Flight Price Prediction

Submitted by:

Deepak Kumar

ACKNOWLEDGMENT

In successfully completing this project, I would like to thank all those who are related to this project.

I have worked on Customer Retention analysing key factors which are important aspects that affects customer shopping decisions. I have found some insights which helps e-retailers customer activation and retention.

Primarily, I would thank God for being able to complete this project with success. Then I will thank FlipRobo Technologies for providing me this opportunity, my SME Khushboo Garg, under whose guidance I learned about this project. The suggestions and directions have helped in the completion of this project.

Finally, I would like to thank my parents and friends who have helped me directly or indirectly throughout my journey.

INTRODUCTION

- ## Business Problem Framing

Nowadays, the number of people using flights has increased significantly. Anyone who has booked a flight ticket knows how unexpectedly the prices vary. Flight price prediction is a challenging task since the factors involved in pricing dynamically change over time and make the price fluctuate.
The cheapest available ticket on a given flight gets more and less expensive over time.

This usually happens as an attempt to maximize revenue based on -
1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it (raising prices on a flight which is filling up to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

- ## Conceptual Background of the Domain Problem

The price of an airline ticket is affected by several factors, such as flight distance, purchasing time, fuel price, etc. Each carrier has its own proprietary rules and algorithms to set the price accordingly.

Flight ticket data is not well organized and ready for direct analysis, collecting and processing those data always requires a great deal of effort. Recent advance in Artificial Intelligence (AI) and Machine Learning (ML) makes it possible to infer such rules and model the price variation. It can also help customers to predict future flight prices and plan their journey accordingly.

- ## Review of Literature

The flight ticket buying system is to purchase a ticket many days prior to flight take-off to stay away from the effect of the most extreme charge. Mostly, aviation routes don’t agree this procedure. Plane organizations may diminish the cost at the time, they need to build the market and at the time when the tickets are less accessible.

They may maximize the costs. So, the cost may rely upon different factors. To foresee the costs this, venture uses AI to exhibit the ways of flight tickets after some time. All organizations have the privilege and opportunity to change its ticket costs at any time. Explorer can set aside cash by booking a ticket at the least costs. People who had travelled by flight frequently are aware of price fluctuations.

The airlines use complex policies of Revenue Management for execution of distinctive evaluating systems. The evaluating system as a result changes the charge depending on time, season, and festive days to change the header or footer on successive pages. The aim of the airways is to earn profit whereas the customer searches for the minimum rate. Customers usually try to buy the ticket well in advance of departure date to avoid hike in airfare as date comes closer. But this is not the fact. The customer may wind up by giving more than they ought to for the same seat.

- ## Motivation for the Problem Undertaken

It is hard for the client to buy an air ticket at the most reduced cost. For this, few procedures are explored to determine time and date to grab air tickets with minimum fare rate. Many of these systems are utilizing the modern computerized system known as Machine Learning.

Someone who purchase flight tickets frequently would be able to predict the right time to procure a ticket to obtain the best deal. Many airlines change ticket prices for their revenue management. The airline may increase the prices when the demand is to be expected to increase the capacity.

To estimate the minimum airfare, data for a specific air route has been collected including the features like departure time, arrival time and airways over a specific period. Features are extracted from the collected data to apply Machine Learning (ML) models. We use the machine learning regression methods to predict the prices at the given time.

Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

  In this project we have used different inbuilt python methods to check the statistics of the data.

  To understand the different datatypes of the attributes I have used ‘dtype’ method, to check if there are any null values present in the dataset, I have used ‘isnull().sum()’ method.{It is also provided in dataset there are no null values}.
  As there were less number of numerical attributes earlier, I have not used ‘describe()’ method, But the describe() method returns description of the data in the DataFrame.

  If the DataFrame contains numerical data, the description contains this information for each column: count - The number of not-empty values. mean - The average (mean) value. std - The standard deviation. min-The minimum value. max- The maximum value of attribute.

  The info() method prints information about the DataFrame. The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-null values). Note: the info() method actually prints the info

- ## Data Sources and their formats

  The accumulation of information is the most significant part of this venture. The different wellsprings of the information on various sites are utilized to prepare the models. Sites provide data about the numerous courses, times, flights, and charge.

  I have used Selenium tool to gather the data from different websites. I have used ‘yatra.com’ and scrape the various details like name of airlines, arrival time, departure time, duration, Source location, Destination location, Date of departure, Number of stops, and collected the data in form of csv format. All the attributes collected are in categorical format which need to be treated except the date of departure.

In [1]:
```
import pandas as pd
import selenium
from selenium import webdriver
import time
from time import sleep

import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
Airline_name = []
Date = []
arrival_time=[]
dep_time = []
source = []
destination=[]
duration = []
total_stops = []
price=[]
```

```
#Loading web driver for June 9 Mumbai to Delhi
driver = webdriver.Chrome('chromedriver.exe')
driver.get('https://flight.yatra.com/air-search-ui/dom2/trigger?type=O&viewName=normal&flexi=0&noOfSegments=1&origin=BOM&originCo
driver.maximize_window()
```

```
last_height = driver.execute_script("return document.documentElement.scrollHeight")
```

```
while True:
    # Scroll down 'till "next load".
    driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")

    # Wait to load everything thus far.
    time.sleep(2)

    # Calculate new scroll height and compare with last scroll height.
    new_height = driver.execute_script("return document.documentElement.scrollHeight")
    if new_height == last_height:
        break
    last_height = new_height

    # One last scroll just in case.
driver.execute_script("window.scrollTo(0, document.documentElement.scrollHeight);")
sleep(2)
```

```
#airline tags
try:
    #tag = driver.find_elements_by_xpath('//div/div[1]/div[1]/div/div[2]/span')
    tag = driver.find_elements_by_xpath('//span[@class="i-b text ellipsis"]')
    for i in tag:
        Airline_name.append(i.text)

    l1 = len(Airline_name)
    length=l1
except:
    Airline_name.append('Nan')

#Airline_name
#len(Airline_name)


#departure time
try:
    tag = driver.find_elements_by_xpath('//div[@class="i-b pr"]')
    for i in tag:
        dep_time.append(i.text)
except:
    dep_time.append('Nan')


#arrival time
try:
    tag = driver.find_elements_by_xpath('//p[@class="bold fs-15 mb-2 pr time"]')
    for i in tag:
        arrival_time.append(i.text)
except:
    arrival_time.append('Nan')


#source
try:
    tag = driver.find_elements_by_xpath('//div[@class="i-b col-4 no-wrap text-right dtime col-3"]/p')
    for i in tag:
        source.append(i.text)
except:
    source.append('Nan')

#destination
try:
    tag = driver.find_elements_by_xpath('//div[@class="i-b pdd-0 text-left atime col-5"]//p[2]')
    for i in tag:
        destination.append(i.text)
except:
    destination.append('Nan')

#total stops
```

```
#total stops

try:
    tag = driver.find_elements_by_xpath('//div[@class=" font-lightgrey fs-10 tipsy i-b fs-10"]/span')
    for i in tag:
        total_stops.append(i.text)
except:
    total_stops.append('Nan')

#Price
try:
    tag = driver.find_elements_by_xpath('//div/div[1]/div[4]/div/div[1]/div/label/div/div[2]')
    for i in tag:
        price.append(i.text)

except:
    price.append('Nan')


#duration
try:
    tag = driver.find_elements_by_xpath('//p[@class="fs-12 bold du mb-2"]')
    for i in tag:
        duration.append(i.text)

except:
    duration.append('Nan')
```

```
len(Airline_name),len(arrival_time),len(dep_time),len(source),len(destination),len(duration),len(total_stops),len(price)
```

```
(177, 177, 177, 177, 177, 177, 177, 177)
```

```
#9 June
df_Mum_Del = pd.DataFrame({'Airline Name':Airline_name, 'Departure Date':'9 June 2022','Arrival Time':arrival_time,'Departure Tim
df_Mum_Del
```

