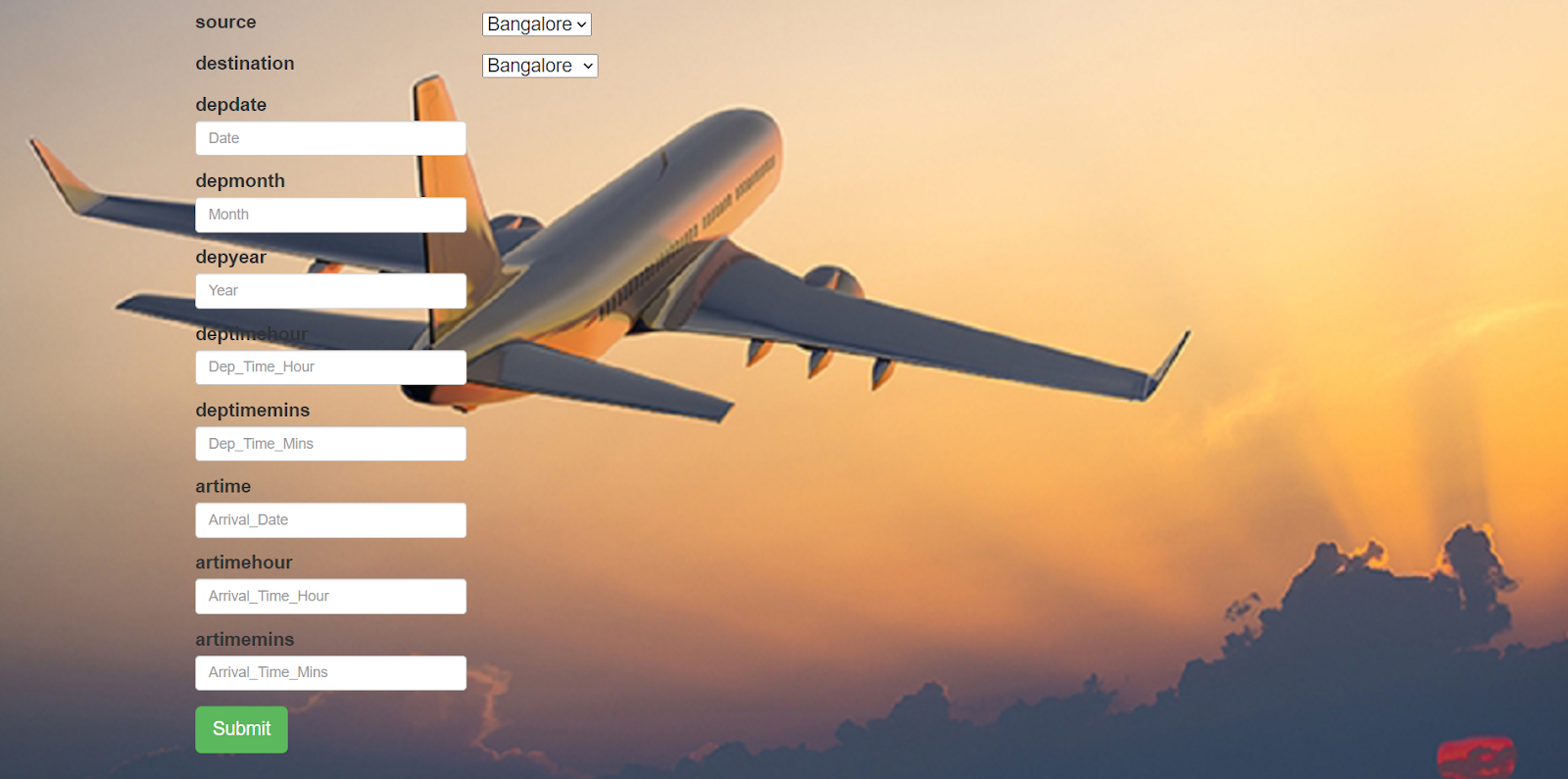
4 13302

Name: Price, dtype: int64

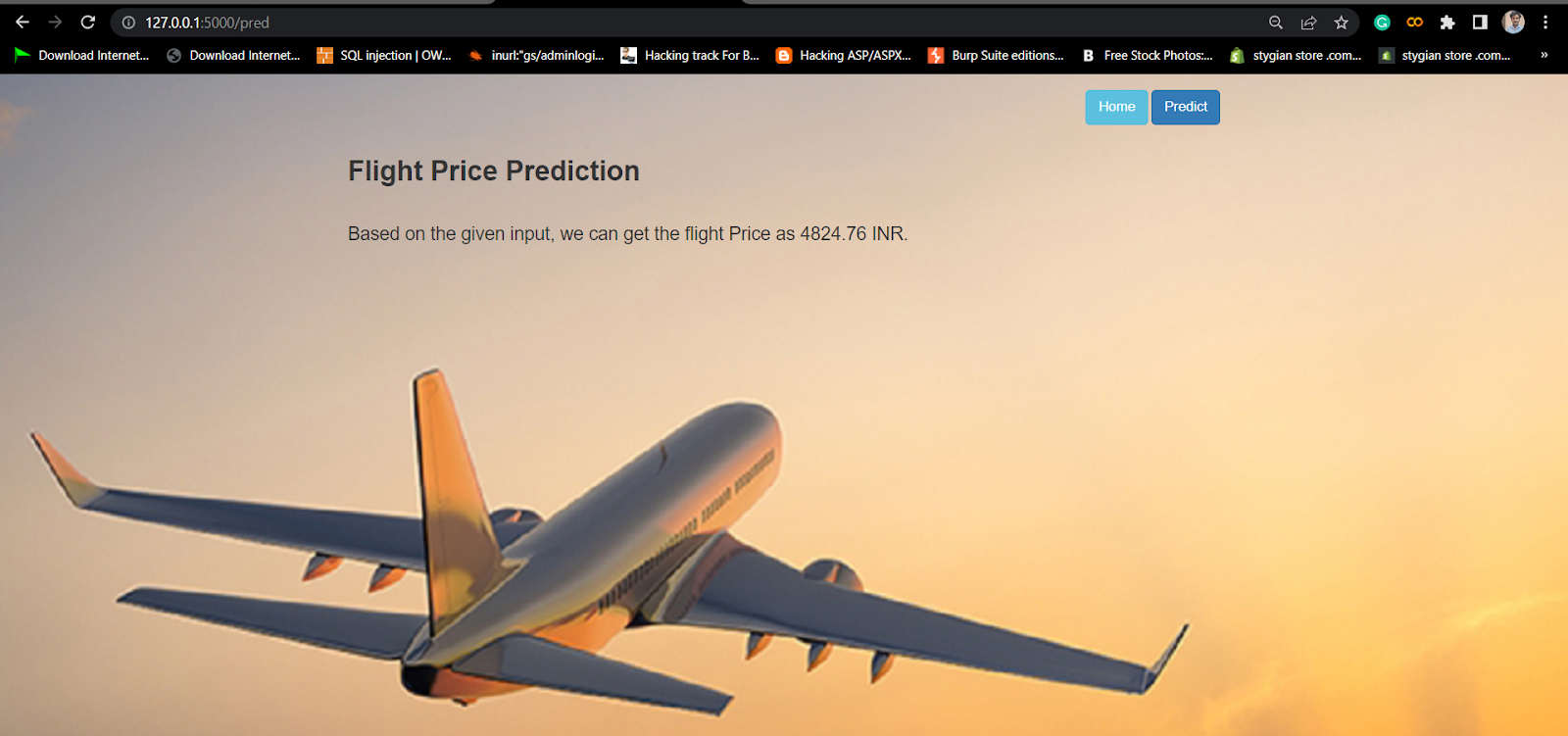
**SOURCE CODE**











**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.