Flight Price Prediction Project

Submitted by:

Harsh Prasad

ACKNOWLEDGMENT

Working on the Flight Price prediction project was interesting. I learnt a lot from this project, especially, extracting data from multiple sources and then cleaning them in the model building phase. The data set was collected from open-source websites like yatra.com, ixigo and makemytrip. This data is used to build a model that will help in understanding the change in the flight price over time.

I would like to express my sincere gratitude to our internship coordinator, Ms. Sapna Verma, who has given me valuable time and given me a chance to learn something despite having a busy schedule and his great guidelines for internship.

INTRODUCTION

- Business Problem Framing

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. 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 in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases).

So, we have to work on a project where we have to collect data of flight fares with other features and work to make a model to predict fares of flights.

Data collection:

You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data better the model

You have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

Data Analysis:

After cleaning the data, you have to do some analysis on the data.

Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time?

What is the best time to buy so that the consumer can save the most by taking the least risk?

Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

Model building:

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

- Conceptual Background of the Domain Problem

Optimal timing for airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we majorly targeted to uncover underlying trends of flight prices in India using historical data and also to suggest the best time to buy a flight ticket. Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination. From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price.

From the customer point of view, determining the minimum price or the best time to buy a ticket is the key issue. The concept of "tickets bought in advance are cheaper" is no longer working. It is possible that customers who bought a ticket earlier pay more than those who bought the same ticket later. Moreover, early purchasing implies a risk of commitment to a specific schedule that may need to be

changed usually for a fee. The ticket price may be affected by several factors thus may change continuously. To address this, various studies were conducted to support the customer in determining an optimal ticket purchase time and ticket price prediction.

- Review of Literature

It is very difficult for the customer to purchase a flight ticket at the minimum price. For this several techniques are used to obtain the day at which the price of air ticket will be minimum. Most of these techniques are using sophisticated artificial intelligence (AI) research known as Machine Learning.

Utilizing AI models, [2] connected PLSR (Partial Least Square Regression) model to acquire the greatest presentation to get the least cost of aircraft ticket buying, having 75.3% precision. Janssen [3] presented a direct quantile blended relapse model to anticipate air ticket costs for cheap tickets numerous prior days take-off. Ren, Yuan, and Yang [4], contemplated the exhibition of Linear Regression (77.06% precision), Naive Bayes (73.06% exactness, SoftMax Regression (76.84% precision) and SVM (80.6% exactness) models in anticipating air ticket costs. Papadakis [5] anticipated that the cost of the ticket drops later on, by accepting the issue as a grouping issue with the assistance of Ripple Down Rule Learner (74.5 % exactness.), Logistic Regression with 69.9% precision and Linear SVM with the (69.4% exactness) Machine Learning models.

Gini and Groves [2] took the Partial Least Square Regression (PLSR) for developing a model of predicting the best purchase time for flight tickets. The data was collected from major travel journey booking websites from 22 February 2011 to 23 June 2011. Additional data were also collected and are used to check the comparisons of the performances of the final model.

Janssen [3] built up an expectation model utilizing the Linear Quantile Blended Regression strategy for San Francisco to New York course with existing every day airfares given by www.infare.com. The

model utilized two highlights including the number of days left until the take-off date and whether the flight date is at the end of the week or weekday. The model predicts airfare well for the days that are a long way from the take-off date, anyway for a considerable length of time close the take-off date, the expectation isn't compelling.

Wohlfarth [15] proposed a ticket buying time enhancement model dependent on an extraordinary pre-preparing step known as macked point processors and information mining systems (arrangement and bunching) and measurable investigation strategy. This system is proposed to change over heterogeneous value arrangement information into added value arrangement direction that can be bolstered to unsupervised grouping calculation. The value direction is bunched into gathering dependent on comparative estimating conduct. Advancement model gauge the value change designs. A tree based order calculation used to choose the best coordinating group and afterward comparing the advancement model.

A study by Dominguez-Menchero [16] recommends the ideal buying time dependent on nonparametric isotonic relapse method for a particular course, carriers, and timeframe. The model gives the most extreme number of days before buying a flight ticket. two sorts of the variable are considered for the expectation. One is the passage and date of procurement.value.

- Motivation for the Problem Undertaken

Travelling through flights has become an integral part of today's lifestyle as more and more people are opting for faster travelling options. The flight ticket prices increase or decrease every now and then depending on various factors like timing of the flights, destination, duration of flights. various occasions such as vacations or festive season. Therefore, having some basic idea of the flight fares before planning the trip will surely help many people

save money and time. In the proposed system a predictive model will be created by applying machine learning algorithms to the collected historical data of flights. This system will give people the idea about the trends that prices follow and also provide a predicted price value which they can refer to before booking their flight tickets to save money. This kind of system or service can be provided to the customers by flight booking companies which will help the customers to book their tickets accordingly.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

Mathematical, statistical and analytics modelling done in the project are as follows:

a. Linear regression:

Equation used for linear regression model is

$$Y = bX + a$$

linear regression uses a traditional slope-intercept form, here *a* and *b* are the coefficients that we try to "learn" and produce the most accurate predictions. *X* represents our input data and *Y* is our prediction.

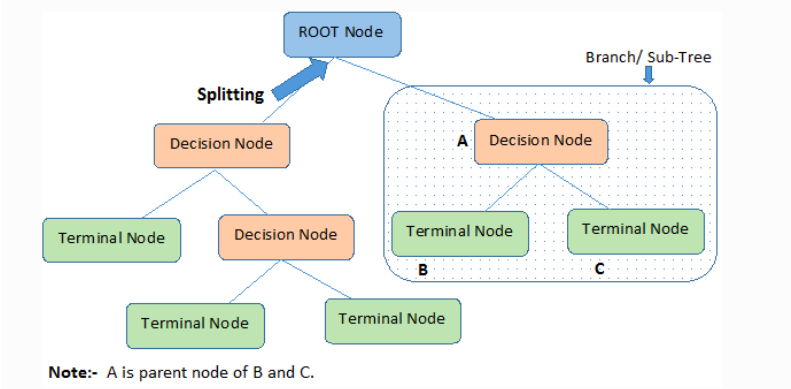
Since we have multiple variables present in the data set so the following equation was used:

$$Y(x_1, x_2, x_3) = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_0$$

Where x1, x2 and x3 are different independent variables and y is dependent variable. w1, w2 and w3 are similar to b and W0 is similar to a.

b. Decision Tree Regressor:

Equation used by this regressor model:



A **decision tree** is an efficient algorithm for describing a way to traverse a dataset while also defining a tree-like path to the expected outcomes. This branching in a tree is based on control statements or values, and the data points lie on either side of the **splitting node,** depending on the value of a specific feature.

1. **Information Gain**:

$$Info(D) = -\sum_{i}^{m} p_i \log_2(p_i),$$

$$Info_A(D) = \sum_{j=1}^{v} \frac{|D_j|}{|D|} \times Info(D_j).$$

pi is the probability that an arbitrary tuple in dataset D belongs to class Ci and is estimated by|Ci, D|/|D|. **Info(D) is simply the mean amount of information needed to identify the class/category of a data point in D.** A log function to base 2 is used, since in most cases, information is encoded in bits.

Information gain is calculated by:

$$Gain(A) = Info(D) - Info_A(D).$$

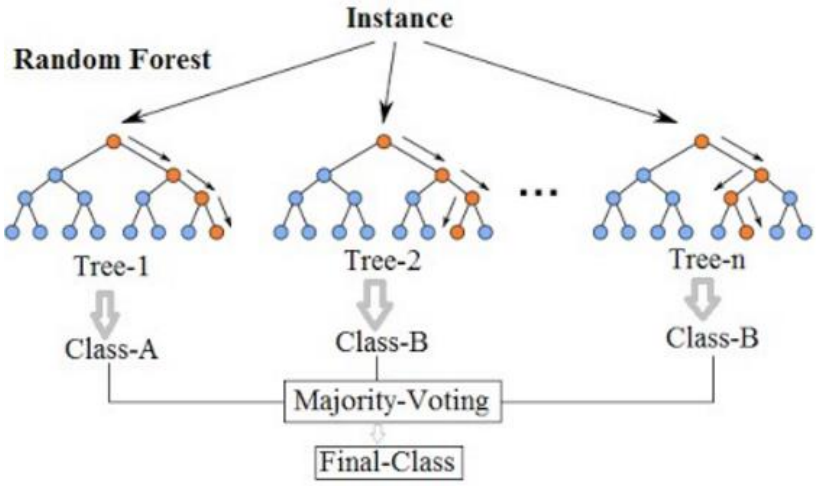
2. Gini Index:

$$Gini(D) = 1 - \sum_{i=1}^{m} p_i^2,$$

where pi is the probability that a tuple in D belongs to class Ci and is estimated by |Ci,D|/|D|. The sum is computed over m classes.

c. Random Forest Regressor:

Random-forest does both row sampling and column sampling with Decision tree as a base. Model h1, h2, h3, h4 are more different than by doing only bagging because of column sampling.



Random forest= DT (base learner) + bagging (Row sampling with replacement) + feature bagging (column sampling) + aggregation (mean/median, majority vote)

Steps taken to implement Random Forest regressor:

1. Suppose there are N observations and M features in the training data set. First, a sample from the training data set is taken randomly with replacement.
2. A subset of M features is selected randomly and whichever feature gives the best split is used to split the node iteratively.
3. The tree is grown to the largest.
4. The above steps are repeated and prediction is given based on the aggregation of predictions from n number of trees.

d. KNeighborsRegressor:

KNN uses a similarity metric to determine the nearest neighbors. This similarity metric is more often than not the Euclidean distance between our unknown point and the other points in the dataset. The general formula for Euclidean distance is:

$$\mathrm{d}(\mathbf{p}, \mathbf{q}) = \mathrm{d}(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

How KN Neigbors algorithm works:
For each entry in our test set we will iterate over all the entries in the training set and calculate the Euclidean distance. Thus, we are calculating the Euclidean distances between each entry in our test set and all entries in our training set. After the Euclidean distance has been calculated, we make a ‘distance’ column in our training set, shuffle the resulting set and sort them in ascending order of distance. Choose the first 3(if K=3) or first 5(if K=5) entries of the sorted dataset and find the mean of their ‘price’ column.

e. Gradient Boosting Regressor:

Input: training set $\{(x_i, y_i)\}_{i=1}^{n}$, a differentiable loss function $L(y, F(x))$, number of iterations M.

Algorithm:

1. Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma).$$

2. For m = 1 to M:
    1. Compute so-called *pseudo-residuals*:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

    2. Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
    3. Compute multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

    4. Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

3. Output $F_M(x)$.

Statistical modelling done during the project:

The following statistical functions were used in the project:

1. Mean, Median, Mode and Variance.

$$\text{Mean } \bar{x} = \frac{\sum xi}{N} \qquad \text{Variance} = \frac{\sum (xi - \bar{x})^2}{N}$$

$$\text{Median} = \begin{cases} \frac{(N+1)^{th}}{2} \text{ term; when N is even} \\ \frac{\frac{N^{th}}{2} \text{ term} + \left(\frac{N}{2} + 1\right) \text{term}}{2} \text{; when N is even} \end{cases}$$

Mode = The value in the data set that occurs most frequently

2. Standard Deviation

$$\sigma = \sqrt{\frac{\sum (X - \mu)^2}{n}}$$

where,

σ = population standard deviation
$\sum$ = sum of...
μ = population mean
n = number of scores in sample.

3. Percentile

$$P_k = k\left(\frac{n+1}{100}\right)^{th} item$$

$k = k^{th}$ *percentile*
$n =$ *no of items in given observation*

4. Z- score

$$Z = \frac{x - \mu}{\sigma}$$

Z = standard score
x = observed value
μ = mean of the sample
σ = standard deviation of the sample

5. Mean absolute error

$$MAE = \frac{\sum_{i=1}^{n} |y_i - x_i|}{n}$$

x_i = actual value n = sample size
y_i = predictions

6. Mean squared error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

MSE = mean squared error
n = number of data points
Y_i = observed values
$\hat{Y}_i$ = predicted values

- Data Sources and their formats

The Data was collected using selenium web scrapping technique from different websites like Yatra,ixigo and Makemytrip.

The following data has been collected:

Airline Name: This column contains the name of the Flights.

Date of journey: This column contains the date of the trip of the flights in the format (weekday, day month).

Source: This column contains the name of the city, from which the flight took off.

Destination: This column contains the name of the city where the flight landed.

Departure Time: This column contains the time at which the flight took off. The format is of 24 hrs, provided like (HH:MM),where HH stands for hour and MM stands for min.

Arrival Time: This column contains the time at which the flight arrived at the destination. The format is of 24 hrs, provided like (HH:MM), where HH stands for hour and MM stands for min.

Duration: This column contains the total hour taken by the flight to reach the destination. The format is (%h %min), where h stands for hours and min stands for minutes.

Total stops: This column contains the number of stops the flight took before reaching its final destination.

Price: This column contains the price of the ticket of the flight. The currency is in Indian rupees. (This is our target/dependent variable)

Attaching the snapshot of the original data:

```
dataset.head()
```

	Airline Name	Date of journey	Source	Destination	Departure Time	Arrival Time	Duration	Total Stops	Price
0	Air Asia	Wed, 6 Oct	New Delhi	Mumbai	12:40	20:15	7h 35m	1 Stop	5,953
1	Air Asia	Wed, 6 Oct	New Delhi	Mumbai	11:55	20:15	8h 20m	1 Stop	5,953
2	Air Asia	Wed, 6 Oct	New Delhi	Mumbai	08:00	16:35	8h 35m	1 Stop	5,953
3	Air Asia	Wed, 6 Oct	New Delhi	Mumbai	04:55	14:15	9h 20m	1 Stop	5,953
4	Air Asia	Wed, 6 Oct	New Delhi	Mumbai	05:20	20:15	14h 55m	1 Stop	5,953

Attaching the snapshot of the cleaned data:

	Airline Name	Source	Destination	Date of journey(day)	Date of journey(month)	Date of journey(weekday)	Dep_hour	Dep_min	Arr_hour	Arr_min	Arriving day	Duration_hr	Duration_min
0	air asia	new delhi	mumbai	6	10	5	12	40	20	15	same day	7	35
1	air asia	new delhi	mumbai	6	10	5	11	55	20	15	same day	8	20
2	air asia	new delhi	mumbai	6	10	5	8	0	16	35	same day	8	35
3	air asia	new delhi	mumbai	6	10	5	4	55	14	15	same day	9	20
4	air asia	new delhi	mumbai	6	10	5	5	20	20	15	same day	14	55

Total number of stops	Name of the first stop	Name of the second stop	Name of the third stop	Price
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953

All of these columns were in object format or string format, initially. So, we have to change it to the desirable formats.

```
In [236]: dataset.dtypes

Out[236]: Airline Name                object
          Source                      object
          Destination                 object
          Date of journey(day)         int64
          Date of journey(month)       int64
          Date of journey(weekday)     int64
          Dep_hour                     int64
          Dep_min                      int64
          Arr_hour                     int64
          Arr_min                      int64
          Arriving day                object
          Duration_hr                  int64
          Duration_min                 int64
          Total number of stops        int32
          Name of the first stop      object
          Name of the second stop     object
          Name of the third stop      object
          Price                        int32
          dtype: object
```

- Data Preprocessing Done

The following steps were taken in the data pre-processing phase:

1. Cleaning the data: The data which we extracted from different websites was containing lot of junk information so we removed those junk data and kept the useful information.