	Airline Name	Departure Date	Arrival Time	Departure Time	Source	Destination	Duration	Total Stops	Price
0	Air India	9 June 2022	19:10	15:25	Mumbai	New Delhi	3h 45m	1 Stop	7,308
1	Go First	9 June 2022	20:05	18:00	Mumbai	New Delhi	2h 05m	Non Stop	8,042
2	Go First	9 June 2022	02:20	00:10	Mumbai	New Delhi	2h 10m	Non Stop	8,042
3	Go First	9 June 2022	11:35	09:25	Mumbai	New Delhi	2h 10m	Non Stop	8,042
4	Go First	9 June 2022	13:10	11:00	Mumbai	New Delhi	2h 10m	Non Stop	8,042

2485	Air India	10 June 2022	13:25\n+ 1 day	09:15	Patna	Lucknow	28h 10m	3 Stop(s)	17,963
2486	Air India	10 June 2022	13:20\n+ 1 day	19:10	Patna	Lucknow	18h 10m	2 Stop(s)	22,842
2487	Air India	10 June 2022	13:20\n+ 1 day	19:10	Patna	Lucknow	18h 10m	2 Stop(s)	22,842
2488	Air India	10 June 2022	13:20\n+ 1 day	16:00	Patna	Lucknow	21h 20m	2 Stop(s)	23,577
2489	Air India	10 June 2022	13:20\n+ 1 day	16:00	Patna	Lucknow	21h 20m	2 Stop(s)	23,577

2490 rows × 9 columns

```
df_flightData.to_csv('flight_datas.csv')
```

There are 2490 rows and 9 columns which I have scraped from this website. All the attributes are categorical except departure date, departure month.

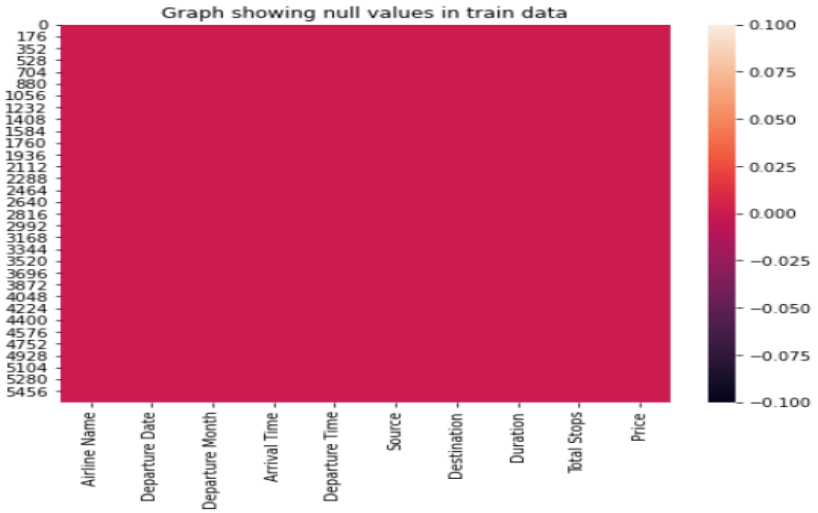
```
1 #checking the names of all the attributes in the data sets
2
3 print(' Column names:\n',df.columns)
```

```
Column names:
Index(['Airline Name', 'Departure Date', 'Departure Month', 'Arrival Time',
       'Departure Time', 'Source', 'Destination', 'Duration', 'Total Stops',
       'Price'],
      dtype='object')
```

```
#Let us check the datatypes inwhich all the attributes are available for both test and train sets

df.dtypes
```

```
Airline Name       object
Departure Date      int64
Departure Month     int64
Arrival Time       object
Departure Time     object
Source             object
Destination        object
Duration           object
Total Stops        object
Price              object
dtype: object
```

```
#visualizing the null values after the removal for both train and test data

plt.figure(figsize=(8,6),facecolor='white')
sns.heatmap(df.isnull())
plt.title('Graph showing null values in train data')
plt.show()
```



- **Data Pre-processing Done**

  All the gathered information required a great deal of work, so after the accumulation of information, it should have been perfect and be ready as indicated by the model prerequisites.

  Different statistical methods and logics in python clean and set up the information. For instance, the price was character type, not a number. Additional features are created to get more accurate results.

Changing the datatype of Price from categorical to Numerical

```
for i in range(len(df)):
    df['Price'][i]=df['Price'][i].replace(',','')

df['Price'] = df['Price'].astype(int)
```

As Vistara, Business, Go Air, Go First, Vistara Premium Economy, Multiple carriers' Premium economy has very less flights we need to take care of this data.
Sources have same locations but mentioned in different ways, we changed the format.
Destination has same locations but mentioned in different ways, we corrected the format.
Total number of stops have the information but mentioned in different ways we have changed the format.

```
#Replacing names of cities mentioned different ways in Source and Destination into same format.
df['Source'].replace({'New Delhi':'Delhi','Banglore':'Bangalore'},inplace=True)

df['Destination'].replace({'New Delhi':'Delhi','Banglore':'Bangalore'},inplace=True)


#converting airlines with less numbers as others
df['Airline Name'].replace(to_replace=['Go First','GoAir','Multiple carriers Premium economy','Jet Airways Business','Trujet','Vi
```

```
#cleaning the 'Total Stops' attribute
df['Total Stops'].replace(to_replace=['1 stop','1 Stop'],value=1,inplace=True)
df['Total Stops'].replace(to_replace=['non-stop','Non Stop'],value=0,inplace=True)
df['Total Stops'].replace(to_replace=['3 stops','3 Stop(s)'],value=3,inplace=True)
df['Total Stops'].replace(to_replace=['2 stops','2 Stop(s)'],value=2,inplace=True)

df['Total Stops'].value_counts()
```

```
1    1652
0     574
2     256
3       8
Name: Total Stops, dtype: int64
```

```
for i in range(len(df['Arrival Time'])):
    df['Arrival Time'][i]=df['Arrival Time'][i].replace('+ 1 day','').split('\n')[0]
    df['Arrival Time'][i]=df['Arrival Time'][i].split(' ')[0]
```

```
#extracting Hour of Arrival and minute of departure from departure attribute
df['Hour_Arr']  = pd.to_datetime(df['Arrival Time']).dt.hour
df['Minute_Arr']  = pd.to_datetime(df['Arrival Time']).dt.minute

#dropping the arrival_time from the dataset permanently
df.drop(columns='Arrival Time', inplace=True)

```

Based on the hours we split the data into early morning flights, morning flights or evening/Night flights.

In [28]:
```
#extracting Hour of departure and minute of departure from departure attribute
df['Hour_Dep']  = pd.to_datetime(df['Departure Time']).dt.hour
df['Minute_Dep']  = pd.to_datetime(df['Departure Time']).dt.minute

#dropping the dep_time from the dataset permanently
df.drop(columns='Departure Time', inplace=True)
```

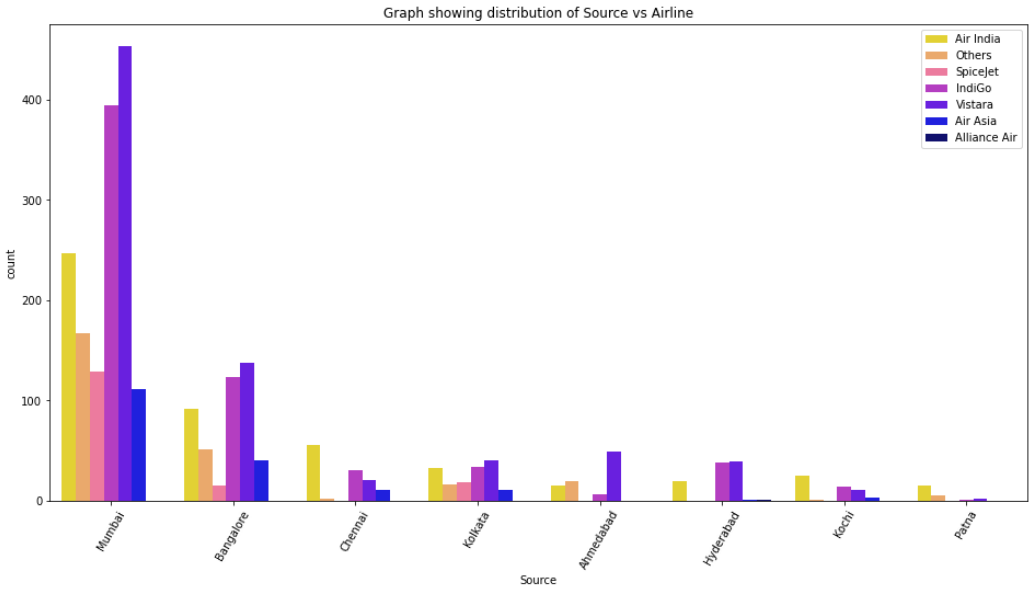
In [29]:
```
for i in range(0,24):
    if i<=6:
        df['Hour_Dep'].replace(i,'Early Morning',inplace=True)
    elif i>6 and i <=12:
        df['Hour_Dep'].replace(i,'Morning',inplace=True)
    elif i>12 and i<=18:
        df['Hour_Dep'].replace(i,'Afternoon',inplace=True)
    elif i>18 and i<24:
        df['Hour_Dep'].replace(i,'Night',inplace=True)
    else:
        pass

df['Hour_Dep'].value_counts()
```

