AMRITA
VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

DEPARTMENT OF COMPUTER SCIENCE

& ENGINEERING

19CSE304 – Foundations of Data Science

Project Report

Abstract:

Flying has become an important part of today's lifestyle as more people opt for faster travel options. The cost of a plane ticket varies depending on a number of factors, including travel schedules, destination, and duration. Having a basic understanding of airline costs before planning a vacation would surely save many people money and time at various times, such as during vacations or the holiday season. The purpose of this project is to predict travel pricing for various airlines using a machine learning model. The user will be given the expected values, which they can use as a reference to figure out how to buy their tickets. Using python libraries, the projected prediction model will be developed by applying machine learning techniques to the acquired historical data of flights.

Introduction:

Flights are frequently used to travel in today's globe since they save time and give greater comfort than other modes of transportation. In addition, if you book your flight at the proper time, you can save a lot of money. However, since the fare varies with time, passengers encounter a tremendous issue in forecasting the fare. Anyone who has ever purchased a plane ticket knows how quickly prices fluctuate. Aircraft uses complex revenue management tactics to implement a unique pricing plan. The cheapest accessible ticket fluctuates over time, and the price of a ticket can be high or low.

This valuation approach adjusts the toll according to the time of day, such as morning, afternoon, or night. Seasonal variations in price, such as winter, summer, and holiday seasons, are also possible. The carrier's primary goal is to increase revenue, while the buyer is looking for the best deal. Purchasers typically try to acquire tickets well in advance of the departure date. They believe that airfare will be more expensive as the date of purchase approaches the departure date, although this is not always the case. A purchaser may end up paying more for a similar seat than they should.

According to a research, India's friendly aeronautics industry is on the rise. In 2020, India will be the third-largest avionics market, and by 2030, it would be the largest. By 2017, Indian aviation traffic is expected to reach 100 million passengers, up from 81 million in 2015. According to Google, "Cheap Air Tickets" is the most searched term in India. When the Indian white-collar class is introduced to air travel, customers are looking for low-cost options. The number of low-cost aeroplane tickets is increasing all the time.

In this project, we will be predicting the flight fare using machine learning concepts like Linear Regression and Decision tree Regressor.

Machine learning is the study of computer algorithms, which improve with experience and use of data. Machine learning algorithms build a model based on sample data and make predictions or decisions using this model without being programmed to do so. Machine Learning concepts serve a wide range of applications. One such application in the Aviation Industry. is the prediction of flight fares and delays which will be of great help to the passengers. As the flight fare depends upon various features like source, destination, time of booking etc., we can predict the fare of the tickets. Similarly, a flight delay can be predicted using parameters like departure delay, weather conditions etc.

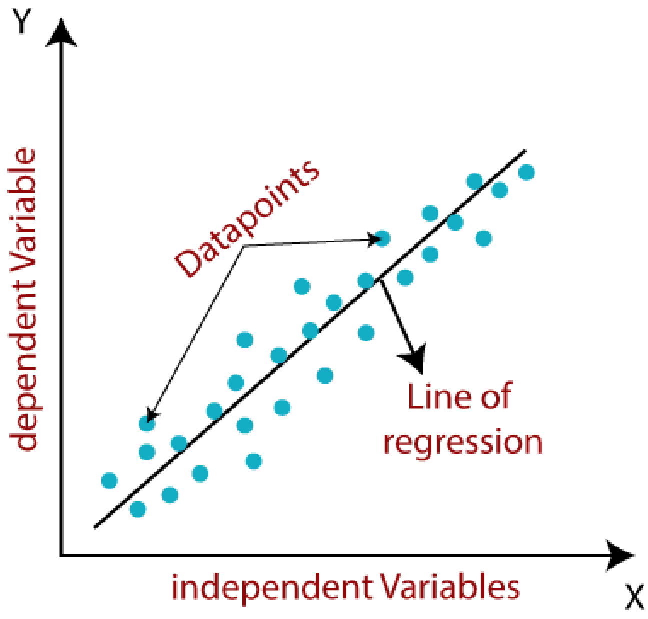
The coding part of these models are done using python libraries such as pandas, NumPy, matplotlib, seaborn and sklearn.

Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as **sales, salary, age, product price,** etc.

Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

The linear regression model provides a sloped straight line representing the relationship between the variables.

Consider the below image:



Mathematically, we can represent a linear regression as:

$$y = a_0 + a_1 x + \varepsilon$$

Here,

Y= Dependent Variable (Target Variable)
X= Independent Variable (predictor Variable)
a0= intercept of the line (Gives an additional degree of freedom)
a1 = Linear regression coefficient (scale factor to each input value).
ε = random error

The values for x and y variables are training datasets for Linear Regression model representation.

A decision tree is a hierarchical model for supervised learning in a sequence of recursive splits in a smaller number of steps. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves. Each decision node m implements a test function f(x) with discrete outcomes labelling the branches. Process starts at the root and is repeated recursively until a leaf node is hit, at which point the value written in the leaf constitutes the output.

Decision tree is a classifier in the form of a tree structure which consists of:

• Root Nodes – It is the node present at the beginning of a decision tree from this node the population starts dividing according to various features.

• Decision Nodes – the nodes we get after splitting the root nodes are called Decision Node.

• Leaf Nodes – the nodes where further splitting is not possible are called leaf nodes or terminal nodes.

• Sub-tree – just like a small portion of a graph is called sub-graph similarly a sub-section of this decision tree is called sub-tree.

Problem Statement:

Flight ticket prices can be something hard to guess, today we might see a price, check out the price of the same flight tomorrow, it will be a different story. We might have often heard travelers saying that flight ticket prices are so unpredictable.

Here I take on the challenge! Here the Dataset has flight tickets for various airlines between the months of March and June of 2019 and between various cities and we have to predict the prices of Flights with the help of some features.