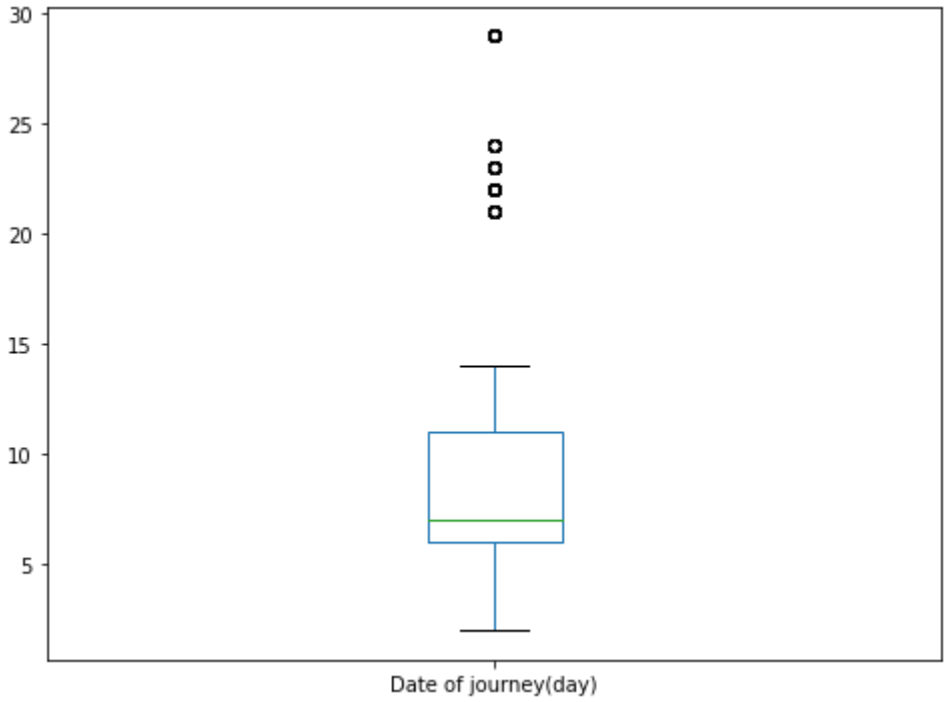
	Airline Name	Source	Destination	Date of journey(day)	Date of journey(month)	Date of journey(weekday)	Dep_hour	Dep_min	Arr_hour	Arr_min	Arriving day	Duration_hr	Duration_min
0	air asia	new delhi	mumbai	6	10	5	12	40	20	15	same day	7	35
1	air asia	new delhi	mumbai	6	10	5	11	55	20	15	same day	8	20
2	air asia	new delhi	mumbai	6	10	5	8	0	16	35	same day	8	35
3	air asia	new delhi	mumbai	6	10	5	4	55	14	15	same day	9	20
4	air asia	new delhi	mumbai	6	10	5	5	20	20	15	same day	14	55

Total number of stops	Name of the first stop	Name of the second stop	Name of the third stop	Price
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953
1	no stop available	no stop available	no stop available	5953

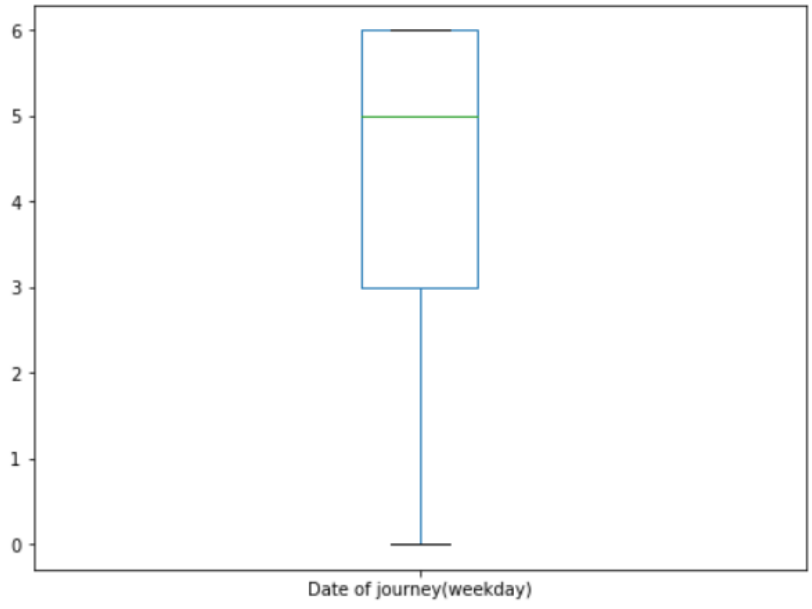
2. Standardization of data: After cleaning the data we observed that some columns contain same information but either in upper case or lower case. The Price column had two values one with numeric values and other with rupees sign. So, we used different methods to bring the data in desirable formats.

3. The dataset does not contain any null value.

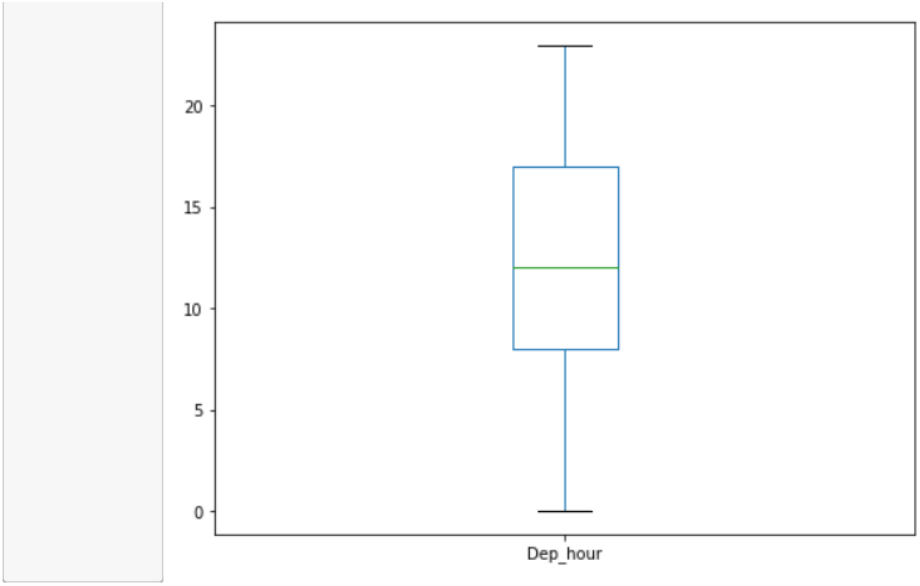
4. Removing Outliers from the dataset: So, first we checked if outliers are present in the data set or not. Since we have various numerical features so we'll perform this process on these features using box plot.



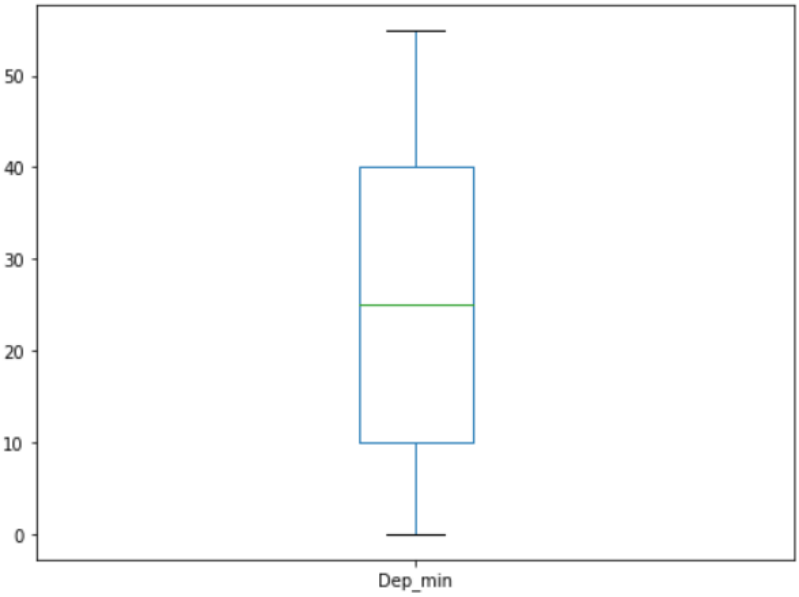
Observations: In the above box plot we can see that some outliers are present far from the upper whisker**.**



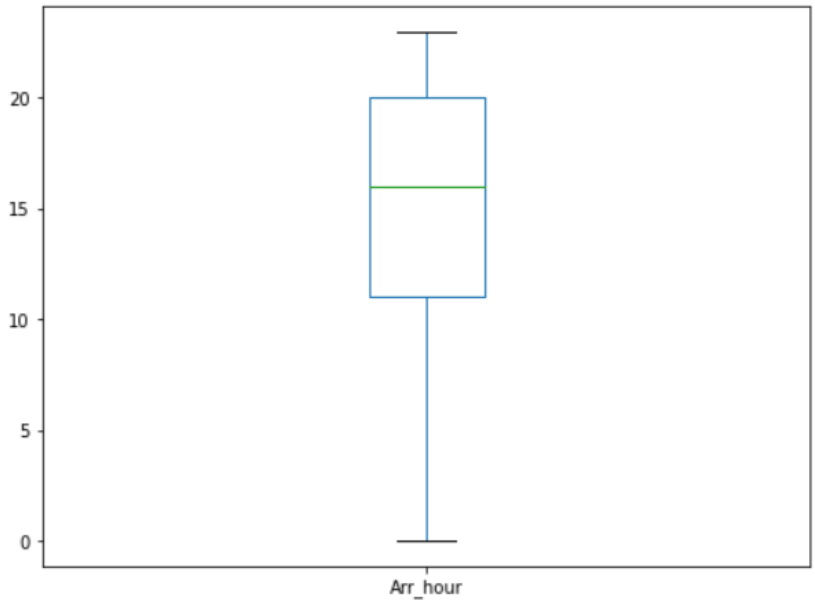
Observations : In the above box plot we can see that no outlier is present in this attribute.



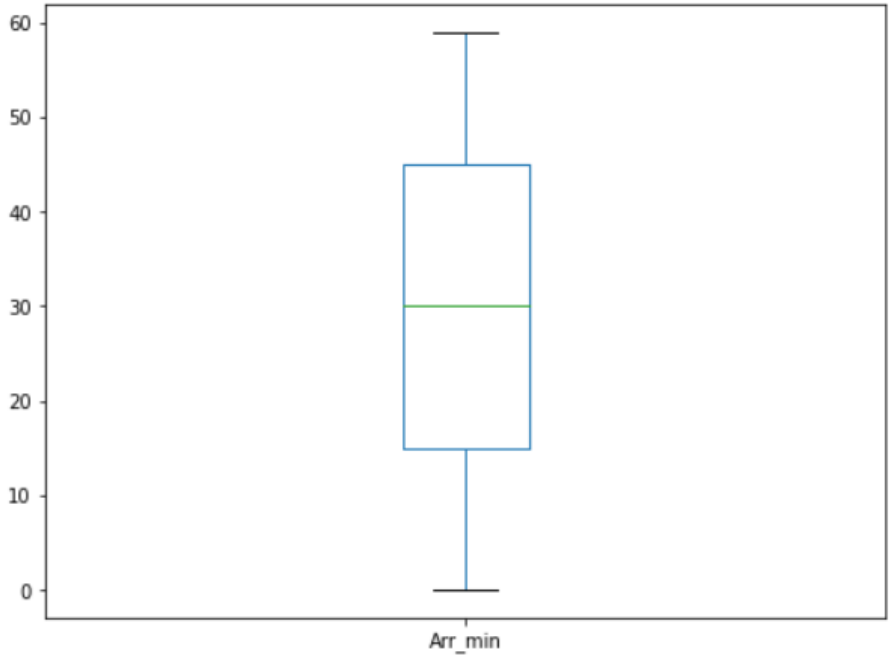
Observations : In the above box plot we can see that no outlier is present in this attribute.



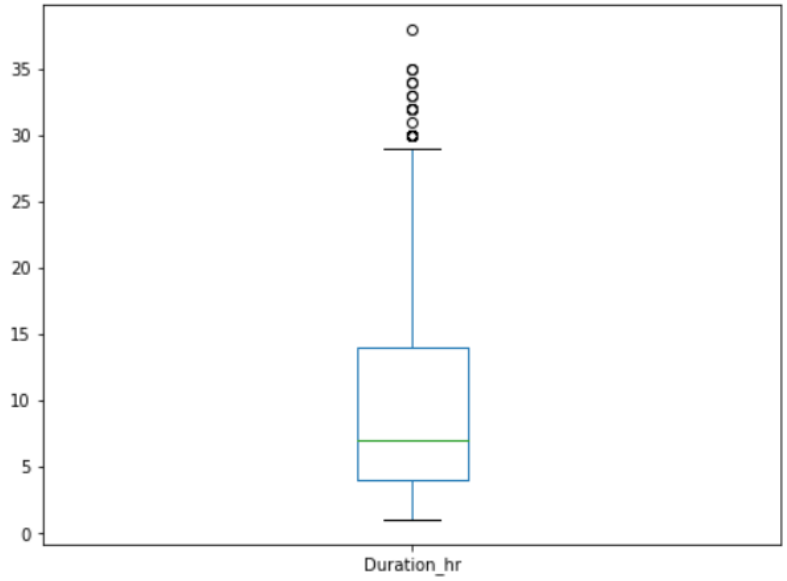
Observations : In the above box plot we can see that no outlier is present in this attribute.



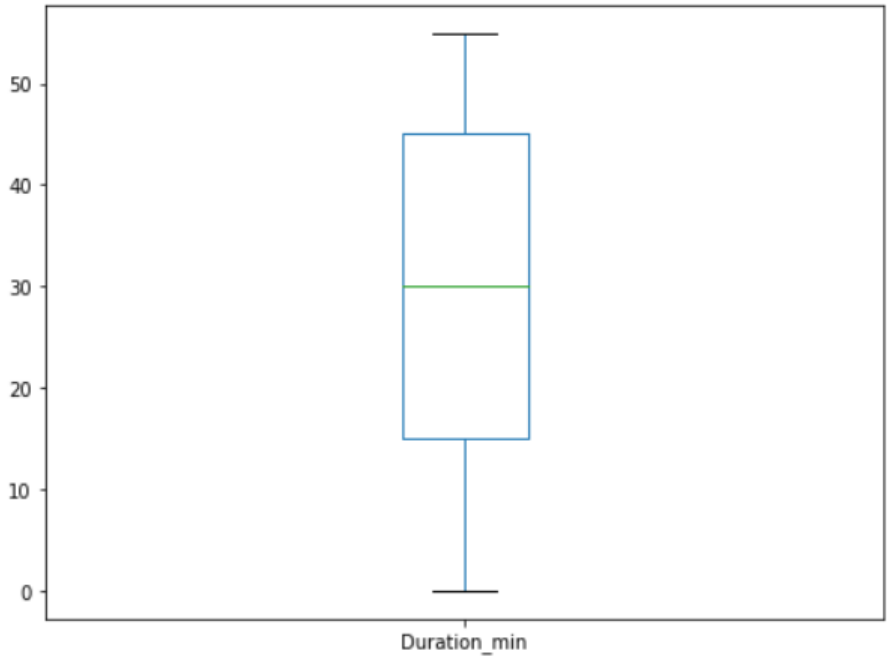
Observations : In the above box plot we can see that no outlier is present in this attribute.



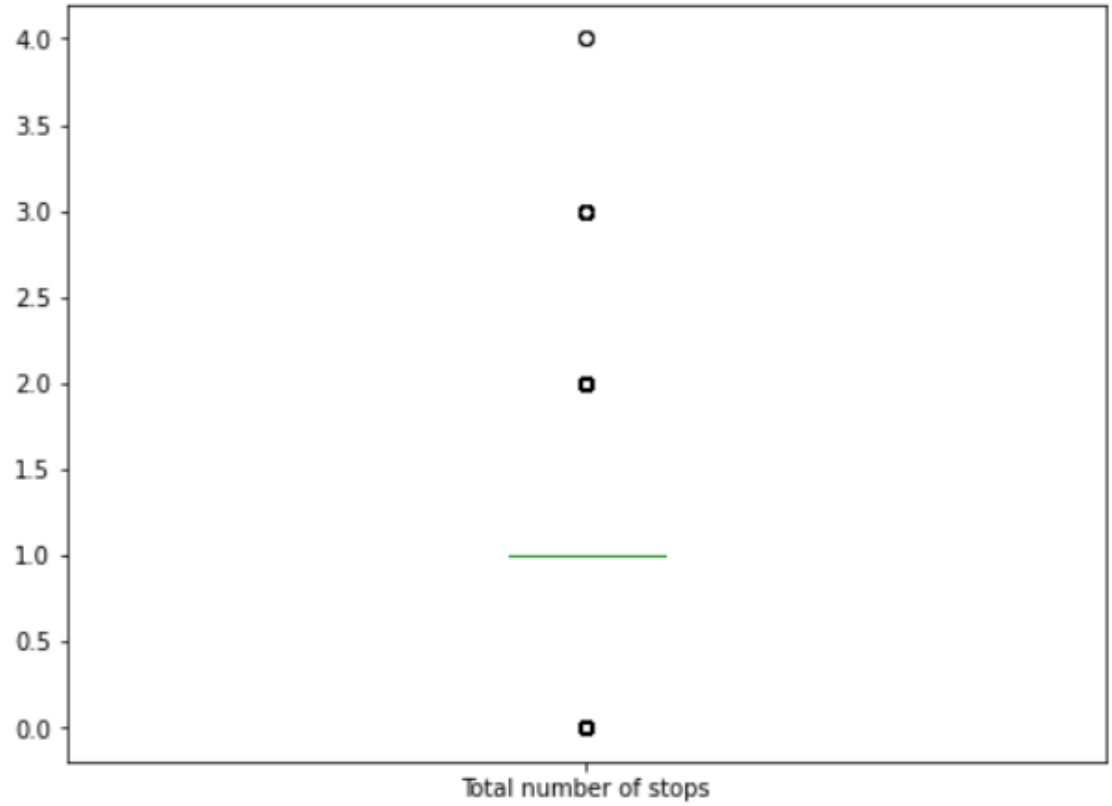
Observations : In the above box plot we can see that no outlier is present in this attribute.

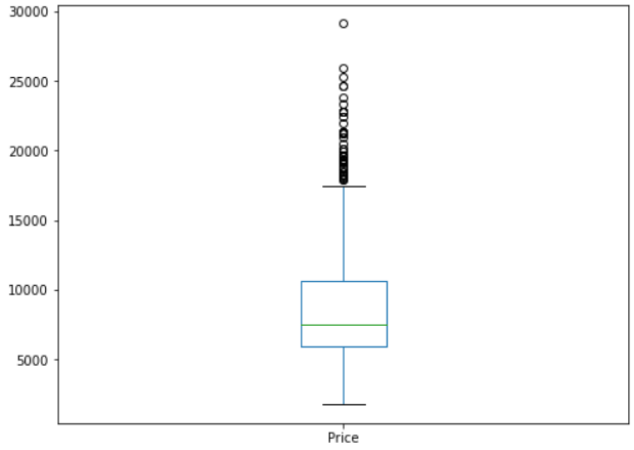


Observations : In the above box plot we can see that some outliers are present near the upper whisker.