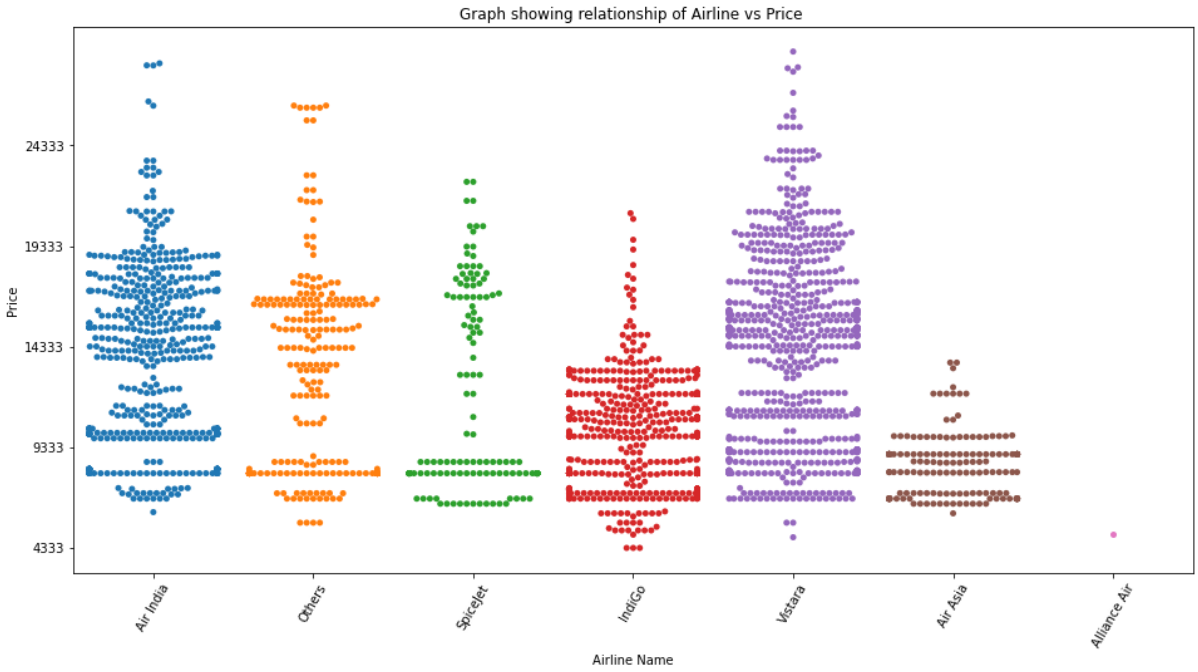
Out[29]:
```
Morning          902
Afternoon        658
Night            514
Early Morning    416
Name: Hour_Dep, dtype: int64
```

- **Data Inputs- Logic- Output Relationships**

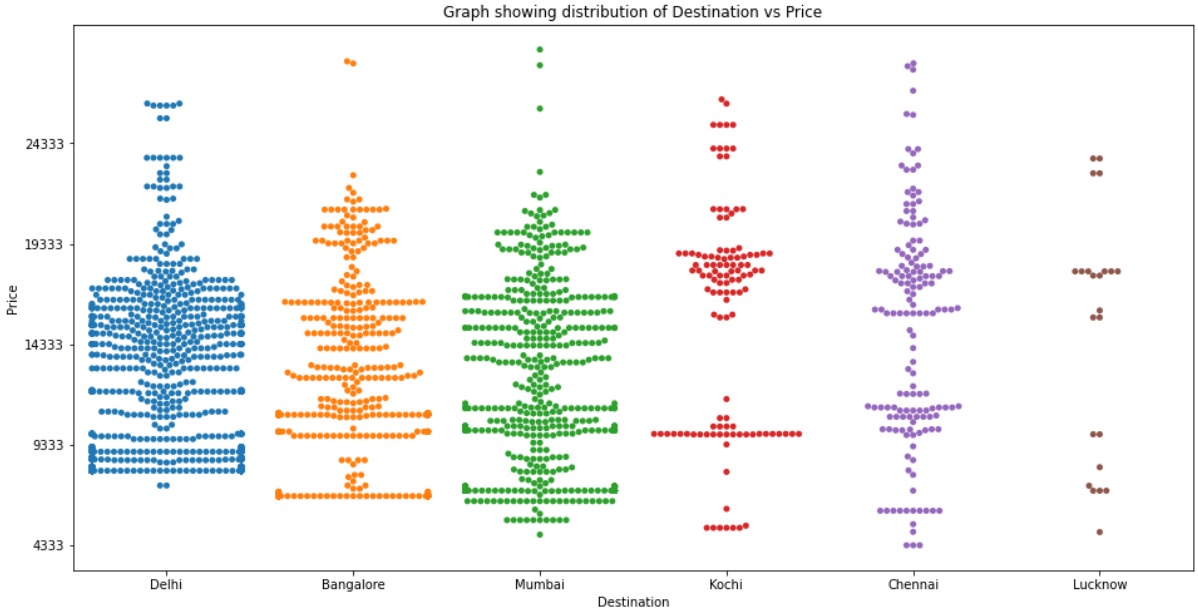
  Indigo flights and Vistara constitute nearly 50% of total flights compared to other flights used.

  The hospitality of the flight, pricing ranges may be considered for this high number of usages.





The number of flights is distributed normally except Weekends or holidays over the period.
Nearly 50 % of total flight travel is observed from New Delhi and Kolkata.
30% of the flights travelling have Cochin as destination.
75% of the flights travelling are not having any stops or have only 1 stop.