Literature Review:

1. Jaya Shukla et al., International Journal of Research in Engineering, IT and Social Sciences, ISSN 2250-0588, Impact Factor: 6.565, Volume 10 Issue 05, May 2020, Page 15-18

   Airline Price Prediction using Machine Learning by Jaya Shukla , Aditi Srivastava , and Anjali Chauhan.

   In this paper, they used the machine learning algorithm used to predict airline prices. Machine Learning models in the computational intelligence field that are evaluated before on different datasets are studied. Their mean and standard deviation are evaluated and compared in order to get better results. The algorithms that they have chosen to study are Linear Regression, Support Vector Regression, Gradient Boosting Algorithm, and Random Forest. Based on the analysis of their results:

   • The SVM algorithm showed a standard deviation of 94.73.

- The Linear Regression algorithm showed the standard deviation of 84.71.
- The KNN algorithm showed a standard deviation of 107.80.
- The Random Forest algorithm showed a standard deviation of 80.15.
- The Gradient Boosting algorithm showed a standard deviation of 62.75.

2. IARJSET
   International Advanced Research Journal in Science, Engineering and Technology Vol. 8, Issue 3, March 2021
   Flight Price Prediction for Users by Machine Learning Techniques Pavithra Maria K , Anitha K L

   In this paper, Evaluating the algorithmic rule, a dataset is collected, pre-processed, performed data modelling and studied a value difference for the number of restricted days by the passengers for travelling. Machine Learning algorithms with square measure for forecasting the accurate fare of airlines and it gives accurate value of plane price ticket at limited and highest value. Information is collected from Kaggle websites that sell the flight tickets therefore restricting data which are often accessed. The results obtained by the random forest and decision tree algorithm has better accuracy, but best accuracy is predicted by decision tree algorithm as shown is the above analysis. Accuracy of the model is also forecasted by the R-squared value.
3. International Journal of Engineering Research & Technology (IJERT)
   ISSN: 2278-0181
   Vol. 8 Issue 06, June-2019

   A Survey on Flight Pricing Prediction using Machine Learning by Supriya Rajankar and Neha Sakharkar

   In the proposed paper the overall survey for the dynamic price changes in the flight tickets is presented. this gives the information about the highs and lows in the airfares acording to the days, weekend and time of the day that is morning, evening and night. also the machine learning models in the computational intelligence feild that are evaluated before on different datasets are studied. their accuracy and performances are evaluated and compared in order to get better result. For the prediction of the ticket prices perfectly differnt prediction models are tested for the better prediction accuracy. As the pricing models of the company are developed in order to maximize the revenue management. So to get result with maximum accuracy regression analysis is used.

4. INTERNATIONAL JOURNAL OF SCIENTIFIC & TECHNOLOGY RESEARCH VOLUME 8, ISSUE 12, DECEMBER 2019 ISSN 2277-8616

   Predicting The Price Of A Flight Ticket With The Use Of Machine Learning Algorithms by Supriya Rajankar, Neha Sakharkar, Omprakash Rajankar

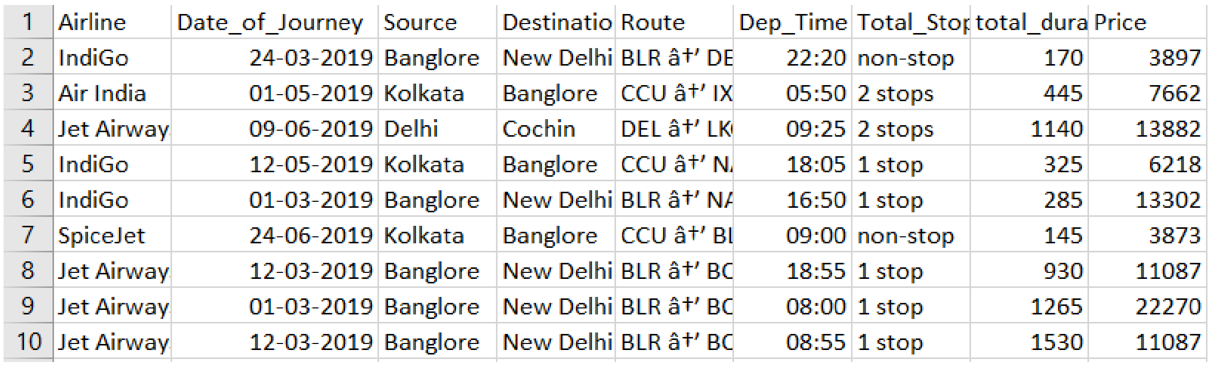
To evaluate the conventional algorithm, a dataset is built for route BOMBAY to DELHI and studied a trend of price variation for the period of limited days. Machine Learning algorithms are applied on the dataset to predict the dynamic fare of flights. This gives the predicted values of flight fare to 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.

Dataset Description:

The dataset used for our project is “Flight Price Prediction” from Kaggle. It contains 13353 rows.

It contains the following attributes:

1. Airline
2. Date of Journey
3. Source
4. Destination
5. Route
6. Departure time
7. Total_stops
8. Total_ Duration
9. Price