Observations : In the above box plot we can see that no outlier is present in this attribute.





Observations : In the above box plot we can see that lots of outliers are present near the upper whisker and far from the upper whisker.

We observed that lots of outliers were present in the data set. So, we removed the outliers using Z Score function. But before that we encoded the categorical features using one hot encoder and ordinal encoder.

In [16]:
```
# As Airline is Nominal Categorical data, we will perform OneHotEncoder

Airline = dataset4[["Airline Name"]]
Airline = pd.get_dummies(Airline, drop_first =True)

Airline.head()
```

Out[16]:

	Airline Name_air india	Airline Name_airasia	Airline Name_airasia india	Airline Name_go first	Airline Name_indigo	Airline Name_spicejet	Airline Name_vistara
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

In [17]:
```
#source
Source =dataset4[["Source"]]
Source =pd.get_dummies(Source, drop_first=True)
Source.head()
```

Out[17]:

	Source_bangalore	Source_bengaluru	Source_goa	Source_hyderabad	Source_kochi	Source_kolkata	Source_mumbai	Source_new delhi	Source_varanasi
0	0	0	0	0	0	0	0	1	0
1	0	0	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	1	0
4	0	0	0	0	0	0	0	1	0

In [18]:
```
# As Destination is Nominal categorical data, we will perform OneHotEncoding

Destination =dataset4[["Destination"]]

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

Destination.head()
```

Out[18]:

	Destination_chennai	Destination_goa	Destination_hyderabad	Destination_jaipur	Destination_kolkata	Destination_lucknow	Destination_mumbai	Destination_n de
0	0	0	0	0	0	0	1	
1	0	0	0	0	0	0	1	
2	0	0	0	0	0	0	1	
3	0	0	0	0	0	0	1	
4	0	0	0	0	0	0	1	

Removing outliers from those attributes where outliers are present, excluding the target attribute ,i.e.,Price.

In [28]:
```
#removing outliers using z score

dataset_outliers_remove1 = dataset4[['Duration_hr','Date of journey(day)','Total number of stops']] # creating list of attributes


from scipy.stats import zscore

z = np.abs(zscore(dataset_outliers_remove1))
threshold = 3
print(np.where(z>3))
```

```
In [29]: for i in dataset_outliers_remove1:
             dataset4[i] = dataset_outliers_remove1[i]

         dataset_new2 = dataset4[(z<3).all(axis = 1)]
```

```
In [30]: dataset_new2.shape
Out[30]: (2101, 36)
```

```
In [32]: data_lose = ((dataset.shape[0] - dataset_new2.shape[0])/dataset.shape[0])*100
         data_lose
Out[32]: 4.673321234119783
```

After removing the outliers, we observed that 4.67% percent of data was lost, which is acceptable.

Now, we'll be removing skewness from the data set for that we first split the data into dependent and independent variable.

```
In [33]: #let's split the data into dependent and independent variables
         X = dataset_new2.drop(['Price'],axis  = 1)
         Y = dataset_new2['Price']
```

```
In [34]: #checking for the skewness present in the features
         X.skew()
Out[34]: Date of journey(day)           1.634489
         Date of journey(month)         2.213688
         Date of journey(weekday)      -1.252206
         Dep_hour                       0.123079
         Dep_min                        0.119821
         Arr_hour                      -0.617594
         Arr_min                       -0.061688
         Arriving day                  -1.196398
         Duration_hr                    1.048226
         Duration_min                   0.015761
         Total number of stops         -0.043910
         Airline Name_air india         1.127187
         Airline Name_airasia           9.858827
         Airline Name_airasia india    10.989486
         Airline Name_go first          2.397970
         Airline Name_indigo            0.894153
         Airline Name_spicejet          3.477902
         Airline Name_vistara           1.539557
         Source_bangalore               1.688975
         Source_bengaluru               5.717143
```

Observations : The following Numerical features shows high skewness:

1. Date of journey(day)
2. Date of journey(month)
3. Date of journey(weekday)
4. Arr_hour
5. Duration_hr
6. Total number of stops

We will remove the skewness from the numerical features only and will try to keep the keep the skewness between -0.5 and +0.5

Then we removed the skewness from the features having skewness more than 0.5 and less than -0.5 by using power transformer function.

In [35]:
```
#power transformer to remove skewness
from sklearn.preprocessing import PowerTransformer
power = PowerTransformer()
dataset_skewness1 = ['Date of journey(day)','Date of journey(month)','Date of journey(weekday)','Arr_hour','Duration_hr',
                     'Total number of stops']
for i in dataset_skewness1:
    X[i] = power.fit_transform(X[[i]])
```

In [36]:
```
X.skew()
```

```
Airline Name_spicejet          3.477902
Airline Name_vistara           1.539557
Source_bangalore               1.688975
Source_bengaluru               5.717143
Source_goa                     8.340359
Source_hyderabad               5.046261
Source_kochi                   7.340206
Source_kolkata                 3.087562
Source_mumbai                  1.394355
Source_new delhi               0.415434
Source_varanasi                5.825460
Destination_chennai            5.046261
Destination_goa                4.518667
Destination_hyderabad          7.138870
Destination_jaipur            11.335944
Destination_kolkata            3.938190
Destination_lucknow            5.008609
Destination_mumbai             1.045795
Destination_new delhi          1.461452
dtype: float64
```

After removing the skewness, we scaled our dataset before start building our model.

5. Scaling the data: In this step we used scaling feature to standardize our data before training our model.

In [37]:
```
#creating another numerical dataset excluding Price(the target feature) and then scaling them
dataset_numerical2 = X[['Date of journey(day)','Date of journey(month)','Date of journey(weekday)','Dep_hour', 'Dep_min',
                        'Arr_hour', 'Arr_min', 'Duration_hr', 'Duration_min', 'Total number of stops']]
```

In [38]:
```
#scaling the data
#Using standard scaler to scale the data.
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
for i in dataset_numerical2:
    X[i]= sc.fit_transform(X[[i]])
```

In [39]:
```
X.head()
```

Out[39]:

	Date of journey(day)	Date of journey(month)	Date of journey(weekday)	Dep_hour	Dep_min	Arr_hour	Arr_min	Arriving day	Duration_hr	Duration_min	Total number of stops	Airline Name_air india	A Name_a
0	-0.433557	-0.460697	0.261025	-0.137321	0.846719	0.856893	-0.818815	2.0	0.013040	0.427754	0.253724	0	
1	-0.433557	-0.460697	0.261025	-0.331998	1.681127	0.856893	-0.818815	2.0	0.169748	-0.454431	0.253724	0	
2	-0.433557	-0.460697	0.261025	-0.916029	-1.378368	0.080234	0.331946	2.0	0.169748	0.427754	0.253724	0	
3	-0.433557	-0.460697	0.261025	-1.694737	1.681127	-0.286099	-0.818815	2.0	0.310891	-0.454431	0.253724	0	
4	-0.433557	-0.460697	0.261025	-1.500060	-0.265824	0.856893	-0.818815	2.0	0.862652	1.604002	0.253724	0	

For scaling the dataset we used imported standardscaler function from sklearn.preprocessing library.

After scaling the data, we can now build our model.

- Data Inputs- Logic- Output Relationships

To see the relationship of Input variables with the output(dependent) variables we used correlation matrix table and we got the following results:

	Airline Name	Source	Destination	Date of journey(day)	Date of journey(month)	Date of journey(weekday)	Dep_hour	Dep_min	Arr_hour	Arr_min	Arriving day
Airline Name	1.000000	0.002979	-0.001057	0.005489	0.044648	0.003468	0.049881	0.143493	0.044823	0.063565	0.172518
Source	0.002979	1.000000	-0.239511	-0.032636	-0.091357	0.025052	0.052653	-0.030215	0.039420	-0.002134	0.001273
Destination	-0.001057	-0.239511	1.000000	0.086116	0.231350	-0.299906	-0.022581	0.048232	-0.036575	-0.121430	0.063692
Date of journey(day)	0.005489	-0.032636	0.086116	1.000000	0.027262	-0.830012	0.016224	0.023311	-0.000723	-0.097197	0.137375
Date of journey(month)	0.044648	-0.091357	0.231350	0.027262	1.000000	-0.305420	0.035873	-0.041298	-0.093339	0.023282	0.238607
Date of journey(weekday)	0.003468	0.025052	-0.299906	-0.830012	-0.305420	1.000000	-0.002069	-0.021093	0.039378	0.100237	-0.200220
Dep_hour	0.049881	0.052653	-0.022581	0.016224	0.035873	-0.002069	1.000000	0.015022	0.032698	-0.046776	-0.386344
Dep_min	0.143493	-0.030215	0.048232	0.023311	-0.041298	-0.021093	0.015022	1.000000	0.030559	0.005789	0.063270
Arr_hour	0.044823	0.039420	-0.036575	-0.000723	-0.093339	0.039378	0.032698	0.030559	1.000000	-0.032626	0.385594
Arr_min	0.063565	-0.002134	-0.121430	-0.097197	0.023282	0.100237	-0.046776	0.005789	-0.032626	1.000000	0.008644
Arriving day	0.172518	0.001273	0.063692	0.137375	0.238607	-0.200220	-0.386344	0.063270	0.385594	0.008644	1.000000
Duration_hr	-0.242773	-0.053998	-0.106632	-0.051851	-0.181478	0.123021	0.057006	-0.077543	-0.024963	0.030404	-0.682351
Duration_min	-0.042525	-0.034892	-0.074952	-0.066341	0.058060	0.059909	0.020027	-0.033970	0.029780	0.050774	-0.028197
Total number of stops	-0.309757	-0.093385	-0.129756	-0.081208	-0.178710	0.156161	-0.091583	-0.052366	0.081162	0.054403	-0.349693
Name of the first stop	-0.020239	0.141759	-0.022055	-0.156149	-0.371998	0.219248	-0.018974	0.045057	0.005675	0.033904	-0.116616
Name of the second stop	0.051742	0.013557	-0.047364	-0.005254	-0.130264	0.040518	0.010936	0.034417	-0.007745	0.001497	-0.023513
Name of the third stop	-0.026635	0.001759	0.024381	0.008404	0.067054	-0.023967	-0.015103	-0.017716	0.021445	-0.035876	0.012104
Price	-0.073151	-0.[Mail]122	-0.206449	-0.164779	-0.378344	0.287385	-0.024870	0.005847	0.099593	0.067952	-0.345122

Arriving day	Duration_hr	Duration_min	Total number of stops	Name of the first stop	Name of the second stop	Name of the third stop	Price
0.172518	-0.242773	-0.042525	-0.309757	-0.020239	0.051742	-0.026635	-0.073151
0.001273	-0.053998	-0.034892	-0.093385	0.141759	0.013557	0.001759	-0.136122
0.063692	-0.106632	-0.074952	-0.129756	-0.022055	-0.047364	0.024381	-0.206449
0.137375	-0.051851	-0.066341	-0.081208	-0.156149	-0.005254	0.008404	-0.164779
0.238607	-0.181478	0.058060	-0.178710	-0.371998	-0.130264	0.067054	-0.378344
-0.200220	0.123021	0.059909	0.156161	0.219248	0.040518	-0.023967	0.287385
-0.386344	0.057006	0.020027	-0.091583	-0.018974	0.010936	-0.015103	-0.024870
0.063270	-0.077543	-0.033970	-0.052366	0.045057	0.034417	-0.017716	0.005847
0.385594	-0.024963	0.029780	0.081162	0.005675	-0.007745	0.021445	0.099593
0.008644	0.030404	0.050774	0.054403	0.033904	0.001497	-0.035876	0.067952
1.000000	-0.682351	-0.028197	-0.349693	-0.116616	-0.023513	0.012104	-0.345122
-0.682351	1.000000	0.024593	0.659609	0.065526	-0.010778	0.003966	0.504387
-0.028197	0.024593	1.000000	0.029712	-0.009332	-0.019301	0.027820	0.035240
-0.349693	0.659609	0.029712	1.000000	-0.050236	-0.092793	0.072398	0.521750
-0.116616	0.065526	-0.009332	-0.050236	1.000000	0.186967	-0.118471	0.219626
-0.023513	-0.010778	-0.019301	-0.092793	0.186967	1.000000	-0.554419	0.045106
0.012104	0.003966	0.027820	0.072398	-0.118471	-0.554419	1.000000	-0.025687
-0.345122	0.504387	0.035240	0.521750	0.219626	0.045106	-0.025687	1.000000

Using the above code, we got the correlation of all the features with each other. So, to get the correlation of input variables with output variable we used the following code and got the following result:

```
In [411]: corr_matrix['Price'].sort_values(ascending = False)
```

```
Out[411]: Price                        1.000000
          Total number of stops        0.521750
          Duration_hr                  0.504387
          Date of journey(weekday)     0.287385
          Name of the first stop       0.219626
          Arr_hour                     0.099593
          Arr_min                      0.067952
          Name of the second stop      0.045106
          Duration_min                 0.035240
          Dep_min                      0.005847
          Dep_hour                    -0.024870
          Name of the third stop      -0.025687
          Airline Name                -0.073151
          Source                      -0.136122
          Date of journey(day)        -0.164779
          Destination                 -0.206449
          Arriving day                -0.345122
          Date of journey(month)      -0.378344
          Name: Price, dtype: float64
```

After analysing the output, we made following observations:

As we saw the relationship of Price with other attributes, earlier in the heatmap. Total number of stops, Duration_hr,Date of journey(weekday),Name of the first stop shows positive correlation with Price attribute. Whereas, Destination, Arriving day and Date of journey(month) shows negative correlation with Price attribute.

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

  The following assumptions were made during the project:

  - In the column date of journey(weekdays), the numbers represent the following days of week, i.e., Friday is represented by 0, Saturday by 1, Sunday by 2, Monday by 3, Tuesday by 4, Wednesday by 5 and Thursday by 6.
  - The non-stop in total number of stops is replaced by 0.
  - In the column Airline name ‘air asia’ and ‘airasia’ are the same.
  - The name of the city Bengaluru and Bangalore are the same.
  - The date of scrapping the data from the web was 6th October 2021

- Hardware and Software Requirements and Tools Used

Hardware requirement:

Type of hardware	Hardware requirements
Hardware	Dual core Intel Pentium compatible processor or multiprocessor-based computer with a 2Ghz or greater processor • 64-bit system • Network interface card
Memory	4 GB or more

Software requirements:

Type of software	software requirements
Operating System	Windows 10
Web browser	chrome
Text Editor	Jupyter notebook

Also, we used following libraries and packages in the project:

In [1]:
```
import numpy as np
import pandas as pd
import sklearn
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error, mean_absolute_error
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

a.) NumPy- for scientific calculation in Python.
b.) Pandas- for data manipulation and analysis.
c.) Matplotlib- used to plot charts in Python.
d.) Sklearn – it provides efficient tools for machine learning and statistical modelling including classification, regression, clustering and dimensionality reduction via a consistence interface in Python.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

We first use the basic statistical methods, i.e., calculating the Mean, Median, Mode, Standard deviation, Variance, Minimum and Maximum value and Percentiles of the data set and made analysis on these factors.

To get the basic information of the dataset we used function like dataset.info ()

To check the null values present in the feature we used dataset.isnull().sum() function.

To get the datatypes of the features we used dataset.dtypes.

After statistical analysis and getting basic information of the dataset We saw there were lot of uncertainties present in the data set so to remove them we used following methods.

1.) **Z score**: to remove the outliers from the data set.

2.) **Power transformation**: to remove skewness from the data set.

To check the relationship among the feature we used different visualisation technique. We got lots of insights from these plots which helped us in choosing our features effectively.

After that we scaled our data so that our model does not get biased for a particular feature. After that we gave the data for training and testing.

- Testing of Identified Approaches (Algorithms)

  First, we split the data,

  Data Splitting:

  We used 80-20% train-test split of data.

  After that we used following Algorithms for testing and training:

  1. Linear Regression
  2. Decision Tree Regressor.
  3. Random Forest Regressor
  4. KNeighbors Regressor
  5. Gradient Boosting Regressor

- Run and evaluate selected models

1. **<u>Linear Regressor</u>**:

   Linear regression model assumes a linear relationship between the input variables (x) and the single output variable (y). More specifically, that y can be calculated from a linear combination of the input variables (x). When there is a single input variable (x), the method is referred to as simple linear regression. When there are multiple input variables, literature from statistics often refers to the method as multiple linear regression.

Input given to the model and output obtained:

In [115]:
```
#first using Linear regression to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absolute
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(x_train,y_train)
prdlr = lr.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdlr))
print("Mean Squared error :",mean_squared_error(y_test,prdlr))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdlr))
print("RMSE:" , np.sqrt(mean_squared_error(y_test,prdlr)))
```