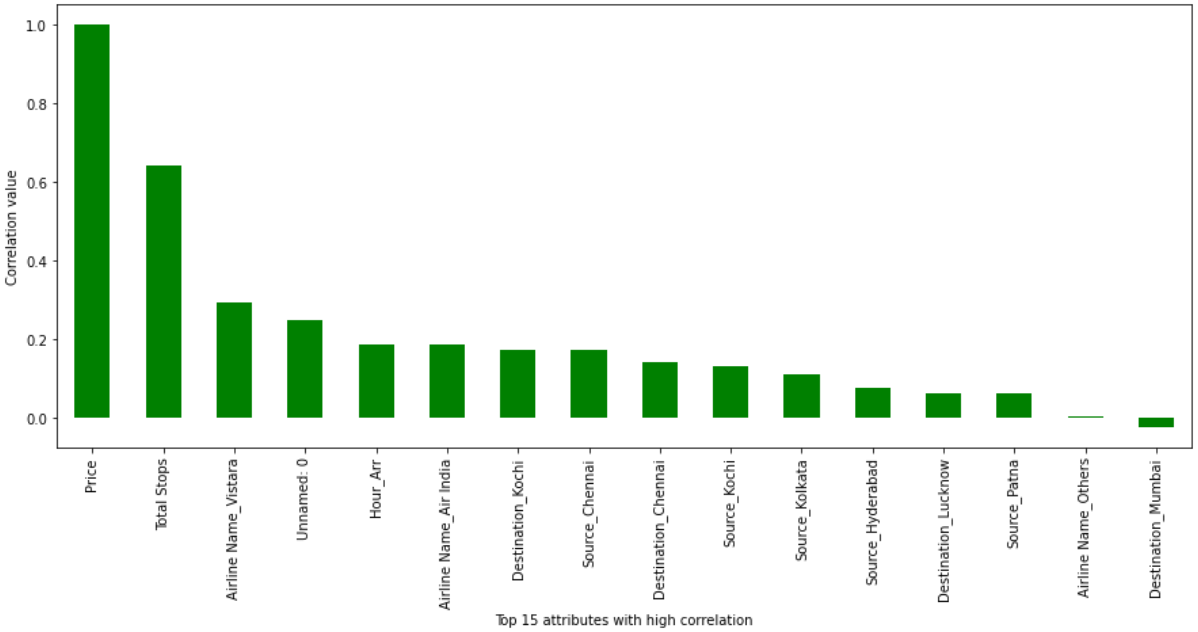
We can see there are peaks formed in all the 4 months of observations marked in different colours. All the spikes seen are due to weekends (Saturday and Sunday). The increase in price of tickets is about 10% of the regular price range.

We can expect the price is gradually higher than regular days.

As per the observations, no matter of date of booking, we can expect pricings are to be higher during the weekends, one-month prior of the expected date of journey can reduce the price of tickets during weekdays.

- **State the set of assumptions (if any) related to the problem under consideration**

The following figure shows the correlation between top 15 attributes and target variable.



1. I assume that dataset gathered from website is complete random sample which is representative of the population.
2. There is minimal or no multicollinearity among the independent variables.

- **Hardware and Software Requirements and Tools Used**

Following are the recommended hardware requirement to build and run machine learning model.

i. 7th generation (Intel Core i7 processor)
ii. 8GB RAM / 16 GB RAM (recommended)

We have used following software and tools for the machine learning model.

ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. The distribution includes data-science packages suitable for Windows, Linux, and macOS.

```
import math
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings('ignore')
```

```
from sklearn.preprocessing import StandardScaler
from scipy.stats import zscore
from statsmodels.stats.outliers_influence import variance_inflation_factor
from sklearn.model_selection import GridSearchCV,train_test_split,cross_val_score

from sklearn.linear_model import LinearRegression, Lasso
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import AdaBoostRegressor,GradientBoostingRegressor,RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from xgboost import XGBRegressor

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
```

The figure shows the important libraries I have imported to execute the project.

I have used built in Data science libraries like pandas, NumPy, Visualization libraries like matplotlib and seaborn. Jupyter Notebook, a shareable notebook that combines live code, visualizations, and text.

Machine learning libraries like scikit-learn for data pre-processing, model selection, model evaluation, SciPy for standardizing& normalizing the data.

Model/s Development and Evaluation

- ## Identification of possible problem-solving approaches (methods)

  The dataset provided has huge volume of the data which did not have any null values, but there were outliers present in the ‘price’ attribute of dataset, unless outlier treatment there is possibility of our machine learning model overfitting the data or increase the variability in the data. Keeping the data loss into concern, Z Score method is implemented to reduce the outliers. The attributes which are having less correlation with target variable have been dropped and removed based on the inference learned from heatmaps and bar plot.

- **Testing of Identified Approaches (Algorithms)**

  To feed the dataset to model, the independent and dependent variables are to be split and the independent attributes are standardized using ‘StandardScaler’ library.

  Now the data obtained is clean and is having least multicollinearity, independent and balanced data. ‘train_test_split’ function in model selection is used for splitting data arrays into two subsets: for training data and for testing data.

```
:  1 #splitting the data
   2 X = df.drop('Price', axis=1)
   3 y = df['Price']
```

Standardization and splitting the dataset into train and test datasets

```
In [48]: #standardizing the data

         scalar = StandardScaler()
         X_scaled = scalar.fit_transform(X)

In [49]: X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,random_state=56, test_size=0.25)

         X_train.shape, X_test.shape, y_train.shape, y_test.shape

Out[49]: ((1854, 21), (619, 21), (1854,), (619,))
```

We train the model using the training set and then apply the model to the test set.

```
1 #Checking the importance of the attributes
2 vif = pd.DataFrame()
3 vif['Features'] = X.columns
4 vif['vif']=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
5
6
7 vif
```

Variance inflation factor (VIF) is a measure of the amount of multicollinearity in a set of multiple regression variables.

	Features	vif
0	Unnamed: 0	6.898783
1	Total Stops	4.045795
2	Hour_Arr	4.496322
3	Hour_Dep	3.170518
4	Airline Name_Air India	3.418638
5	Airline Name_Alliance Air	1.017568
6	Airline Name_IndiGo	4.000259
7	Airline Name_Others	2.122930
8	Airline Name_SpiceJet	1.834763
9	Airline Name_Vistara	4.525642
10	Source_Bangalore	3.203864
11	Source_Chennai	inf
12	Source_Hyderabad	inf
13	Source_Kochi	inf
14	Source_Kolkata	2.373186
15	Source_Mumbai	10.817592
16	Source_Patna	inf
17	Destination_Chennai	inf
18	Destination_Delhi	4.448584
19	Destination_Kochi	inf
20	Destination_Lucknow	inf

I have chosen 5 regression machine learning algorithms which is suitable for the pre-processed and cleaned data to train and test the dataset

i. Linear Regression

Linear Regression is the process of finding a line that best fits the data points available on the plot, so that we can use it to predict output values for inputs that are not present in the data set we have, with the belief that those outputs would fall on the line

ii. Random Forest Regressor

A random forest is a meta estimator that fits several classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

iii. Gradient Boosting Regressor

Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees.

iv. Decision Tree Regressor

A Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. Decision trees can handle both categorical and numerical data

v. XGBoost Regressor

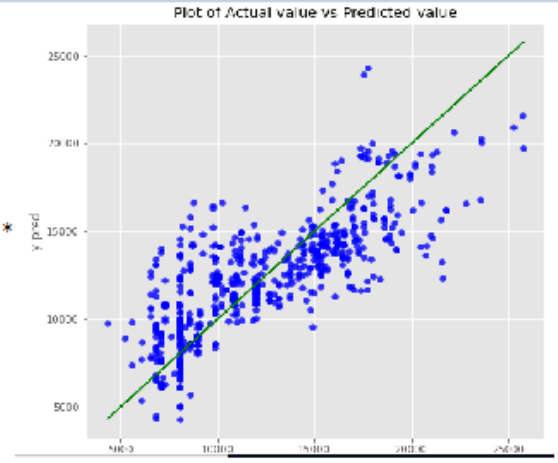
XGBoost is an implementation of gradient boosted decision trees designed for speed and performance that is dominative competitive machine learning.