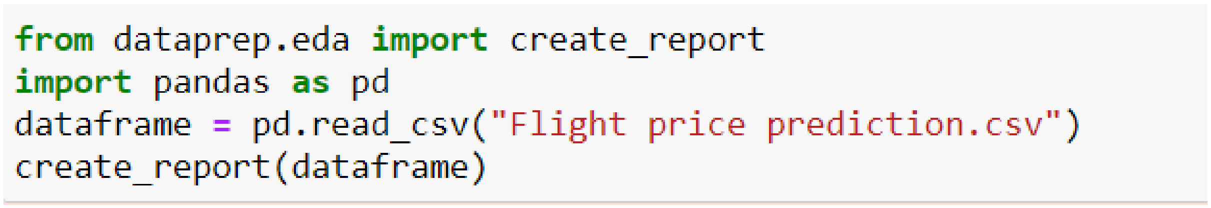
1	Airline	Date_of_Journey	Source	Destinatio	Route	Dep_Time	Total_Stop	total_dura	Price
2	IndiGo	24-03-2019	Banglore	New Delhi	BLR â†' DE	22:20	non-stop	170	3897
3	Air India	01-05-2019	Kolkata	Banglore	CCU â†' IX	05:50	2 stops	445	7662
4	Jet Airway	09-06-2019	Delhi	Cochin	DEL â†' LK	09:25	2 stops	1140	13882
5	IndiGo	12-05-2019	Kolkata	Banglore	CCU â†' N.	18:05	1 stop	325	6218
6	IndiGo	01-03-2019	Banglore	New Delhi	BLR â†' NA	16:50	1 stop	285	13302
7	SpiceJet	24-06-2019	Kolkata	Banglore	CCU â†' BL	09:00	non-stop	145	3873
8	Jet Airway	12-03-2019	Banglore	New Delhi	BLR â†' BC	18:55	1 stop	930	11087
9	Jet Airway	01-03-2019	Banglore	New Delhi	BLR â†' BC	08:00	1 stop	1265	22270
10	Jet Airway	12-03-2019	Banglore	New Delhi	BLR â†' BC	08:55	1 stop	1530	11087

Glimpse of our dataset

Data Analysis:

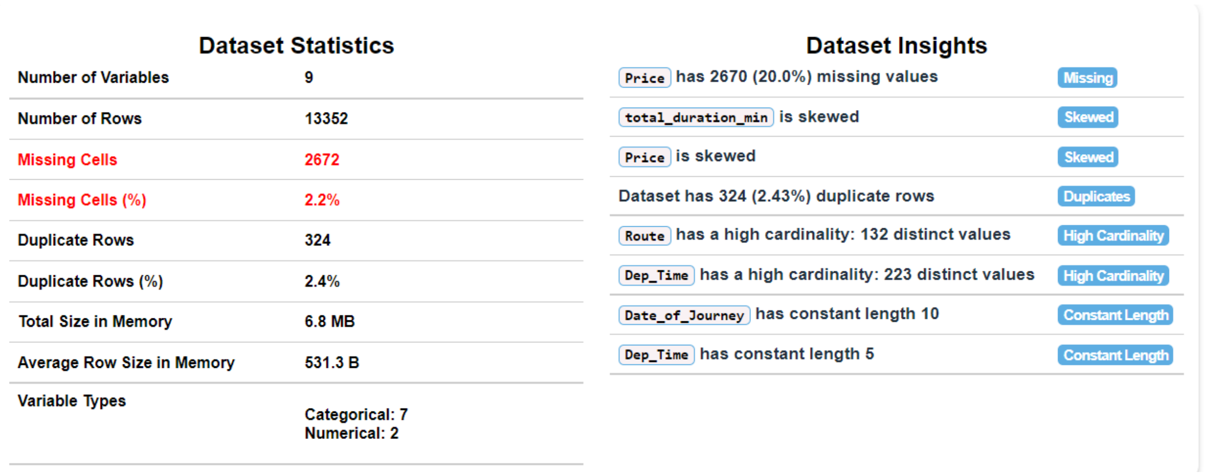
The procedure of extracting information from given raw data is called data analysis.

Here we will use eda module of data-prep library to do this step.



```
from dataprep.eda import create_report
import pandas as pd
dataframe = pd.read_csv("Flight price prediction.csv")
create_report(dataframe)
```

Overview



Dataset Statistics

Number of Variables	9
Number of Rows	13352
Missing Cells	2672
Missing Cells (%)	2.2%
Duplicate Rows	324
Duplicate Rows (%)	2.4%
Total Size in Memory	6.8 MB
Average Row Size in Memory	531.3 B
Variable Types	Categorical: 7 Numerical: 2

Dataset Insights

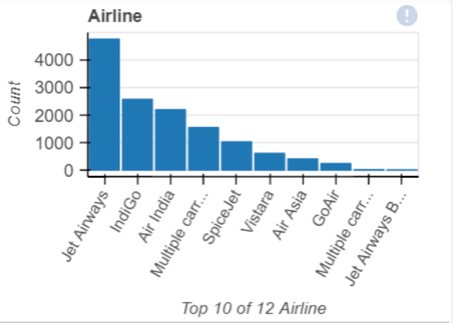
Price has 2670 (20.0%) missing values	Missing
total_duration_min is skewed	Skewed
Price is skewed	Skewed
Dataset has 324 (2.43%) duplicate rows	Duplicates
Route has a high cardinality: 132 distinct values	High Cardinality
Dep_Time has a high cardinality: 223 distinct values	High Cardinality
Date_of_Journey has constant length 10	Constant Length
Dep_Time has constant length 5	Constant Length

Variables

After you select the variable section you will get information as shown in the below figures.

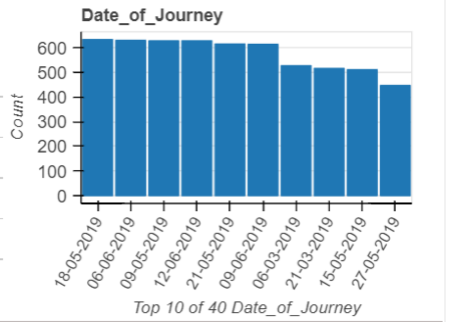
Airline
categorical
Show Details

Approximate Distinct Count	**12**
Approximate Unique (%)	**0.1%**
Missing	**0**
Missing (%)	**0.0%**
Memory Size	**975.7 KB**



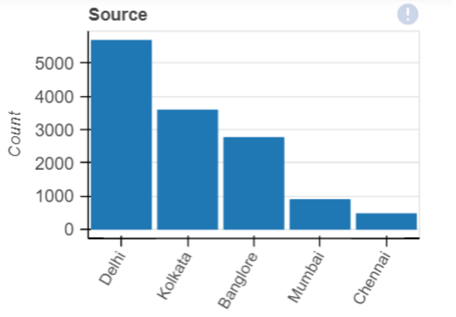
Date_of_Journey
categorical
Show Details