```
Accuracy of the model : 0.703616215761686
Mean Squared error : 3337227.6218489194
Mean Absolute eroor : 1354.3287774893913
RMSE: 1826.8080418721938
```

Observations :

As we can see that the accuracy of the Linear regression model is 70.36% which is satisfactory.And the mean absolute error is 1354.32, which is quite low.It means that our model is learning quite efficiently.

Observations:

As we can see that the accuracy of the Linear regression model is 70.36% which is satisfactory. And the mean absolute error is 1354.32, which is quite low. It means that our model is learning quite efficiently.

2. **Decision Tree Regressor:**

Decision trees use multiple algorithms to decide to split a node into two or more sub-nodes. The creation of sub-nodes increases the homogeneity of resultant sub-nodes. In other words, we can say that the purity of the node increases with respect to the target variable. The decision tree splits the nodes on all available variables and then selects the split which results in most homogeneous sub-nodes.

Input given to the model and output obtained:

In [117]:
```
#first using DecisionTreeRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absol
from sklearn.tree import DecisionTreeRegressor
dtc = DecisionTreeRegressor()
dtc.fit(x_train,y_train)
prdtc = dtc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdtc))
print("Mean Squared error :",mean_squared_error(y_test,prdtc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdtc))
print("RMSE:" , np.sqrt(mean_squared_error(y_test,prdtc)))
```

```
Accuracy of the model : 0.642015090596614
Mean Squared error : 4030845.111638955
Mean Absolute eroor : 892.3800475059383
RMSE: 2007.696469000968
```

Observations :

As we can see that the accuracy of the Linear regression model is 64.20% which is low.And the mean absolute error is 892.38, which is quite low as compared to linear regression.Since the accuracy is low It means that our model is not learning efficiently.

Observations:

As we can see that the accuracy of the Linear regression model is 64.20% which is low. And the mean absolute error is 892.38, which is quite low as compared to linear regression. Since the accuracy is low It means that our model is not learning efficiently.

3. **Random Forest Regressor:**

Random forest is a bagging technique and not a boosting technique. The trees in random forests are run in parallel. There is no interaction between these trees while building the trees.

It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

Input given to the model and output obtained:

In [118]:
```
#first using RandomForestRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absol
from sklearn.ensemble import RandomForestRegressor
rfc = RandomForestRegressor()
rfc.fit(x_train,y_train)
prrfc = rfc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prrfc))
print("Mean Squared error :",mean_squared_error(y_test,prrfc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prrfc))
print("RMSE:" , np.sqrt(mean_squared_error(y_test,prrfc)))
```

```
Accuracy of the model : 0.8213869474168237
Mean Squared error : 2011150.5568200862
Mean Absolute eroor : 858.743960807601
RMSE: 1418.1503999294596
```

Observations :

As we can see that the accuracy of the Linear regression model is 82.14% which is a good accuracy.And the mean absolute error is 882.38, which is low as compared to previous regression models .Since the accuracy is good, it means that our model is learning efficiently.

Observations:

As we can see that the accuracy of the Linear regression model is 82.14% which is a good accuracy. And the mean absolute error is 882.38, which is low as compared to previous regression models .Since the accuracy is good, it means that our model is learning efficiently.

4. **KNNeighbors Regressor:**

The KNN algorithm uses ‘feature similarity’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set.

Input given to the model and output obtained:

In [119]:
```
#first using KNeighborsRegressor to check the accuracy of the model,with the help of r2 score,Mean squared error and mean absolu
from sklearn.neighbors import KNeighborsRegressor
KNN = KNeighborsRegressor(n_neighbors=5)
KNN.fit(x_train,y_train)
prKNN = KNN.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prKNN))
print("Mean Squared error :",mean_squared_error(y_test,prKNN))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prKNN))
print("RMSE:" , np.sqrt(mean_squared_error(y_test,prKNN)))
```

```
Accuracy of the model : 0.67062421346307
Mean Squared error : 3708711.5802375297
Mean Absolute eroor : 1203.7881235154393
RMSE: 1925.8015422772746
```

Observations :

As we can see that the accuracy of the Linear regression model is 67.06% which is low.And the mean absolute error is 1203.79, which is quite high as compared to other regression model.Since the accuracy is low It means that our model is not learning efficiently.

Observations:

As we can see that the accuracy of the Linear regression model is 67.06% which is low. And the mean absolute error is 1203.79, which is quite high as compared to other regression model. Since the accuracy is low It means that our model is not learning efficiently.

5. Gradient Boosting Regressor:

"Boosting" in machine learning is a way of combining multiple simple models into a single composite model. This is also why boosting is known as an additive model, since simple models (also known as weak learners) are added one at a time, while keeping existing trees in the model unchanged. As we combine more and more simple models, the complete final model becomes a stronger predictor. The term "gradient" in "gradient boosting" comes from the fact that the algorithm uses gradient descent to minimize the loss.

Input given to the model and output obtained:

In [120]:
```
#using boosting technique to build model
from sklearn.ensemble import GradientBoostingRegressor
gbc = GradientBoostingRegressor()
gbc.fit(x_train,y_train)
prdgbc = gbc.predict(x_test)
print("Accuracy of the model :",r2_score(y_test,prdgbc))
print("Mean Squared error :",mean_squared_error(y_test,prdgbc))
print("Mean Absolute eroor :",mean_absolute_error(y_test,prdgbc))
print("RMSE:" , np.sqrt(mean_squared_error(y_test,prdgbc)))
```

```
Accuracy of the model : 0.7806277774467636
Mean Squared error : 2470091.413578793
Mean Absolute eroor : 1079.6857585741122
RMSE: 1571.6524468147506
```

Observations :

As we can see that the accuracy of the Linear regression model is 78.06% which is satisfactory.And the mean absolute error is 1079.68, which is quite high as compared to other regression model .Also, our model is learning quite efficiently.

Observations:

As we can see that the accuracy of the Linear regression model is 78.06% which is satisfactory. And the mean absolute error is 1079.68, which is quite high as compared to other regression model .Also, our model is learning quite efficiently.

We concluded the result on three metrices that are:

R2 score, Mean Absolute error, Mean squared error.

- Key Metrics for success in solving problem under consideration

  The four main metrices used to predict accuracy that are, r2_score, Mean Absolute error, mean squared error and cross validation score.

1. Mean Squared Error:

   We used mean squared error as the metric because the mean squared error or mean squared deviation of an estimator measures the average of the squares of the errors—that is, the average squared difference between the estimated values and the actual value. It helps in selecting the model with least mean squared error.

2. Mean absolute Error:

   We used mean absolute error because mean absolute error is a measure of errors between paired observations expressing the same phenomenon. Examples of Y versus X include comparisons of predicted versus observed, subsequent time versus initial time, and one technique of measurement versus an alternative technique of measurement.it tells us the deviation between the actual values and the predicted values

3. R2 Score:

   We used r2 score or the coefficient of determination, denoted R^2 or r^2 a is the proportion of the variance in the dependent variable that is predictable from the independent variable. This tells the actual accuracy of the model.

4. Cross validation score:

We used cross validation score so that we can check whether there is biasness present in the model or not.

Cross validation score of different models are as follows:

In [48]:
```
#setting up cross validation parameter
from sklearn.model_selection import KFold

cv = KFold(n_splits=10, shuffle=True)
```

In [50]:
```
#Checking cross validation score for linear regression
from sklearn.model_selection import cross_val_score
cross_val1 = cross_val_score(LinearRegression(),X,Y,cv = cv)

print("Cross Validation score for Linear regression : ",cross_val1.mean())
```

```
Cross Validation score for Linear regression :  0.5855629802184067
```

In [53]:
```
model_accuracy(r2_score(y_test,prdlr),cross_val1.mean())
```

```
The difference between the accuracy and cross validation score is :  11.805323554327929
```

In [54]:
```
#calculating cross validation score of Random Forest regressor
cross_val2 = cross_val_score(RandomForestRegressor(),X,Y,cv = cv)
print("Cross Validation Score for Random Forest Regressor",cross_val2.mean())
```

```
Cross Validation Score for Random Forest Regressor 0.7219899242891957
```

In [55]:
```
model_accuracy(r2_score(y_test,prrfc),cross_val2.mean())
```

```
The difference between the accuracy and cross validation score is :  9.787062247582323
```

In [56]:
```
#calculating cross validation score of Decision Tree regressor

cross_val3 = cross_val_score(DecisionTreeRegressor(),X,Y,cv = cv)
print("Cross Validation Score for Decision tree regressor",cross_val3.mean())
```

```
Cross Validation Score for Decision tree regressor 0.5394133092658218
```

In [57]:
```
model_accuracy(r2_score(y_test,prdtc),cross_val3.mean())
```

```
The difference between the accuracy and cross validation score is :  10.664001296627855
```

In [58]:
```
#calculating cross validation score of KNeighbors Regressor
cross_val4 = cross_val_score(KNeighborsRegressor(),X,Y,cv = cv)
print("Cross Validation Score K Neighbors ",cross_val4.mean())
```

```
Cross Validation Score K Neighbors  0.5526569581869537
```

In [59]:
```
model_accuracy(r2_score(y_test,prKNN),cross_val4.mean())
```

```
The difference between the accuracy and cross validation score is :  11.79672552761163
```

In [61]:
```
#calculating cross validation score of Gradient boosting regressor

cross_val5 = cross_val_score(gbc,X,Y,cv = cv)
print("Cross validation score of Gradient Boosting classifier is ",cross_val5.mean())
```

```
Cross validation score of Gradient Boosting classifier is  0.6753002751338381
```

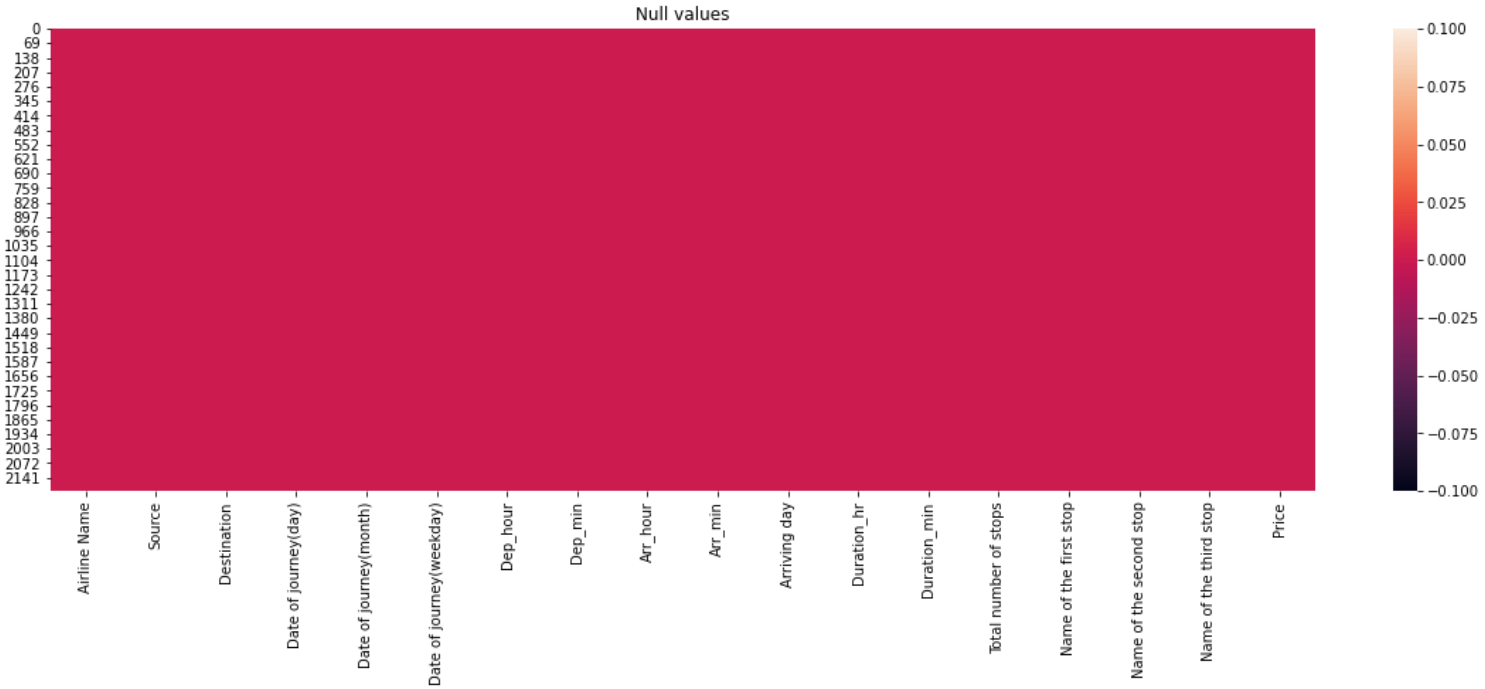
In [62]:
```
model_accuracy(r2_score(y_test,prdgbc),cross_val5.mean())
```