- **Run and Evaluate selected models**

1. Linear Regression

```
lm = LinearRegression()
lm.fit(X_train, y_train)
y_pred = lm.predict(X_test)

print("******************Results******************")
print('The r2 score is:', r2_score(y_test, y_pred))
print('The mean absolute error', mean_absolute_error(y_test, y_pred))
print('The mean squared error', mean_squared_error(y_test, y_pred))
print('root mean square error', math.sqrt(mean_squared_error(y_test, y_pred)))
cv = cross_val_score(lm, X,y,cv=5)
print('The cross validation score', cv.mean())

print("\n***************XXXXXXXXXX******************")

#graph
plt.figure(figsize=(8,8), facecolor='w')
plt.scatter(y_test, y_pred,alpha=0.8,color='blue')
plt.plot(y_test, y_test, color='green')
plt.title('Plot of Actual value vs Predicted value')
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

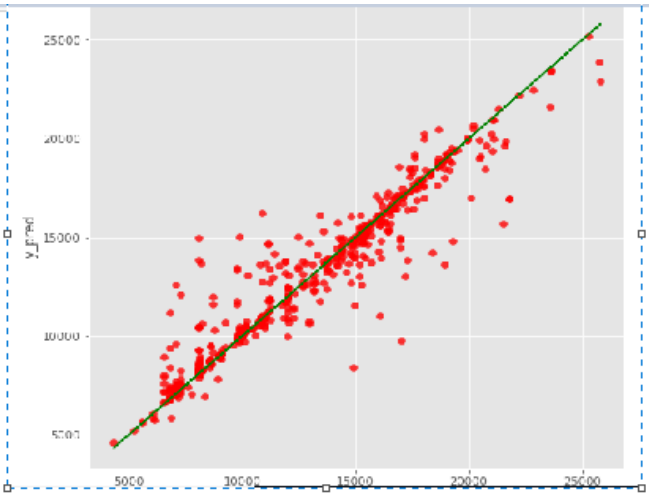
```
**********Results**********
The r2 score is: 0.6018174077951006
The mean absolute error 2105.8270400734705
The mean squared error 7372398.583639456
root mean square error 2715.2161209818005
The cross validation score 0.4406576627148519

**********XXXXXXXXXX**********
```



2. Random Forest

```
]:
1  rand_for = RandomForestRegressor()
2  rand_for.fit(X_train, y_train)
3  y_pred = rand_for.predict(X_test)
4
5  print("**********Results**********")
6  print('The r2 score is:', r2_score(y_test, y_pred))
7  print('The mean absolute error', mean_absolute_error(y_test, y_pred))
8  print('The mean squared error', mean_squared_error(y_test, y_pred))
9  print('root mean square error', math.sqrt(mean_squared_error(y_test, y_pred)))
10 cv = cross_val_score(rand_for, X,y,cv=5)
11 print('The cross validation score', cv.mean())
12
13 print("\n**********XXXXXXXXXX**********")
14
15 #graph
16 plt.figure(figsize=(8,8), facecolor='w')
17 plt.scatter(y_test, y_pred,alpha=0.8,color='red')
18 plt.plot(y_test, y_test, color='green')
19 plt.title('Plot of Actual value vs Predicted value')
20 plt.xlabel("y_test")
21 plt.ylabel("y_pred")
22 plt.show()
```

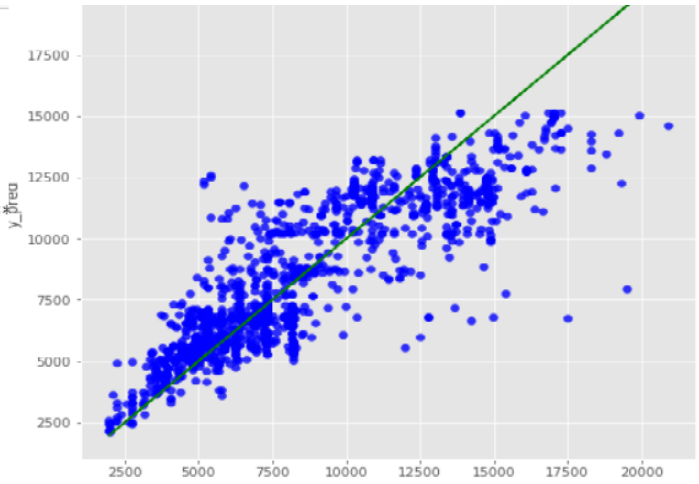
```
**********Results**********
The r2 score is: 0.9152285657419693
The mean absolute error 620.5987399030695
The mean squared error 1569553.3006510502
root mean square error 1252.818143487334
The cross validation score 0.1751879784105317

**********XXXXXXXXXX**********
```



3. Gradient Boosting Regression

```
gbr = GradientBoostingRegressor()
gbr.fit(X_train, y_train)
y_pred = gbr.predict(X_test)

print("******************Results*******************")
print('The r2 score is:', r2_score(y_test, y_pred))
print('The mean absolute error', mean_absolute_error(y_test, y_pred))
print('The mean squared error', mean_squared_error(y_test, y_pred))
print('root mean square error', math.sqrt(mean_squared_error(y_test, y_pred)))
cv = cross_val_score(gbr, X,y,cv=5)
print('The cross validation score', cv.mean())

print("\n****************XXXXXXXXXX********************")

#graph
plt.figure(figsize=(8,8), facecolor='w')
plt.scatter(y_test, y_pred,alpha=0.8,color='blue')
plt.plot(y_test, y_test, color='green')
plt.title('Plot of Actual value vs Predicted value')
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

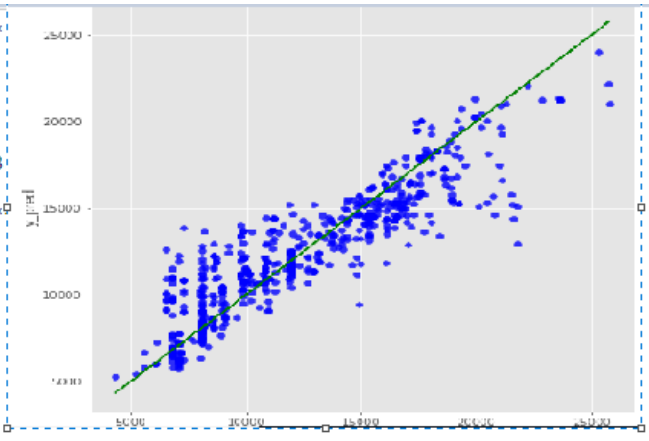
```
******************Results*******************
The r2 score is: 0.7414329844252934
The mean absolute error 1400.4772471554022
The mean squared error 3567299.2928513656
root mean square error 1888.7295446546511
The cross validation score 0.536161839157244

****************XXXXXXXXXX********************
```



4. XGBoost regression

```
]:
xgbr = XGBRegressor()
xgbr.fit(X_train, y_train)
y_pred = xgbr.predict(X_test)

print("******************Results*******************")
print('The r2 score is:', r2_score(y_test, y_pred))
print('The mean absolute error', mean_absolute_error(y_test, y_pred))
print('The mean squared error', mean_squared_error(y_test, y_pred))
print('root mean square error', math.sqrt(mean_squared_error(y_test, y_pred)))
cv = cross_val_score(xgbr, X,y,cv=5)
print('The cross validation score', cv.mean())

print("\n****************XXXXXXXXXX********************")

#graph
plt.figure(figsize=(8,8), facecolor='w')
plt.scatter(y_test, y_pred,alpha=0.8,color='blue')
plt.plot(y_test, y_test, color='green')
plt.title('Plot of Actual value vs Predicted value')
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

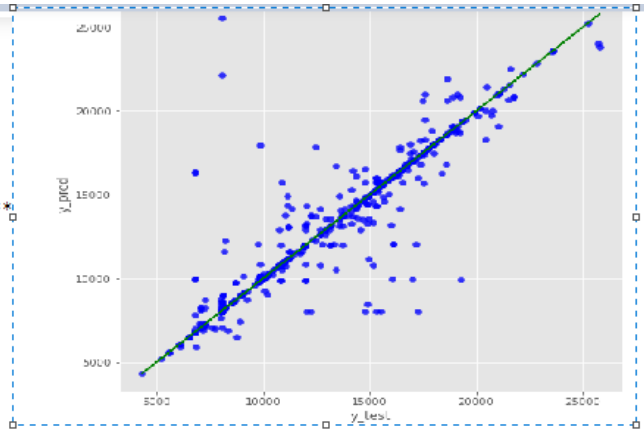
```
******************Results*******************
The r2 score is: 0.8074626360461423
The mean absolute error 1338.2404726244613
The mean squared error 3564852.4498546114
root mean square error 1888.081685164763
The cross validation score 0.10267835882638343