Approximate Distinct Count	**40**
Approximate Unique (%)	**0.3%**
Missing	**0**
Missing (%)	**0.0%**
Memory Size	**977.9 KB**



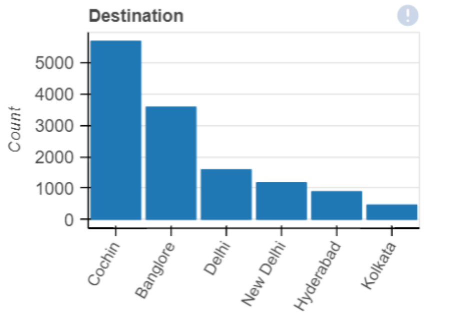
Source
categorical
Show Details

Approximate Distinct Count	**5**
Approximate Unique (%)	**0.0%**
Missing	**0**
Missing (%)	**0.0%**
Memory Size	**929.5 KB**



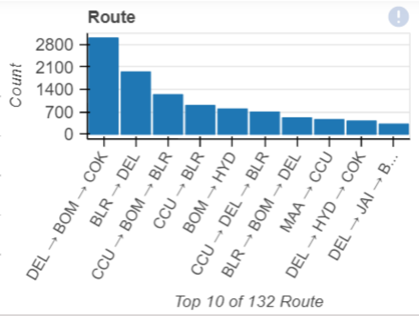
Destination
categorical
Show Details

Approximate Distinct Count	**6**
Approximate Unique (%)	**0.0%**
Missing	**0**
Missing (%)	**0.0%**
Memory Size	**937.7 KB**



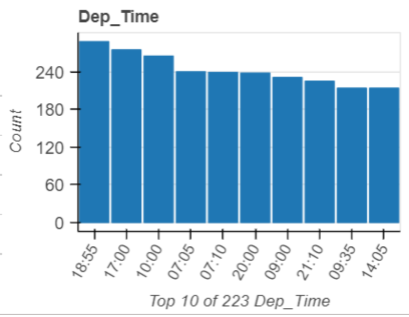
Route
categorical
Show Details

Approximate Distinct Count	132
Approximate Unique (%)	1.0%
Missing	1
Missing (%)	0.0%
Memory Size	1.7 MB



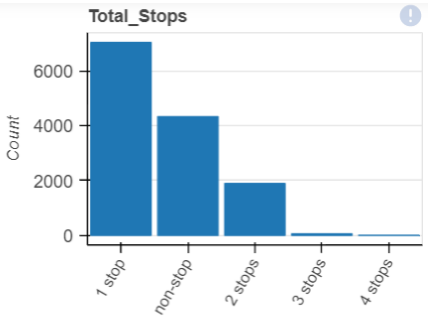
Dep_Time
categorical
Show Details

Approximate Distinct Count	223
Approximate Unique (%)	1.7%
Missing	0
Missing (%)	0.0%
Memory Size	912.7 KB



Total_Stops
categorical
Show Details

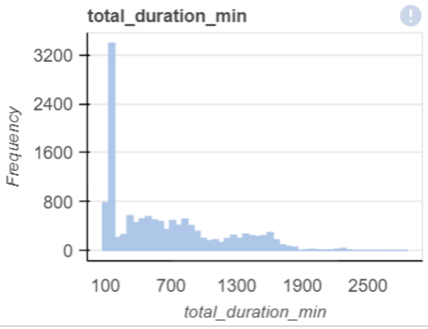
Approximate Distinct Count	5
Approximate Unique (%)	0.0%
Missing	1
Missing (%)	0.0%
Memory Size	936.1 KB



total_duration_min
numerical
Show Details

Approximate Distinct Count	373
Approximate Unique (%)	2.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	208.6 KB

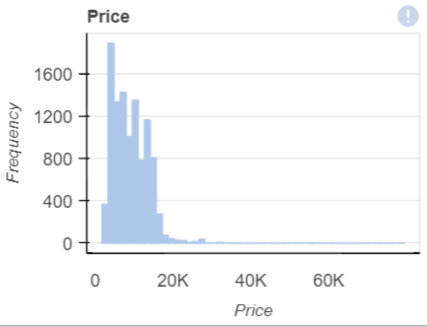
Mean	642.5183
Minimum	75
Maximum	2860
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%



Price
numerical
Show Details

Approximate Distinct Count	1870
Approximate Unique (%)	17.5%
Missing	2670
Missing (%)	20.0%
Infinite	0
Infinite (%)	0.0%
Memory Size	166.9 KB

Mean	9086.2927
Minimum	1759
Maximum	79512
Zeros	0
Zeros (%)	0.0%
Negatives	0
Negatives (%)	0.0%