```
The difference between the accuracy and cross validation score is :  10.177420722547026
```

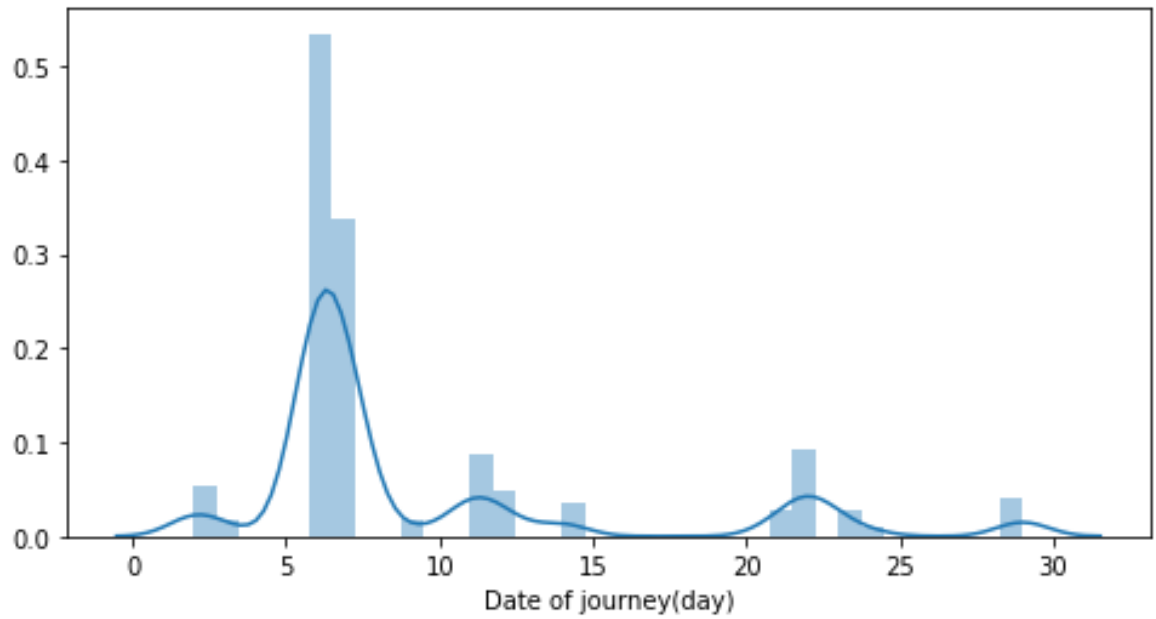
- Visualizations

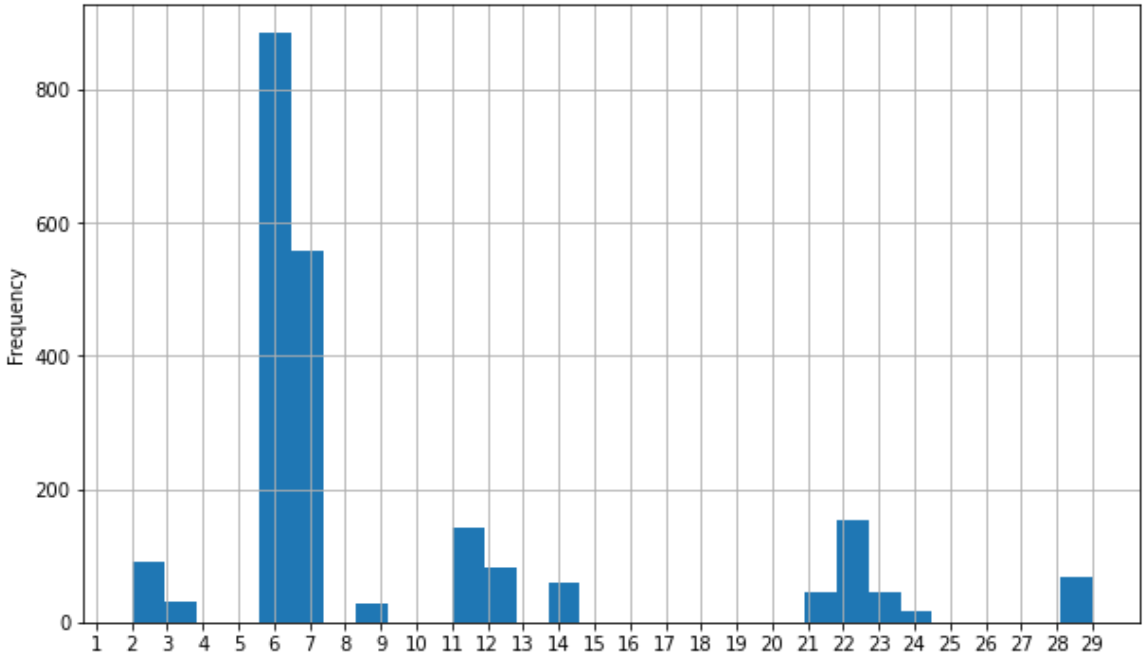
Visualizations and observations

Heatmap to check null values:

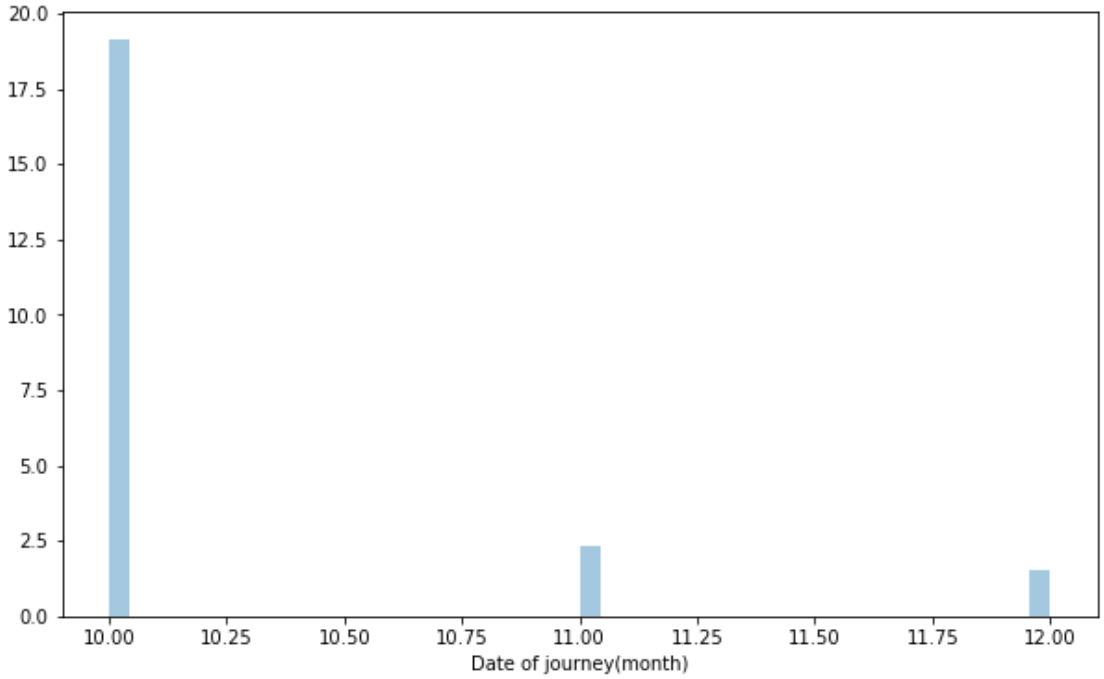


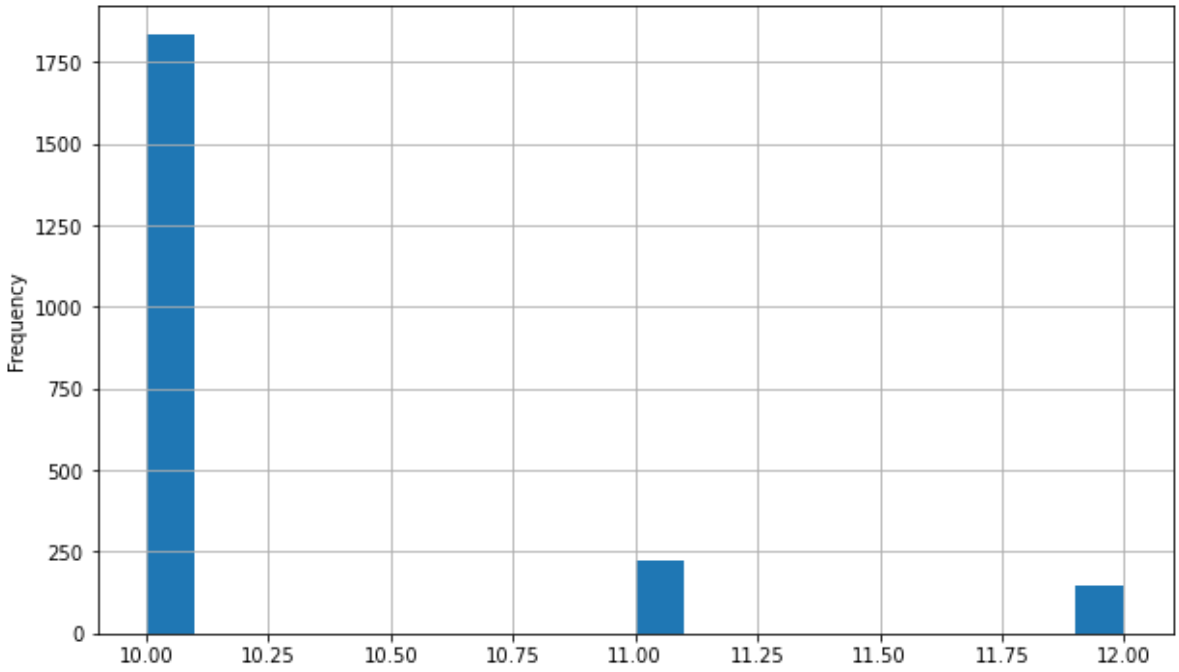
Observations: In the above heatmap, we can see that no null values are present.



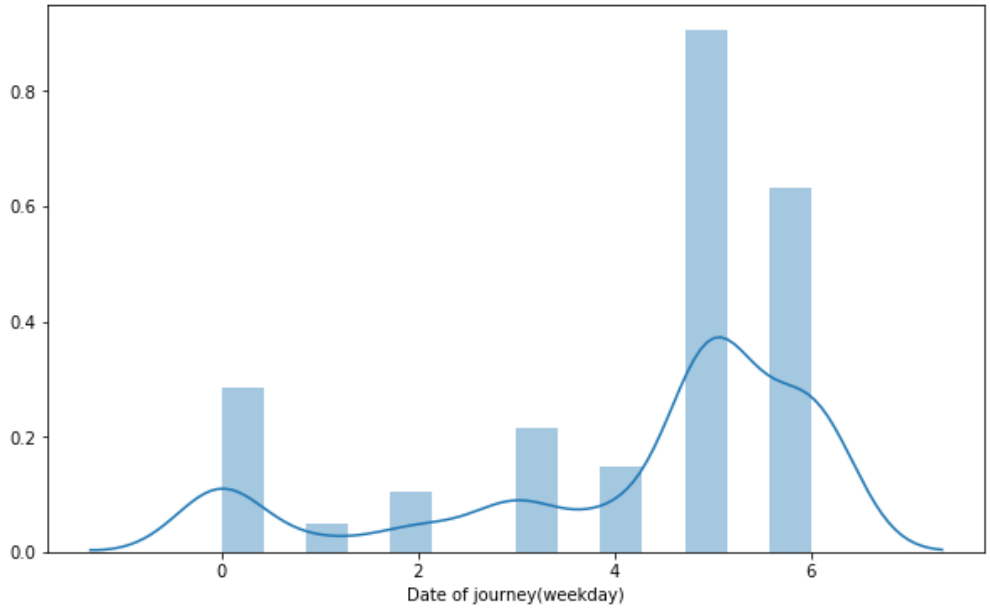


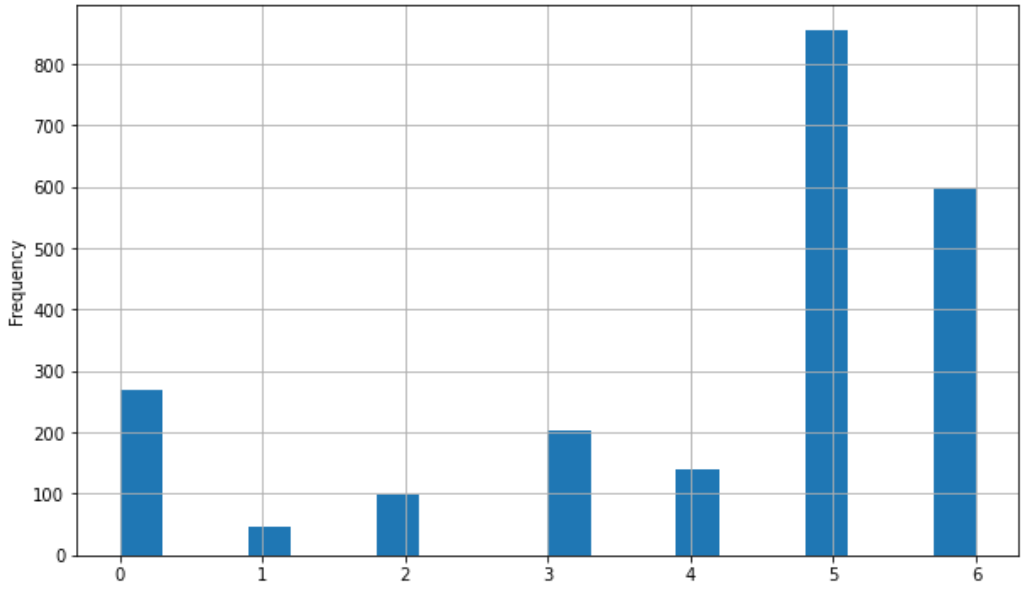
Observations: Most of the flights flew between the dates 5 and 7, i.e., around 1300 flights.



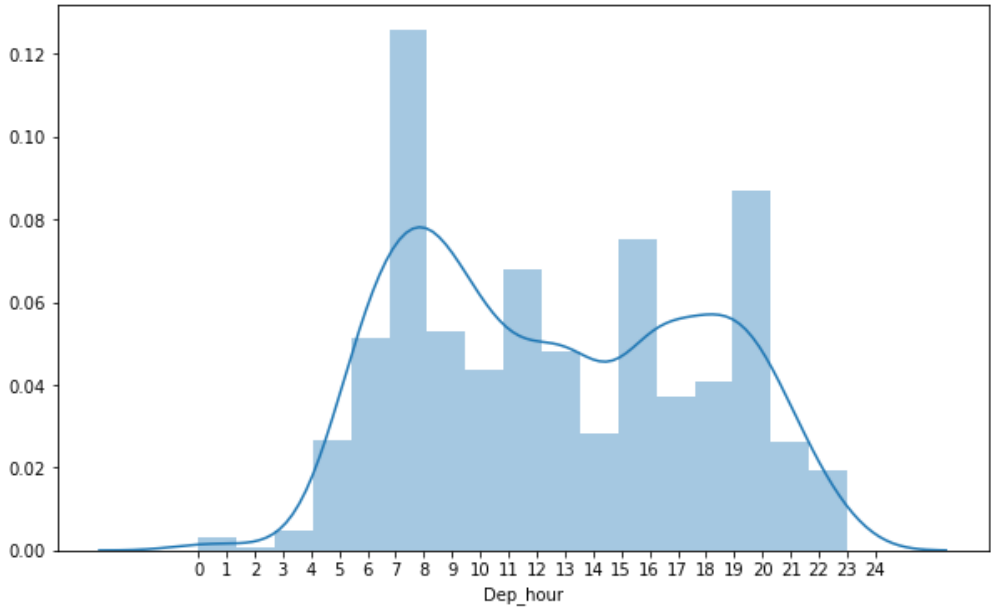


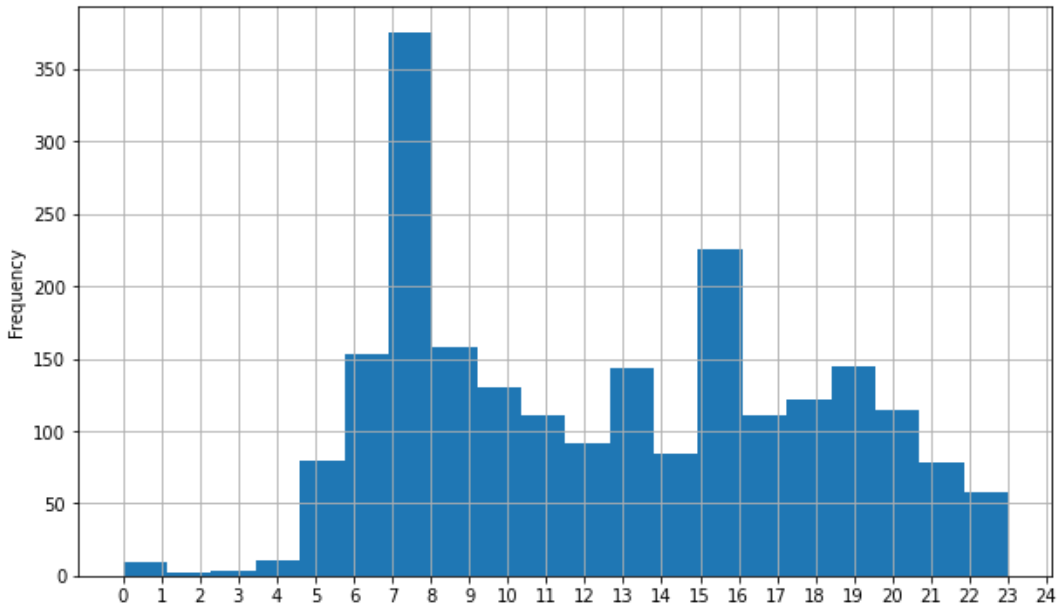
Observations: In the above distribution plot and histogram we can see that the flight flew between October (10) to December (12). More than 1750 flights flew in the month of October (10).



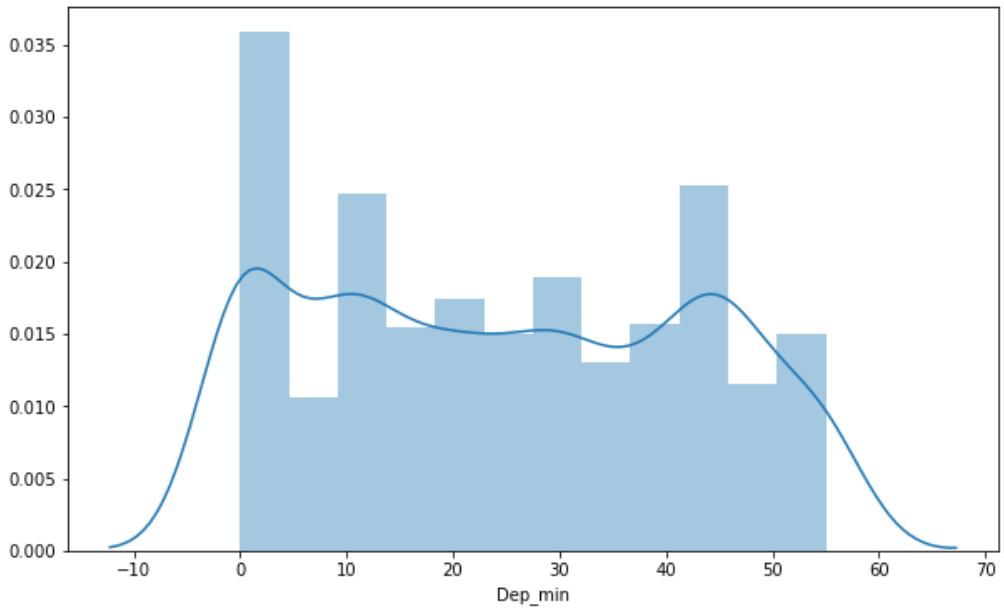


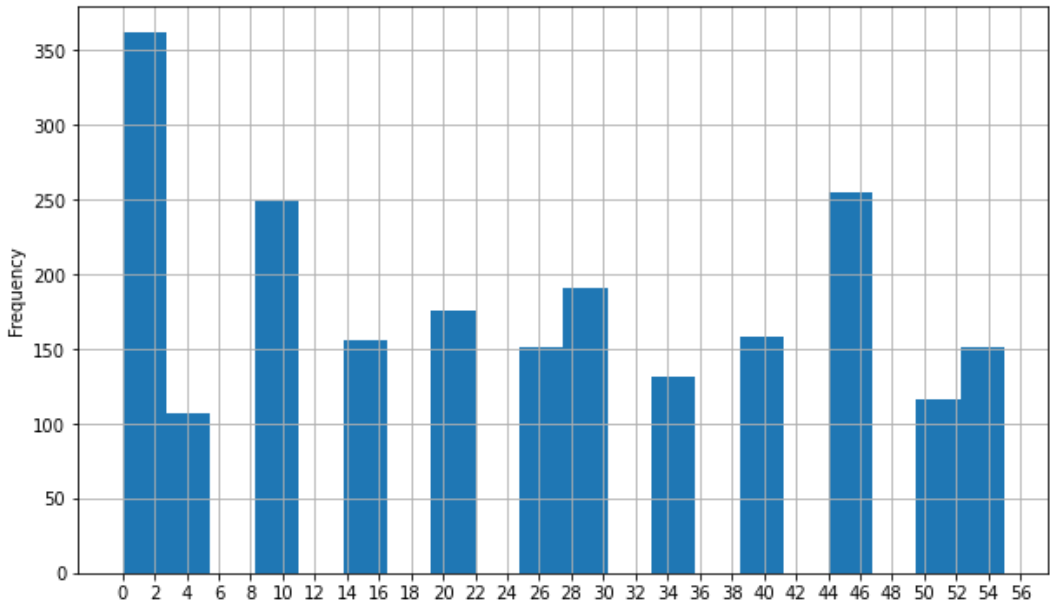
Observations: In the above distribution plot and histogram we can see that the graph attains spike on three days, i.e., Wednesday (5), Thursday(6) and Friday(0).Also we can see that, more than 800 flight flew on Wednesday.



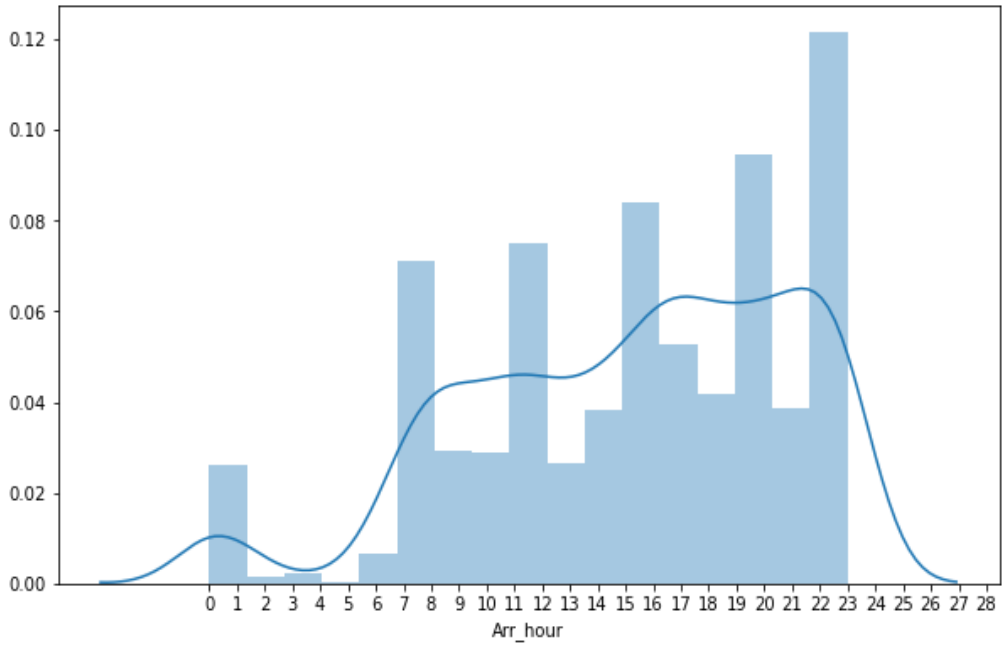


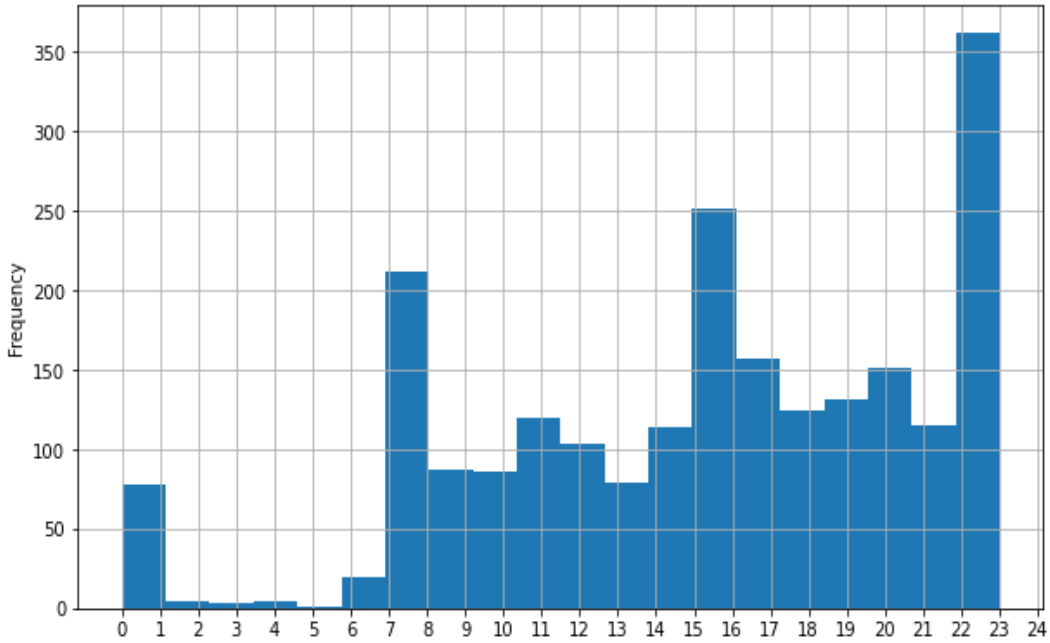
Observations: In the above distribution plot and histogram we can see that the data in the graph is spreaded. Also , the graph attains spike 3 times i.e. between 7 to 8 hours(highest) ,13 to 14 hours and 15 to 16 hours. Most of the flights departed from the airport during these hours.



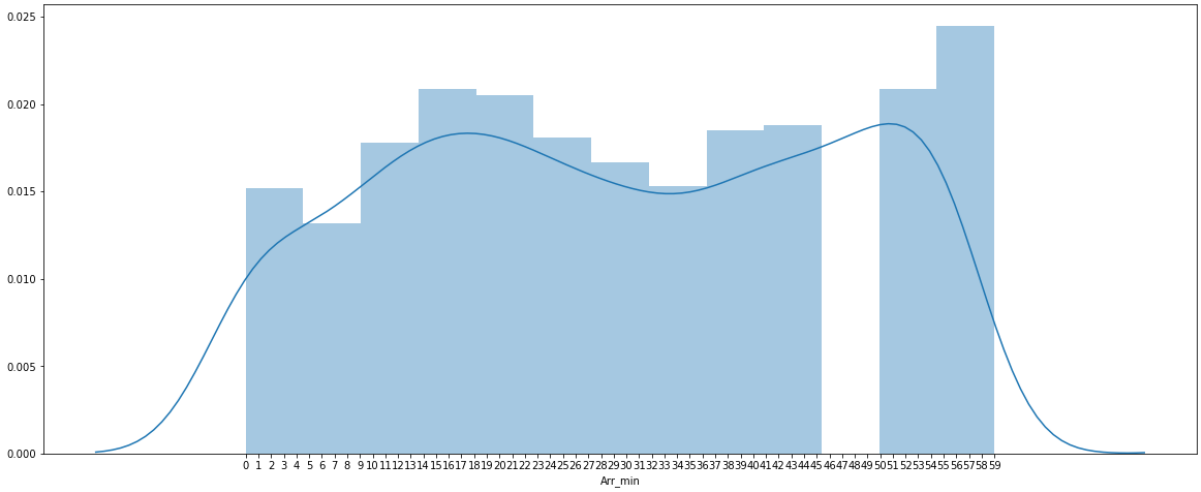


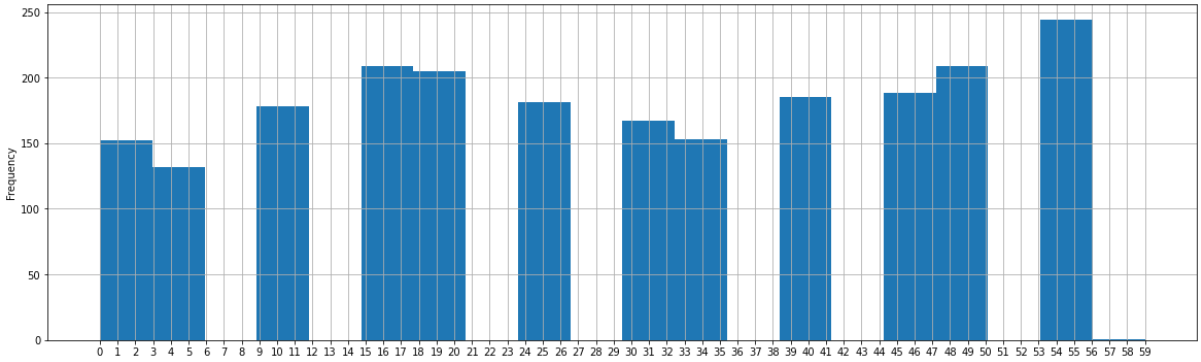
Observations: In the above distribution plot and histogram we can see that the data in the graph is spreaded. Also, the graph attains spike 3 times, i.e., between 0 to 3(maximum),8 to 11 and 44 to 47.So we can conclude that most of the flights departed either in the first quarter of the hour or in the the last.



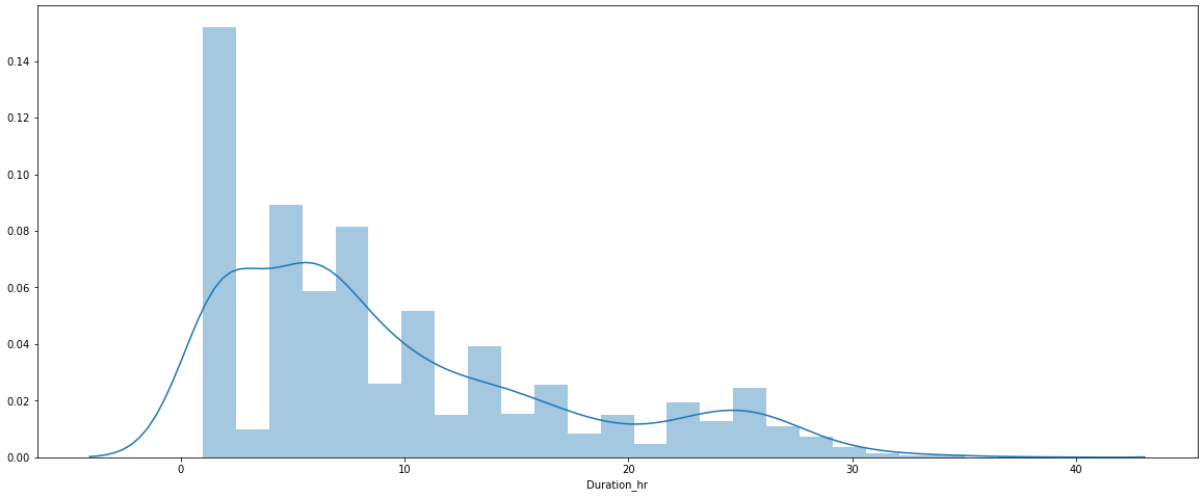


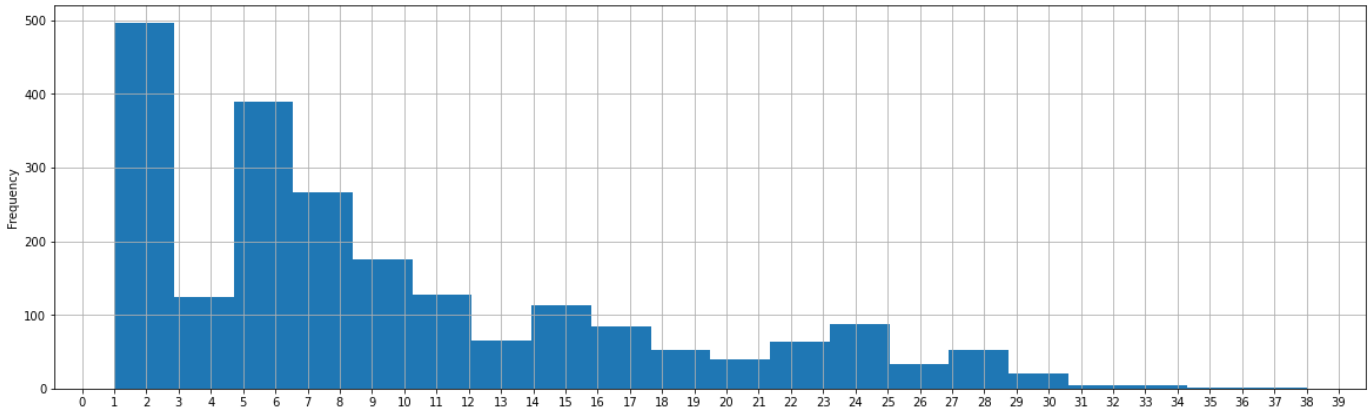
Observations: In the above distribution plot and histogram we can see that the data in the graph is slightly left skewed. Also, the graph attains multiple spikes. Most of the flights arrived at the airport between 7 to 8 hours, 15 to 16 hours but maximum number of the flight arrived between 22 to 23 hours (in the night)



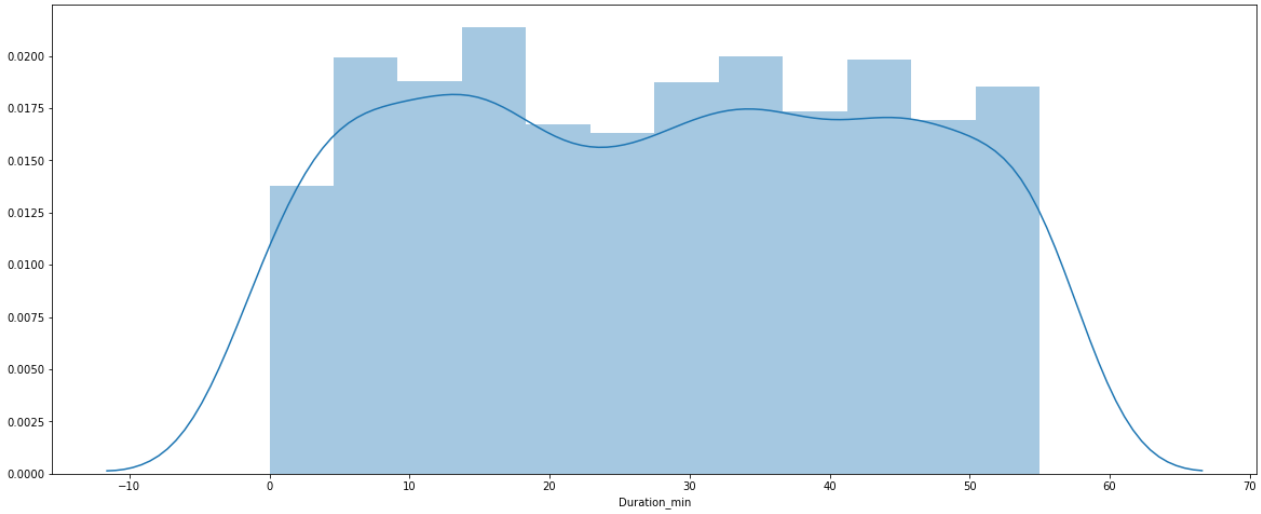


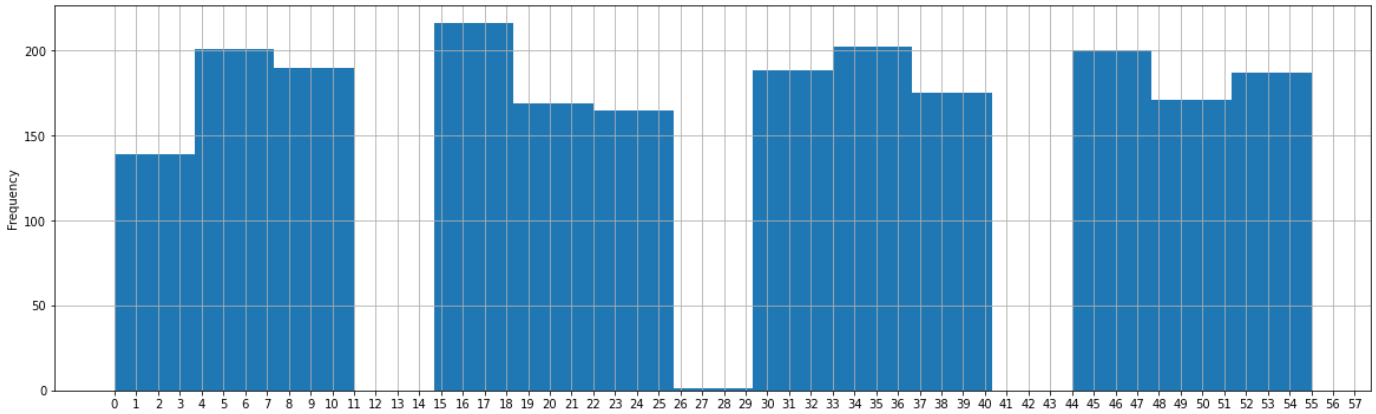
Observations: In the above distribution plot and histogram we can see that the data in the graph is spreaded .Also , the graph attains 3 spikes ,i.e., between 0 to 3 min, 15 to 20 min(highest), and 53 to 56 min. Maximum flights arrived at the end of the hour i.e. between 53 to 56 min of the hour.



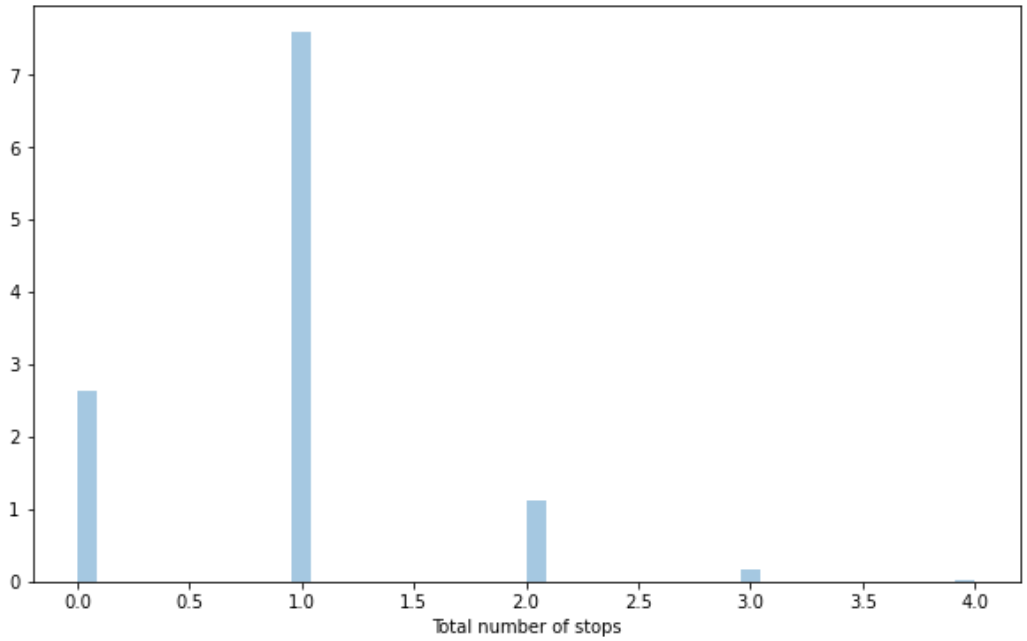


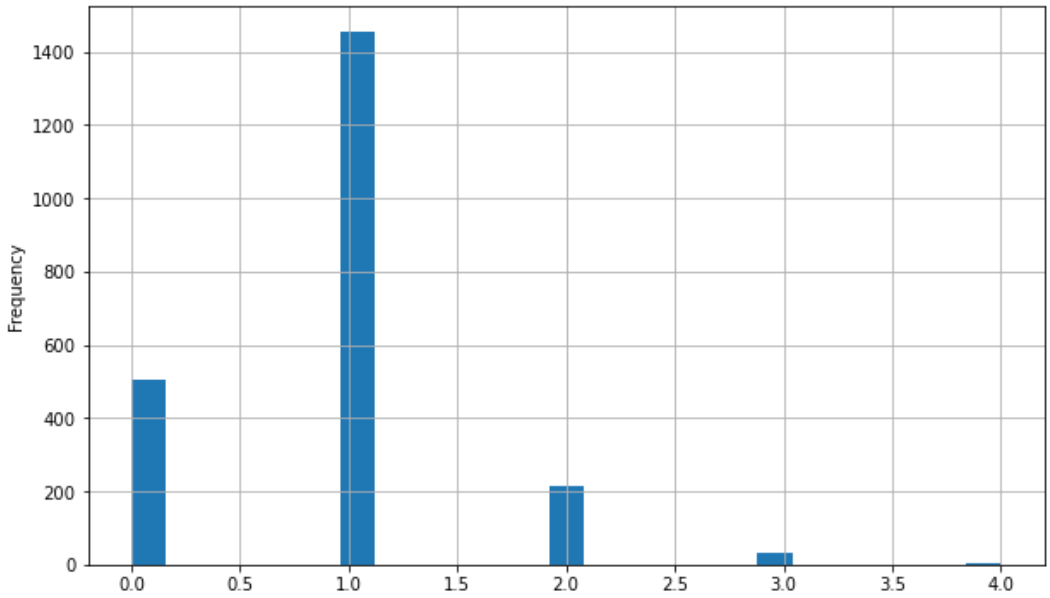
Observations: In the above distribution plot and histogram we can see that the data in the graph is right skewed. Also ,the graph attains 2 peaks , i.e., between 1 to 3 and 5 to 7.From the above graphs we can conclude that most of the flights took either 1 to 3 hours to reach their destination or 5 to 7.



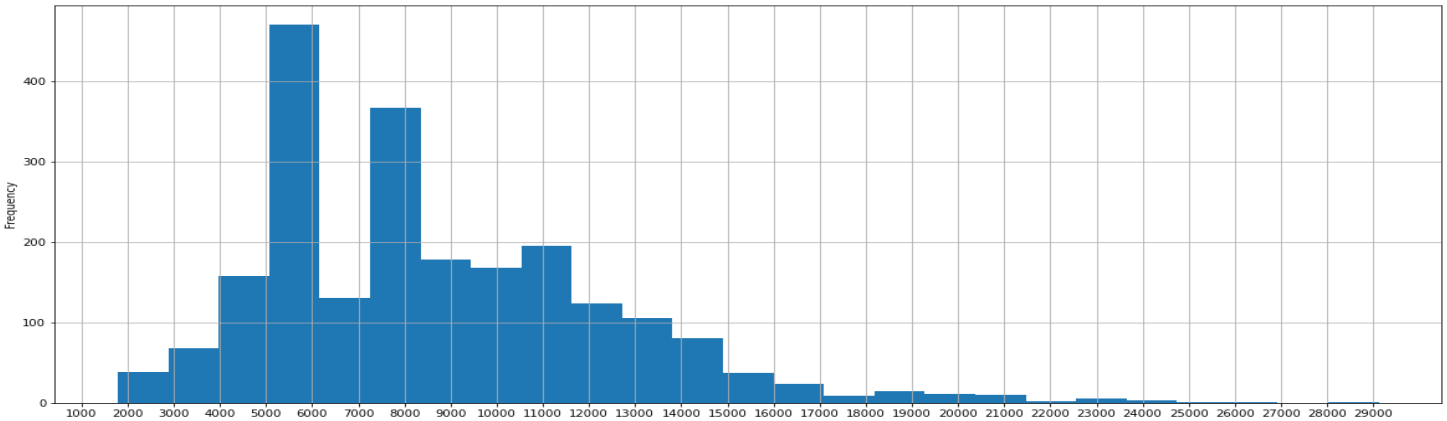


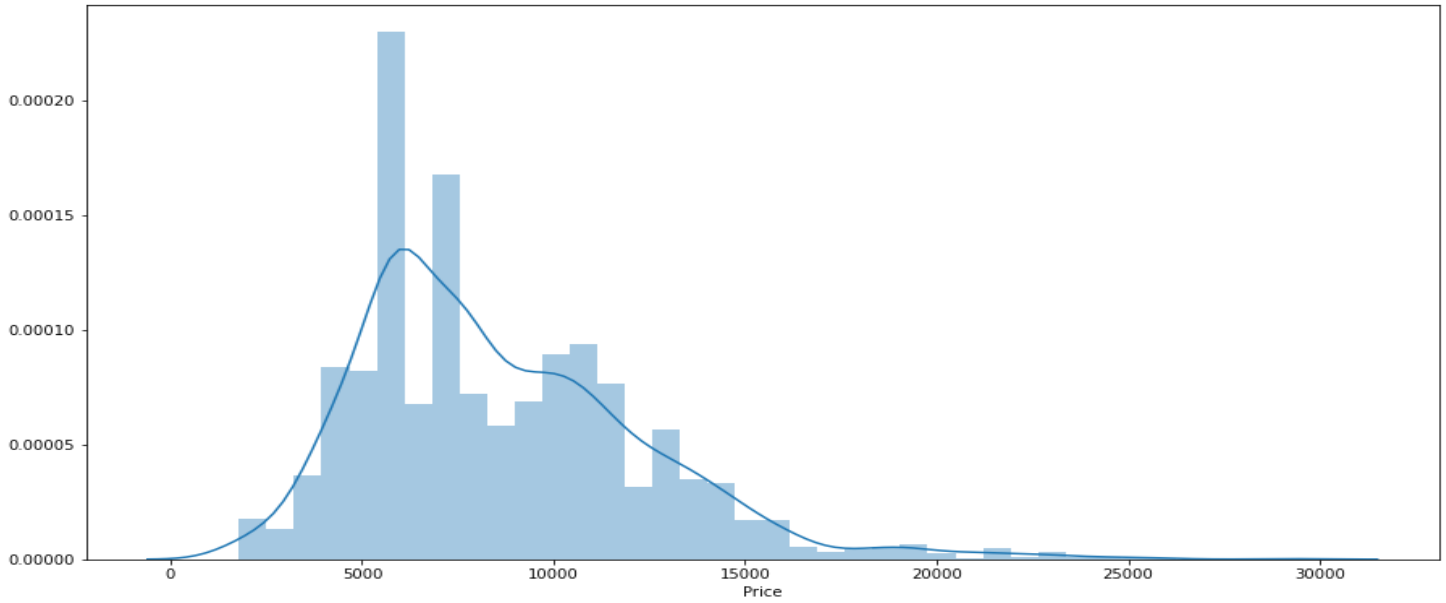
Observations: In the above distribution plot and histogram we can see that the data in the graph is spreaded .Also ,the graph has multiple peaks between 0 to 55 minutes.



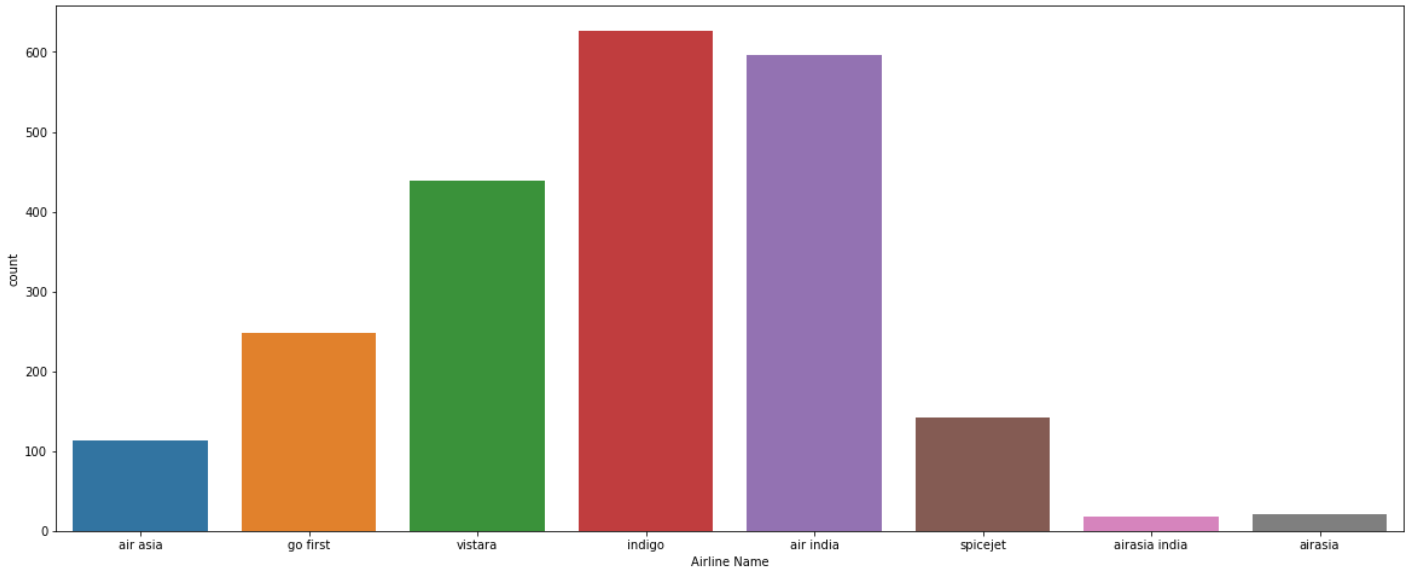