****************XXXXXXXXXX********************
```



5. Deciscion Tree Regression

```
dec_reg = DecisionTreeRegressor()
dec_reg.fit(X_train, y_train)
y_pred = dec_reg.predict(X_test)

print("******************Results********************")
print('The r2 score is:', r2_score(y_test, y_pred))
print('The mean absolute error', mean_absolute_error(y_test, y_pred))
print('The mean squared error', mean_squared_error(y_test, y_pred))
print('root mean square error', math.sqrt(mean_squared_error(y_test, y_pred)))
cv = cross_val_score(dec_reg, X,y,cv=5)
print('The cross validation score', cv.mean())

print("\n****************XXXXXXXXXX********************")

#graph
plt.figure(figsize=(8,8), facecolor='w')
plt.scatter(y_test, y_pred,alpha=0.8,color='blue')
plt.plot(y_test, y_test, color='green')
plt.title('Plot of Actual value vs Predicted value')
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```

```
******************Results********************
The r2 score is: 0.8491222935693841
The mean absolute error 571.43295638126
The mean squared error 2793518.8804523423
root mean square error 1671.3823262354854
The cross validation score -0.2518368556425753

****************XXXXXXXXXX********************
```



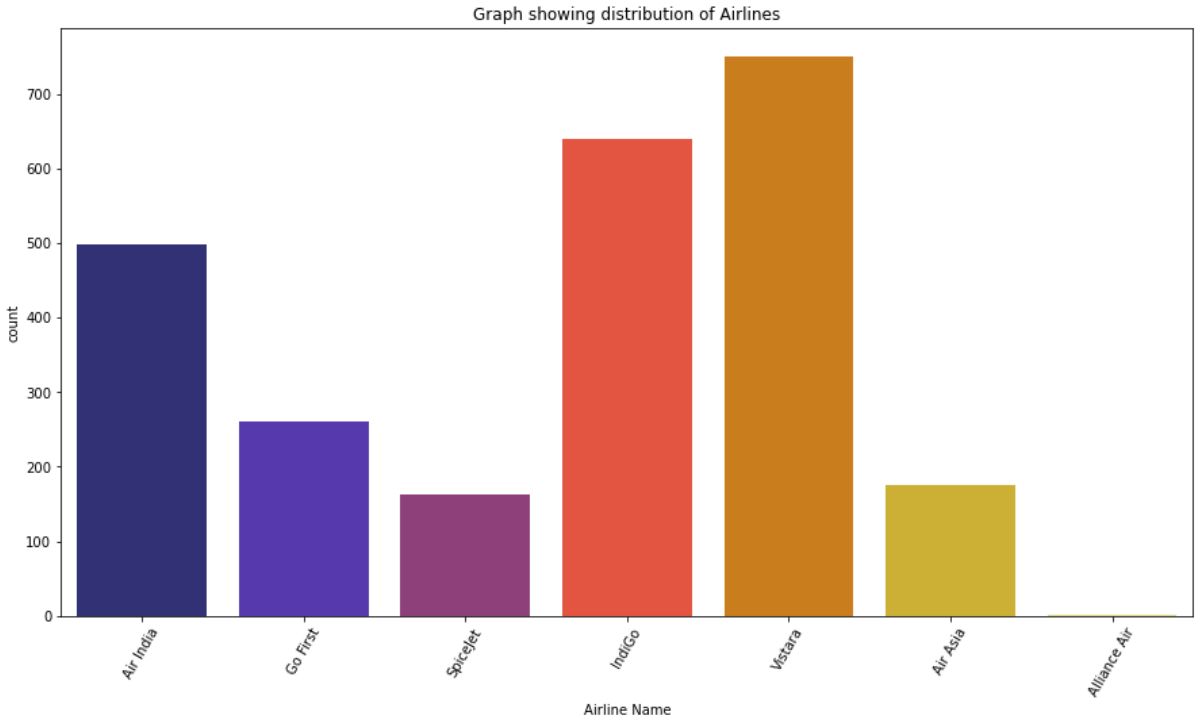
- **Key Metrics for success in solving problem under consideration**

  A key/evaluation metric quantifies the performance of a predictive model This typically. Following are the metrics I have used to evaluate the model performance.

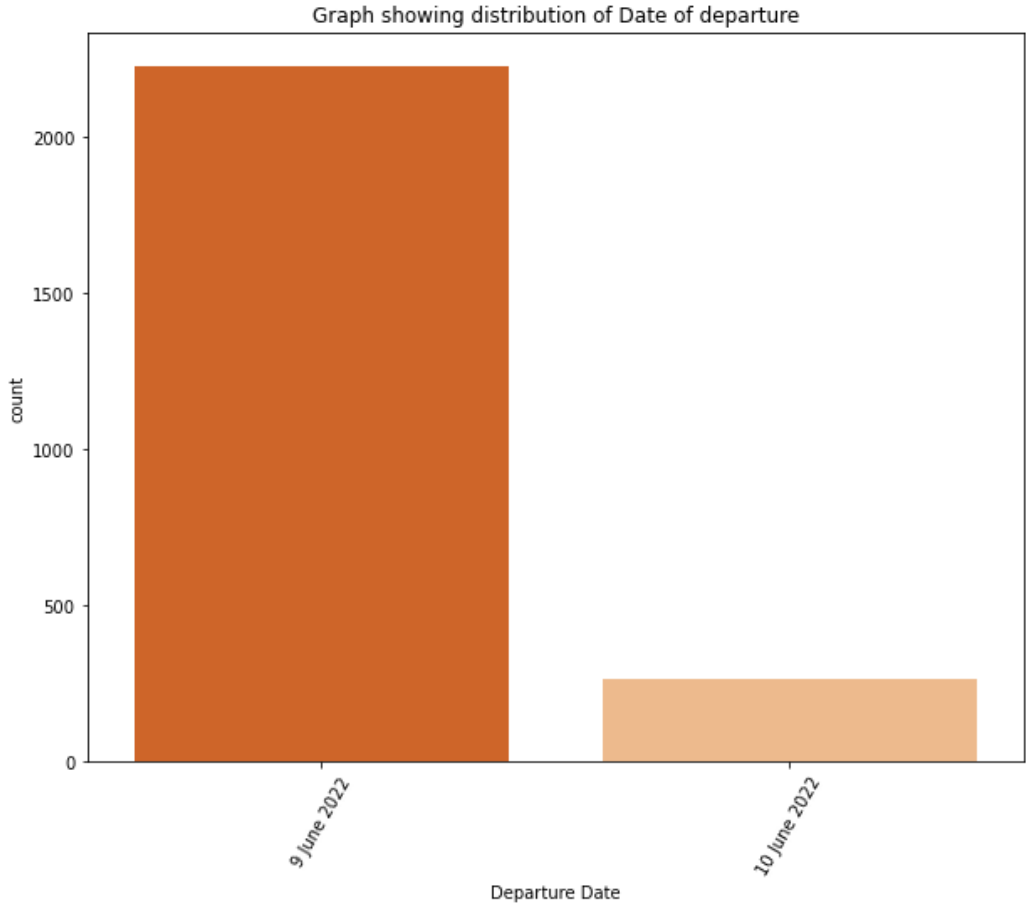
  I have used R2 score, mean absolute error, mean squared error and root mean squared error.