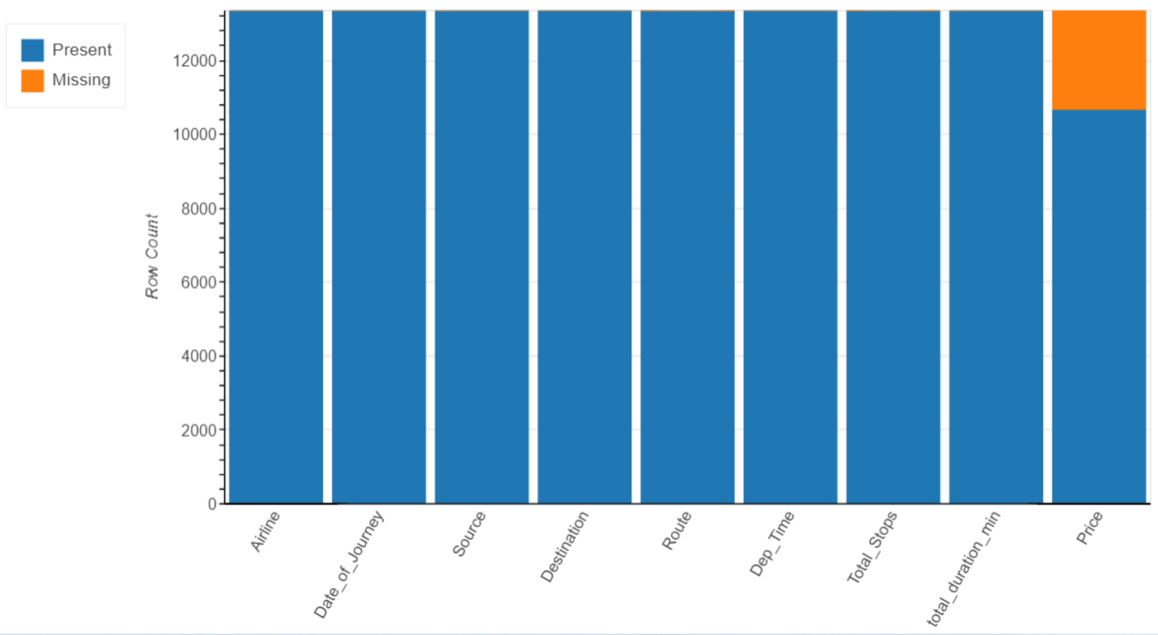


Missing Values

This section has multiple ways using which we can analyze missing values in variables. We will discuss three mostly used methods, bar-chart, spectrum, and Heat Map. Let’s explore each one by one.

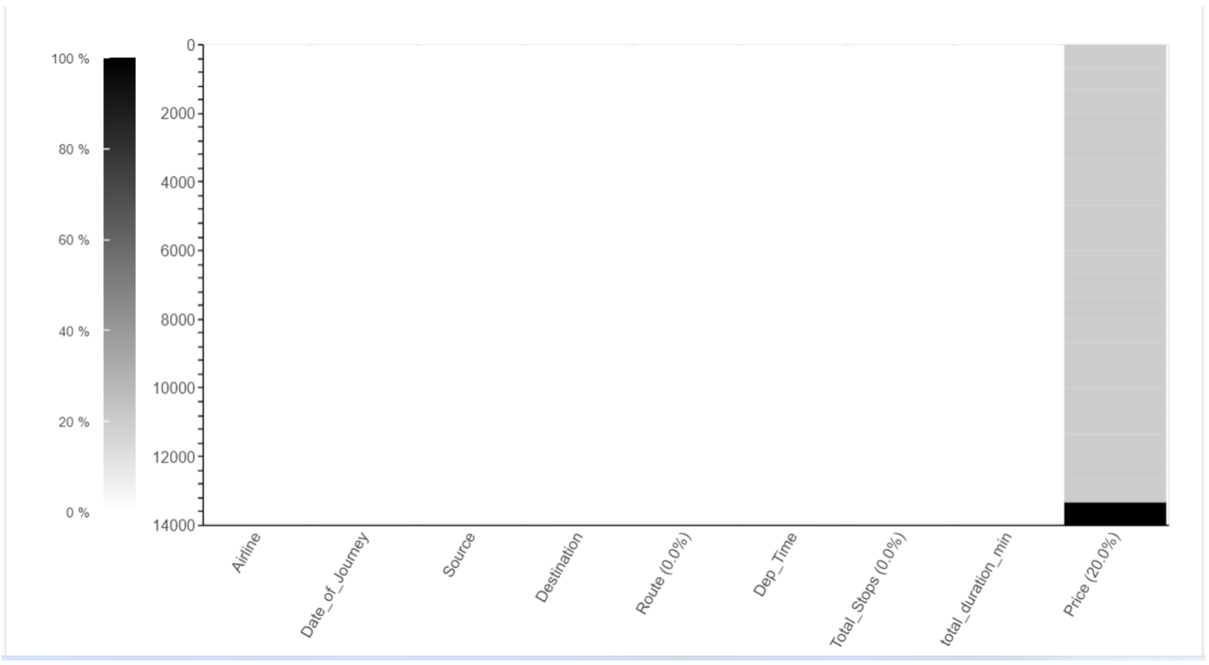
Bar Chart:

The bar chart method shows the ‘number of missing and present values’ in each variable in a different color.