Observations: In the above distribution plot and histogram we can see that the graphs have some fixed values, i.e., between 0 to 4 which indicated the number of stops each flight has before reaching its destination. Most of the flights has 1 stop.





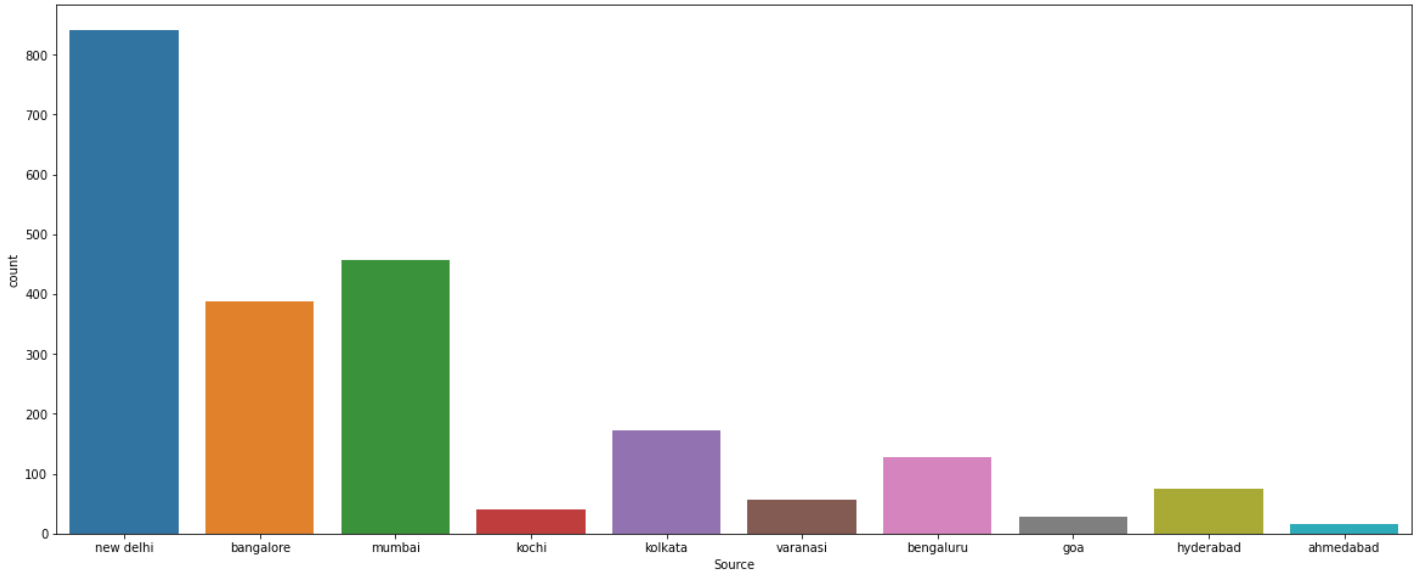
Observations: In the above distribution plot and histogram we can see that the graph is slightly right skewed. Also, the graph attains 3 peaks, i.e., between rupees 5000 to 6100, 7000 to 8500 and 10500 to 11500.So from here we can conclude that the price of most of the flights ranges either between 5000 to 6100 or 7000 to 8500.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 8 different flight names.

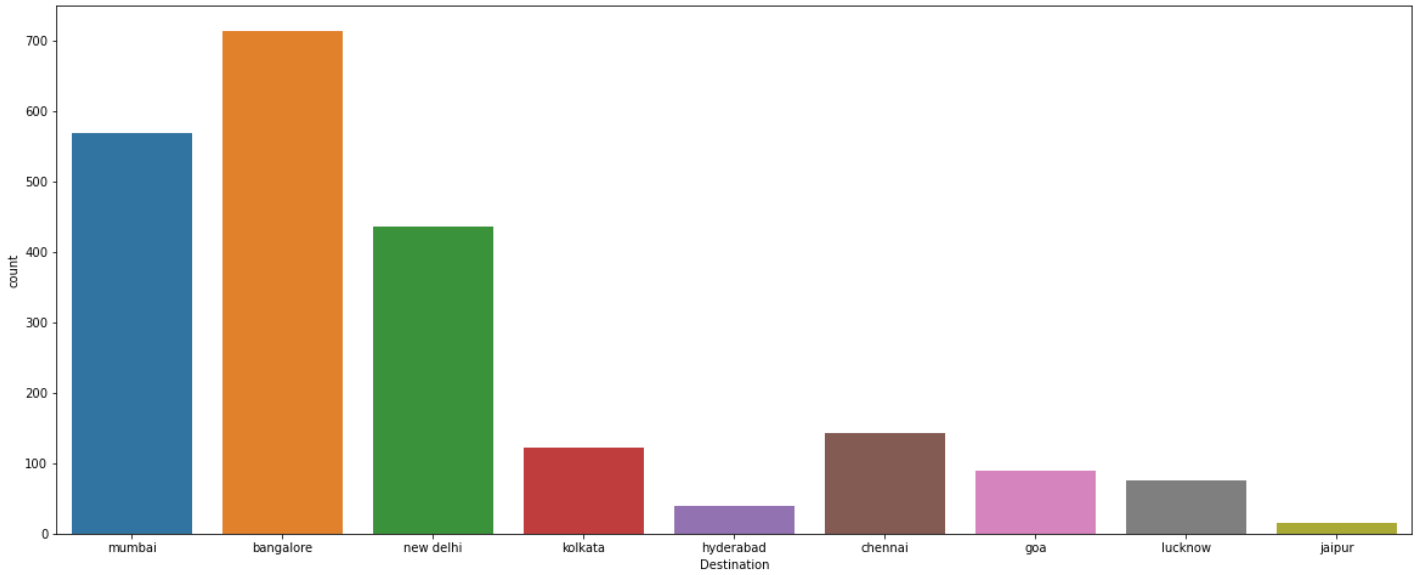
2. Indigo flight occurs most of the frequent in this dataset followed by air india and vistara airlines. Here, air asia and airasia are same.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 10 different sources of the flight.

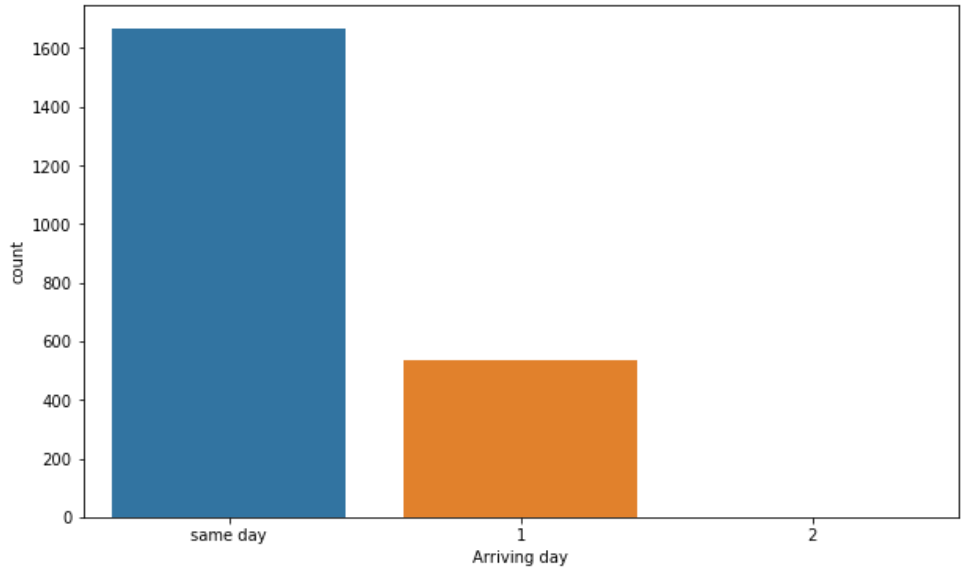
2. New Delhi occurs most of the frequent in this dataset followed by Bangalore and then Mumbai. Here, Bengaluru and Bangalore are the same.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 9 different Destination of the flight.

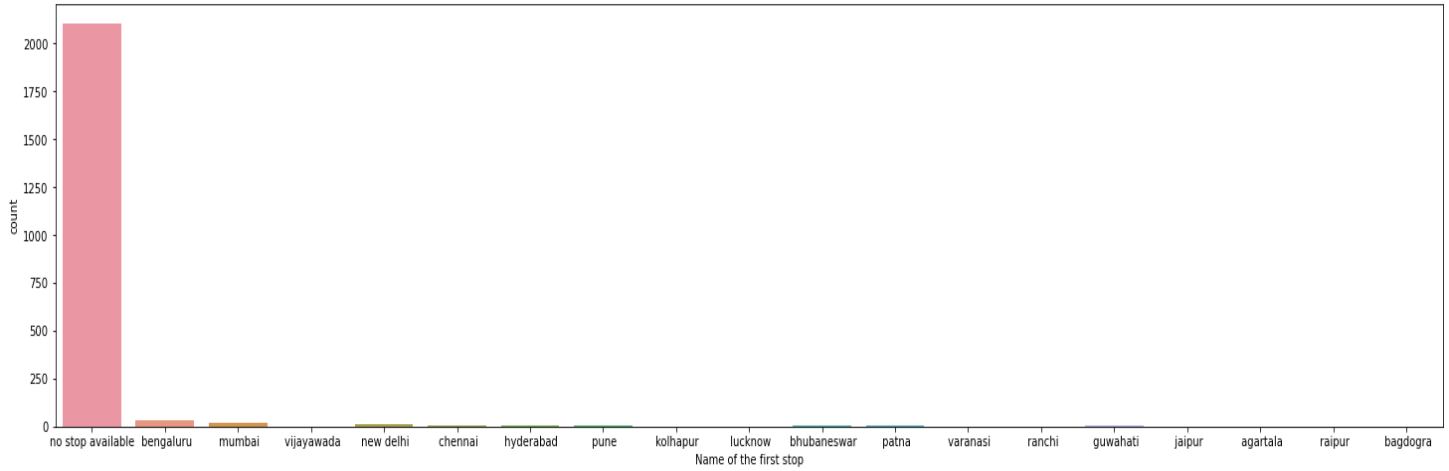
2. Bangalore occurs most of the frequent in this dataset followed by Mumbai and New Delhi.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 3 different arriving time.

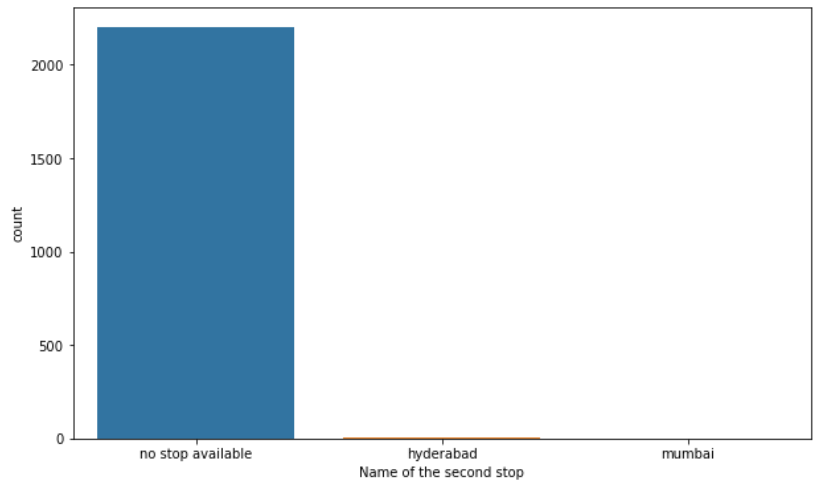
2. Most of the flights arrive on the same day of the flight. And only flight takes two days to reach its destination.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 19 different first stop of the flights.

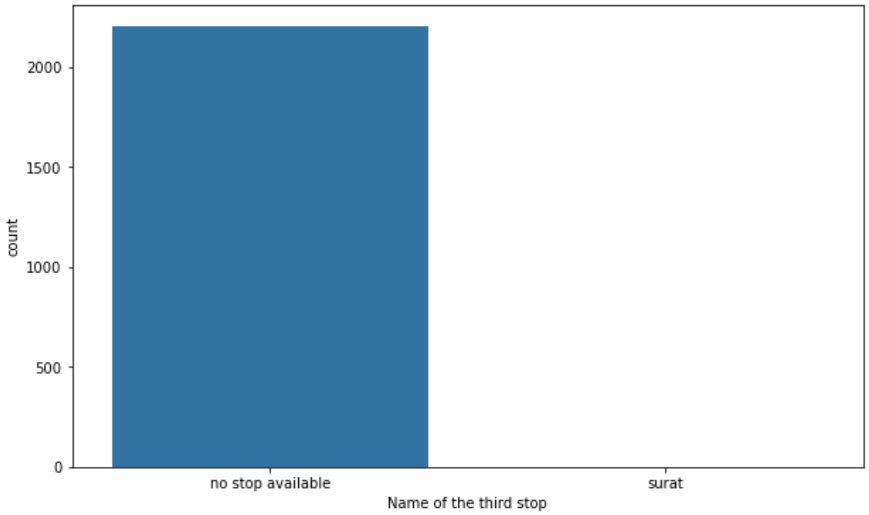
2. Most of the flights do not have any stop between the source and destination. Around 32 flights have first stops as Bengaluru.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 3 different second stop of the flights.

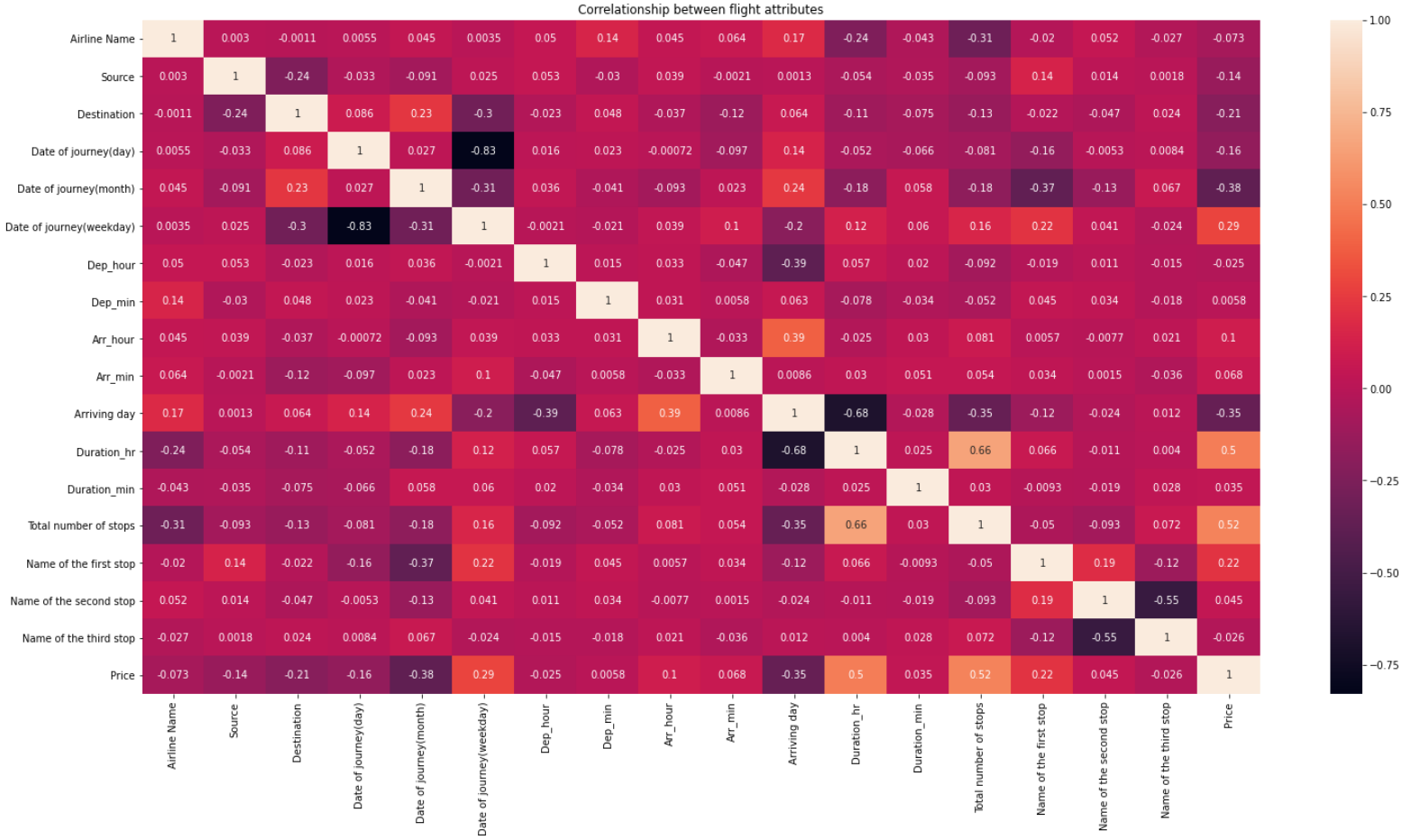
2. Most of the flights do not have second stop between the source and destination. Only 4 flights have second stop.



Observations: From the above count plot and information we made following conclusion:

1. This attribute contains 2 different third stop of the flights.

2. Most of the flights do not have any stop between the source and destination. Only 1 flight has a third stop.



- Interpretation of the Results
    - The interpretation made was that there were no null values present in the data set. Also, from the heatmap that shows the relationship between all the features we concluded the following result
    - Source and destinations are showing a negative correlation with each other means, if one increases other decreases and vice versa.
    - Airline name and Dep_hour, Airline name and Arriving day shows positive correlation with each other, if one increases other also increases.
    - Airline name and Duration_hr, Airline name and Total number of stops shows negative correlation with each other.
    - Destination and Date of journey(weekdays), Destination and Price shows negative correlation with each other.
    - Date of journey (day) and date of journey(weekdays), Duration_hr and Arriving day are highly negative correlated to each other.
    - Arriving day and Arr_hour, Total number of stops and duration_hr, Price and Duration_hr, Price and total number of stops are more positively correlated to each other as compared to other features.