- **Visualizations**

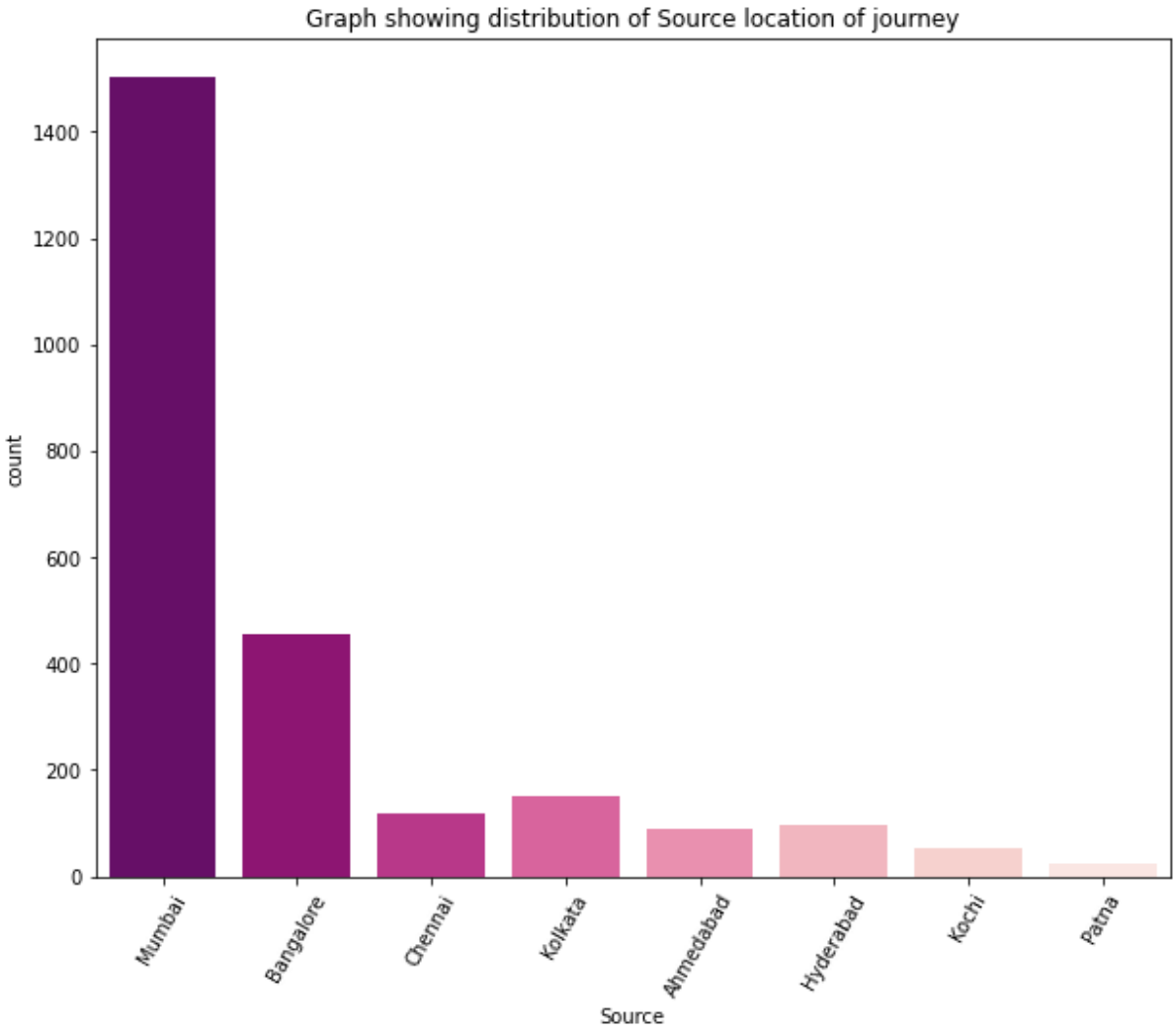
  We have used matplotlib and seaborn to interpret the relationship, we have plotted the graph using histogram to know how the data is distributed and box plot is used to check the outliers present and how is variance spread around the mean of the data.



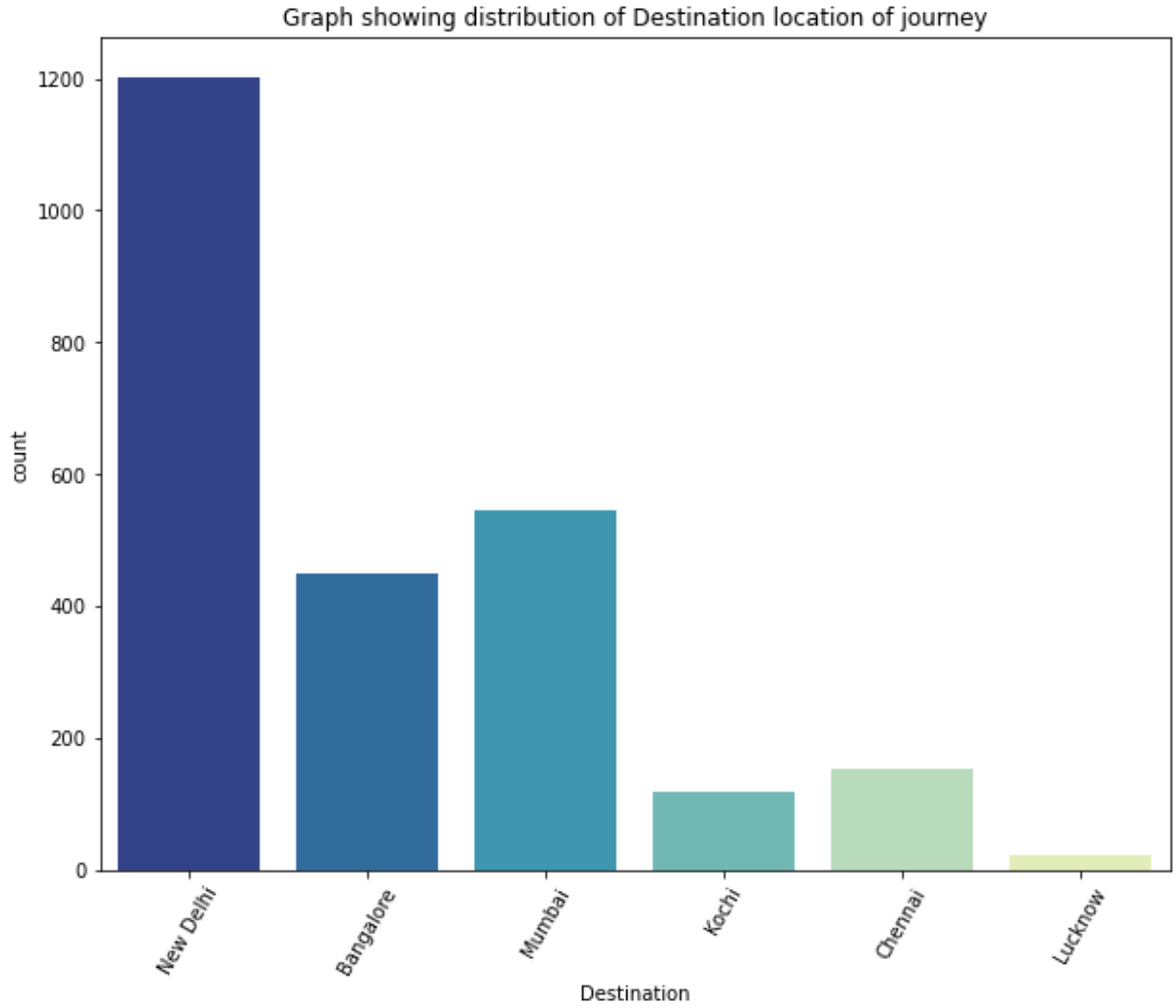
Indigo flights and Vistara constitutes nearly 50% of total flights compared to other flights used.