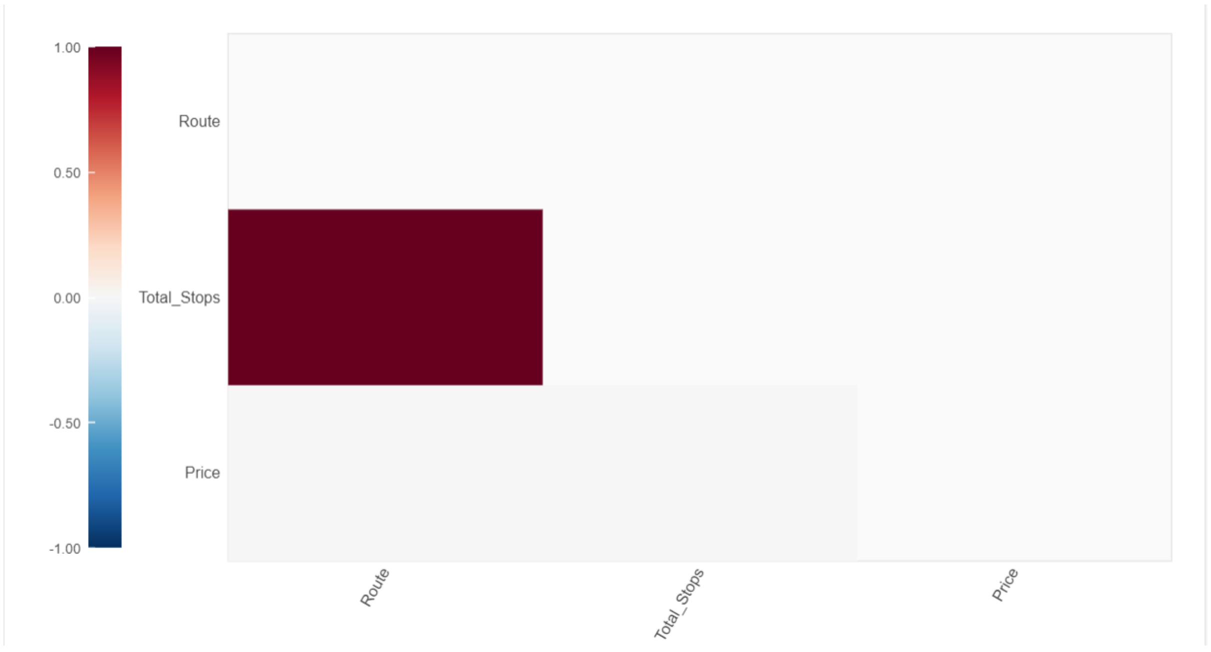


Spectrum:

The spectrum method shows the percentage of missing values in each variable.


100 %
80 %
60 %
40 %
20 %
0 %
0
2000
4000
6000
8000
10000
12000
14000
Airline
Date_of_Journey
Source
Destination
Route (0.0%)
Dep_Time
Total_Stops (0.0%)
total_duration_min
Price (20.0%)

Heatmap:


1.00
0.50
0.00
-0.50
-1.00
Route
Total_Stops
Price
Route
Total_Stops
Price

The Heat Map method shows variables having missing values in terms of correlation. Since ‘Route’, ‘Total_Stops’ and ‘Price’ all are highly correlated, they have missing values.

As we can observe ‘Route’ and ‘Total_Stops’ variables have missing values. Since we did not find any missing values information from Bar-Chart and Spectrum method but we found missing value variables using the Heat Map method. Combining both of these information, we can say that the ‘Route’ and ‘Total_Stops’ variables have missing values but are very low.

Purpose of this project:

This project aims to develop an model which will predict the flight prices for various flights using machine learning algorithms. The user will get the predicted values and with its reference the user can decide to book their tickets accordingly.

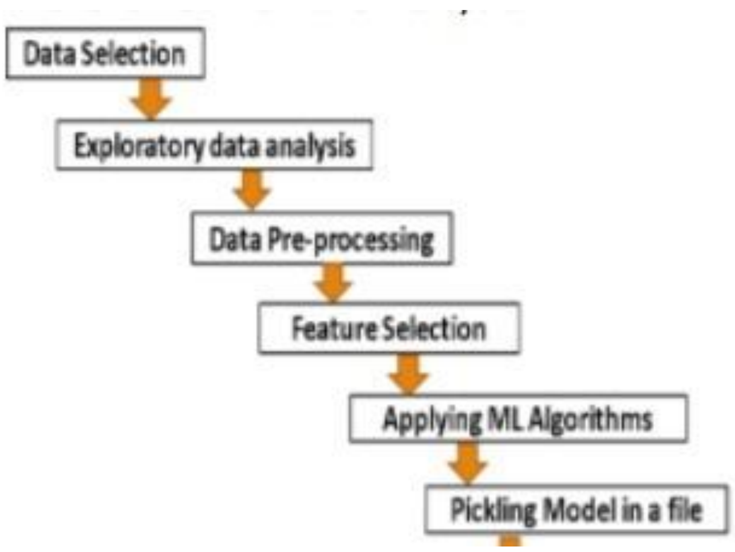
Existing Model:

The model combines moving average, rule learning and Q-learning together. Their data contains two routes: Los Angeles to Boston and Seattle to Washington, D.C. And there are five features in their model, which are flight number, hours until the departure date, airline, price and route. They generate several rules, for example, when the hours before takeoff is greater or equals to 252 and the current price is greater or equals to 2223 and route is from LA to Boston, we should wait. After that, there should be several “buy” and “wait” suggestions. The final result is achieved by using an ensemble method doing the voting. By utilizing the sequence of buy or wait signal, the cost of each stimulating passenger was calculated.