    - If we compare the flight prices of the month of October, we can see that price of the flight changes frequently near the date of departure. The flight prices increase in large increments near the departure dates. But increases and decreases in very small increments afterwards. Similarly, if we compare the flight prices in the month of November and December, we can see that flight prices decrease in small increments over time. Therefore, we concluded that the flight prices tend to decrease over time, sometimes they show sudden increases but most of the time they tend to decrease.
    - The data was collected on 6th of the October. We observed that near the departure day, the flight price increases in large jumps.
    - From the above data we can conclude that the best time to purchase a flight ticket is in the afternoon between 12 p.m. to 6 p.m.
    - The morning flights are cheaper than the evening flights, But slightly expensive than afternoon flights. But the flight tickets are most expensive between 7 p.m. to 10 p.m.

The final model selected:

After checking the accuracy of all the models, we observed that Random Forest regressor model shows Maximum accuracy i.e., around 82 percent. The next step is to check the cross-validation score of these models in order to check whether there is any biasness in the model or not.

After comparing the cross-validation scores of the models we saw that Random Forest Regressor model shows least difference and Linear regressor model shows the maximum difference. Also, we saw earlier that Random Forest regressor model has the highest accuracy and has least Mean Absolute error compared to other models. The difference between the cross-validation score and accuracy of this model is lowest as compared to other models. So, we'll be using Random Forest regressor model as our final model.

Hyperparameter tuning of Random Forest Regressor

In [221]:
```
from sklearn.model_selection import GridSearchCV
#creating parameter list to pass in GridSearchCV
parameters = {'bootstrap': [True, False],
              'n_estimators': [1,100],
             'criterion': ['squared_error','mse','absolute_error','mae'],
             'min_samples_leaf': [1, 2, 4],
             'min_samples_split' : [1,10],
             'max_depth' : [1,10],
             'max_features' : ['auto', 'sqrt', 'log2']}
```

In [222]:
```
GVC = GridSearchCV(RandomForestRegressor(),parameters,cv = 5)
GVC.fit(x_train,y_train)
```

Out[222]:
```
GridSearchCV(cv=5, estimator=RandomForestRegressor(),
             param_grid={'bootstrap': [True, False],
                         'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'max_depth': [1, 10],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_leaf': [1, 2, 4],
                         'min_samples_split': [1, 10],
                         'n_estimators': [1, 100]})
```

In [66]:
```
GVC.best_params_
```

Out[66]:
```
{'bootstrap': True,
 'criterion': 'mse',
 'max_depth': 10,
 'max_features': 'auto',
 'min_samples_leaf': 2,
 'min_samples_split': 10,
 'n_estimators': 100}
```

In [121]:
```
Boosted_model = RandomForestRegressor(bootstrap= 'True',criterion = 'mse',max_depth = 10,max_features = 'auto',
                                      min_samples_split = 10 ,n_estimators = 100,min_samples_leaf = 2)
Boosted_model.fit(x_train,y_train)
pred = Boosted_model.predict(x_test)
print("accuracy score of the final model is :",r2_score(y_test,pred)*100)
```