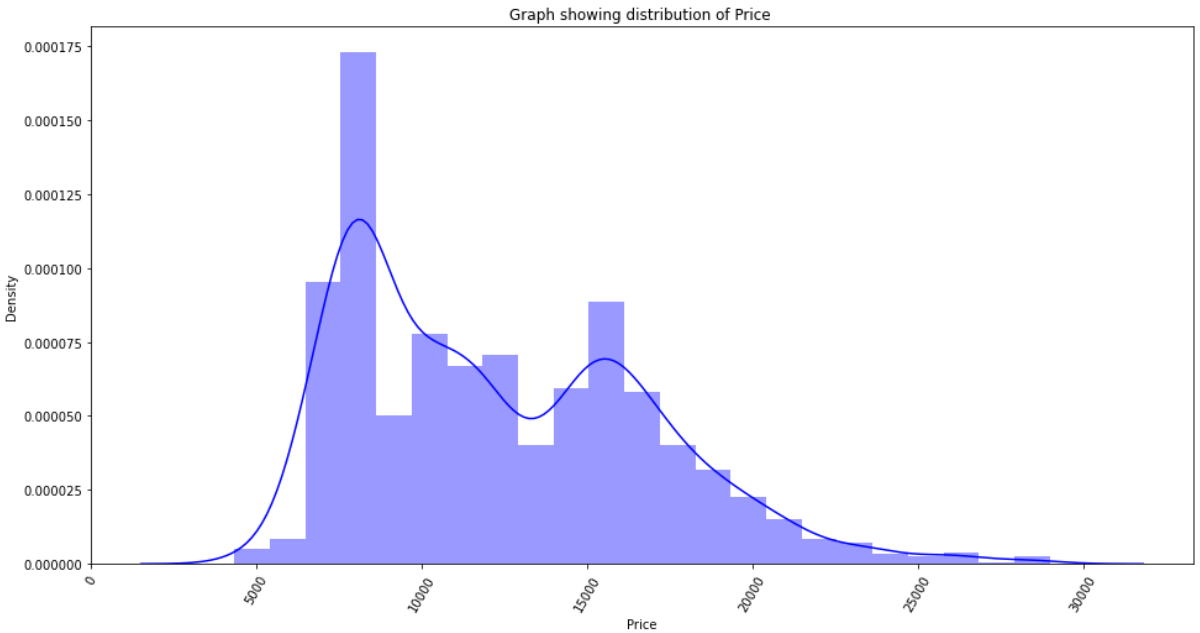
The number of flights are distributed normally except Weekends or holidays over the period.
They hospitality of the flight, pricing ranges may be considered for this high number of usage.



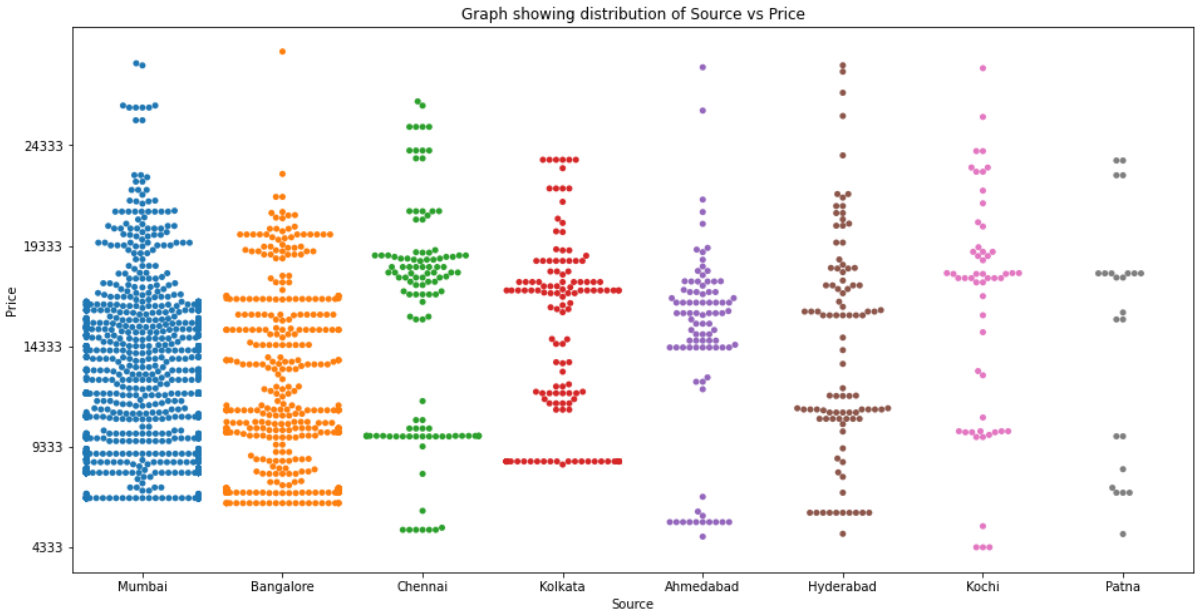
Nearly 50 % of total flight travel is observed from Mumbai and Banglore



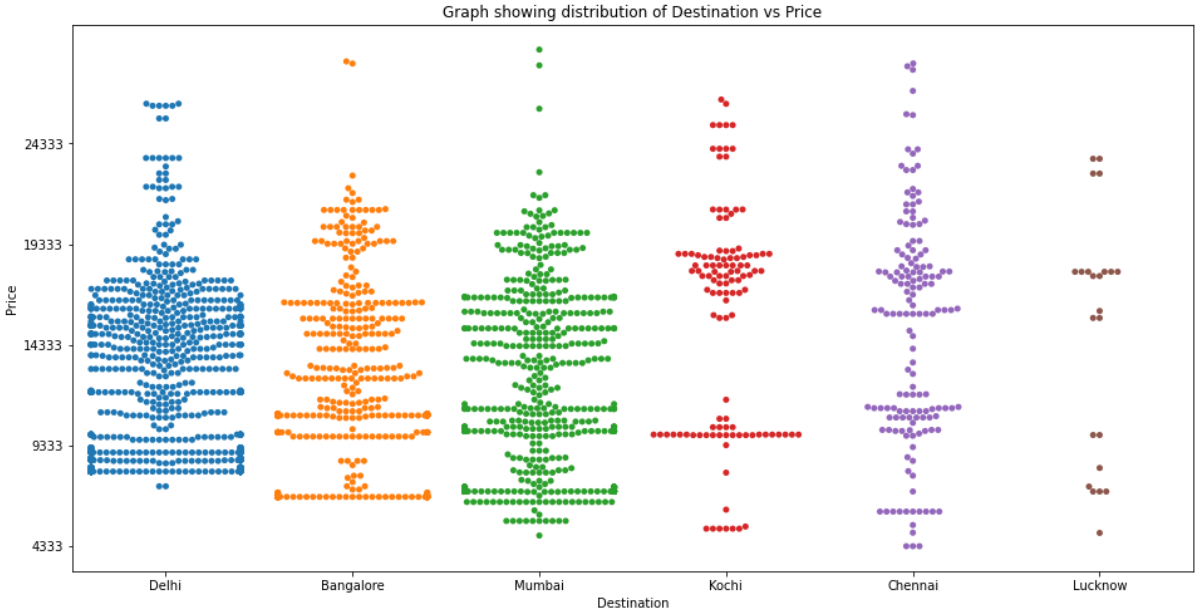
Close to 50% of the flights travelling have New Delhi as destination.



The price range was varied in between 5000 to 20000 on average; some luxury Business class flights are having up to 30000 cost are also seen.



As per the observations, no matter of date of booking, we can expect pricings are to be higher during the weekends, one-month prior of the expected date of journey can reduce the price of tickets during weekdays.



- **Interpretation of the Results**

  We have trained several models above for the dataset we had prepared, and we got different results for different algorithm. For the selected test data set, output of the model is plotted across the test dataset. Graph shows the comparative study of original values and predicted results. By the analysis of the results obtained from the algorithm such as Linear Regression, Random Forest, Decision Tree, Gradient Boosting, XG

Boosting gives the predicted values of the fare to purchase the flight ticket at the right time.

Random Forest Regressor gives 82.5% accuracy.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  To evaluate the conventional algorithm, a dataset is built for various routes in India and studied a trend of price variation for the period of limited days. 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. Data is collected from the websites which sell the flight tickets so only limited information can be accessed. The values of R-squared obtained from the algorithm give the accuracy of the model.

- ## Learning Outcomes of the Study in respect of Data Science

  1. This project has demonstrated the importance of having large dataset for training and testing the machine learning model.
  2. Through data cleaning we were able to remove unnecessary columns and outliers from our dataset due to which our model would have suffered from overfitting or under fitting.

3. Through different powerful tools of visualization, we were able to analyse and interpret different hidden insights about the data.
4. Build Models, since it was a supervised regression problem, I built 5 models to evaluate performance of each of them: a. Linear Regression b. Random Forest c. Decision Tree d. XGboost regressor e. Gradient Boosting.

• Limitations of this work and Scope for Future Work

In the future, if more data could be accessed such as the current an availability of seats, the predicted results will be more accurate. One can focus on collection of real time customer-oriented data which can be useful for EDA. And more inference can be provided based on the analysis.