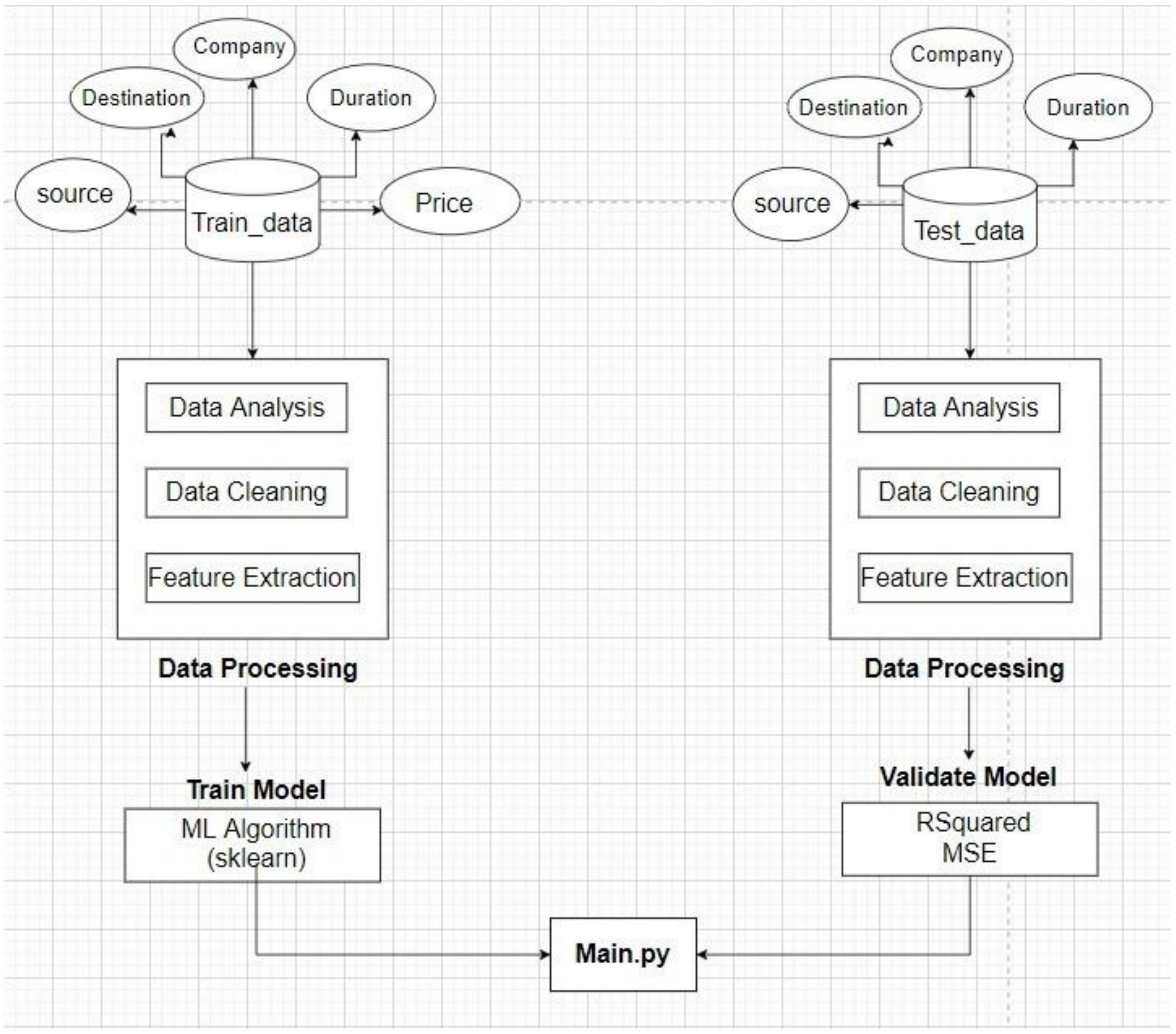
Proposed Model:

In our Model we had incomplete price value which are to be predicted , we transformed the categorical variables into dummy variables (Nan). All the collected data needed a lot of work so after the collection of data, it is needed to be clean and prepare according to the model requirements. All the unnecessary data is removed like duplicates and null values. We also split the data into train and tests sets which has price dummy variables to predict. Already which has price is split as train set. We tried regression models and evaluated the using Root Mean Squared Error.

System Architecture:



Flow Chart:



Code Implementation:

UNDERSTANDING THE PROBLEM:

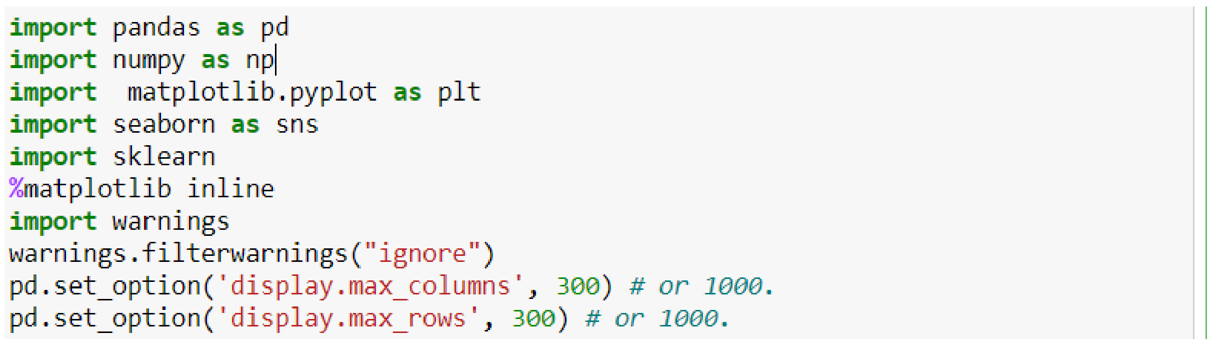
A dataset is formed by taking into consideration of the information of 12 different airlines. The problem is based on the given information about each airline we must calculate or predict the price of the particular airline from the source to the destination based on the travelling time period.

We predict the price of the particular flight by the use of any prediction modelling that gives the best accuracy and based on that model we are going to predict the price of the airlines.

DATASET:

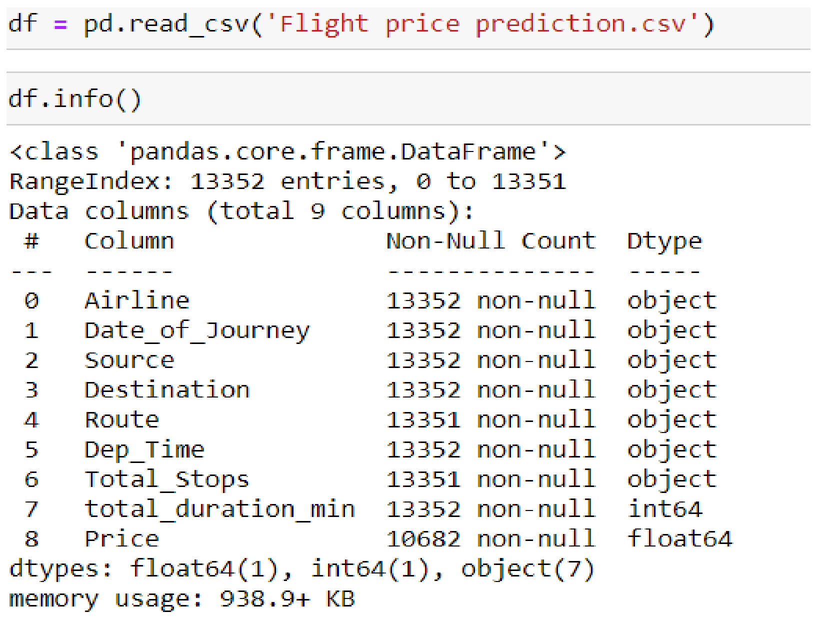
Here we are using the Flight Price Prediction dataset. The dataset contains 9 attributes are Airline, Date_of_Journey, Source, Destination, Route, Dep_time, Total_Stops, Total_duration_min, Price.