```
accuracy score of the final model is : 82.1887442998764
```

Conclusion: The final model has an accuracy of the 82.18 %, after using the following best parameters:

'criterion': 'mse',

'max_depth': 10,

'max_features': 'auto',

'min_samples_split': 10,

'n_estimators': 100,

'bootstrap': 'True'

,'min_samples_leaf: 2'

Now we'll use this model to predict the price of the flight ticket.

CONCLUSION

Optimal timing for airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we majorly targeted to uncover underlying trends of flight prices in India using historical data and also to suggest the best time to buy a flight ticket. Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Therefore, various models are proposed to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price. By performing different models, it was aimed to get different perspectives and eventually compared their performance. With this study, it purpose was to predict prices of flights. For this project, we have collected data from 10 routes across India. The dataset contains 18 columns and 2204 rows after cleaning. With the help of the data visualizations and exploratory data analysis, the dataset was uncovered and features were explored deeply. The relation between features were examined. At the last stage, predictive models were applied to predict price of cars in an order: random forest, linear regression, KNeighbors regression, Decision Tree Regression, Gradient Boosting regression.

During this project I learnt a lot about new algorithms, data cleaning process, data scaling, normalization of data using various techniques. the data visualization process, especially count plot in the beginning helped me understanding the dataset. Further by using scatter plot I got to know relationship of different feature with target variables in less time and more effectively. The other visualisation techniques, like box plot, histogram, distribution plot, heat maps help in getting more information in less time and are effective and efficient also. While finalising the model for the project I used basic parameters to check the efficiency of each model. However, Random Forest algorithm was best among the others. After doing hyperparameter tuning of the model, I found that the random forest regressor works best in the parameters{'bootstrap': False, 'criterion': 'mse', 'max_depth': 10,'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 100}.

The main challenge faced during the project was Cleaning the data and extracting the useful data from the junk data. The Web scrapping and Data cleaning process took the most time. But I learned a lot from this project.