In our dataset actually the price is given up to 10,682 rows include of every airline and now we are going to predict the flight price of the remaining 2670 rows.



```python
import pandas as pd
import numpy as np
import  matplotlib.pyplot as plt
import seaborn as sns
import sklearn
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_columns', 300) # or 1000.
pd.set_option('display.max_rows', 300) # or 1000.
```

- ➔ Here We imported all the libraries that we need to predict this problem
- ➔ Now load the CSV file of our dataset using pandas dataframe. store the file in a variable called df. Then print the information of our dataset.

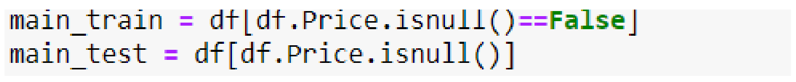


```
df = pd.read_csv('Flight price prediction.csv')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13352 entries, 0 to 13351
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  ------              --------------  -----  
 0   Airline             13352 non-null  object 
 1   Date_of_Journey     13352 non-null  object 
 2   Source              13352 non-null  object 
 3   Destination         13352 non-null  object 
 4   Route               13351 non-null  object 
 5   Dep_Time            13352 non-null  object 
 6   Total_Stops         13351 non-null  object 
 7   total_duration_min  13352 non-null  int64  
 8   Price               10682 non-null  float64
dtypes: float64(1), int64(1), object(7)
memory usage: 938.9+ KB
```

➔ Here we fixed the target variable as price because our aim is to find the price of the airline showing the NaN value, So we need to remove the datapoints where the target value is NULL.



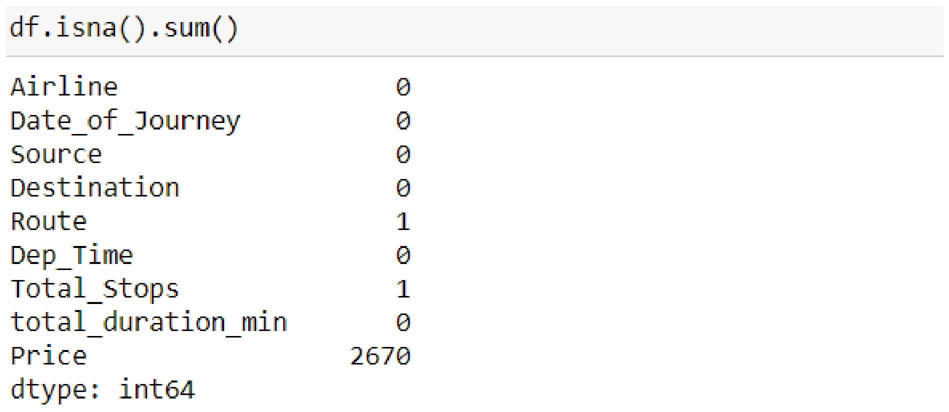
```
main_train = df[df.Price.isnull()==False]
main_test = df[df.Price.isnull()]
```

➔ Here main_test holds the rows which are having price value
main_test hold the rows which shows the price value NaN that we are going to predict later.

DATA PREPROCESSING:

Data Pre-Processing is a process of preparing the raw data and making it suitable for a prediction model. It is the first step that we cannot feed the raw data into the algorithm. So, we are pre -processing this dataset to make it fit and compatible for our predictive algorithm to learn.

Now we are going to find the null values in a dataset if found then we are going to remove that particular to get maximum accuracy in modelling.



```
df.isna().sum()
```

```
Airline                 0
Date_of_Journey         0
Source                  0
Destination             0
Route                   1
Dep_Time                0
Total_Stops             1
total_duration_min      0
Price                2670
dtype: int64
```

➔ After found that there is a null value in a attributes Route and Total_Stops we are going to remove that particular row. We are using NULL imputation to identifying the missing values in our dataset.

NULL IMPUTATION:

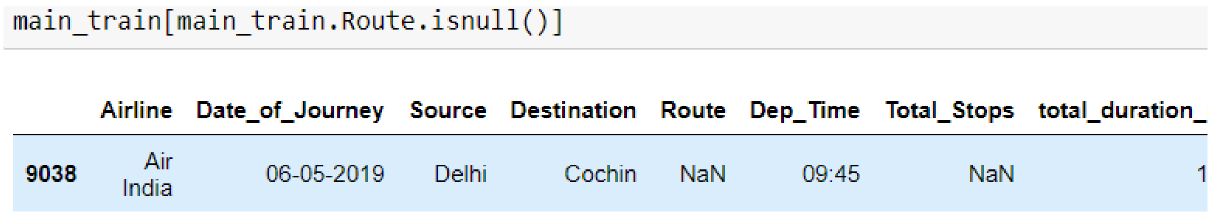
Datasets may have NULL value or missing values and this can cause problems for the predictive algorithms, so we need to identify the NULL values and replace or drop them for each column in our input prior to modelling our prediction task. This process is called data imputation or imputing for short.

After we are finding the missing data, we need to know the type of missing data. It is very important to know what sort of missing data we are dealing with when addresses missing values in our dataset.

There are three types of missing data:

1. **MCAR (Missing Completely at Random):** Data values are MCAR if the event that causes them to be missing is independent of other variables. Accordingly, there is no relationship between the presence of missing values, and the values of either the present or absent variables.
2. **MAR (Missing at Random):** Data is MAR if the event that caused the values to be missing is independent of the variable whose values are missing given other non-missing variables.
3. **MNAR (Missing Not at Random):** Data values are MNAR if the event that causes the values to be missing is correlated with the variable whose values are missing.

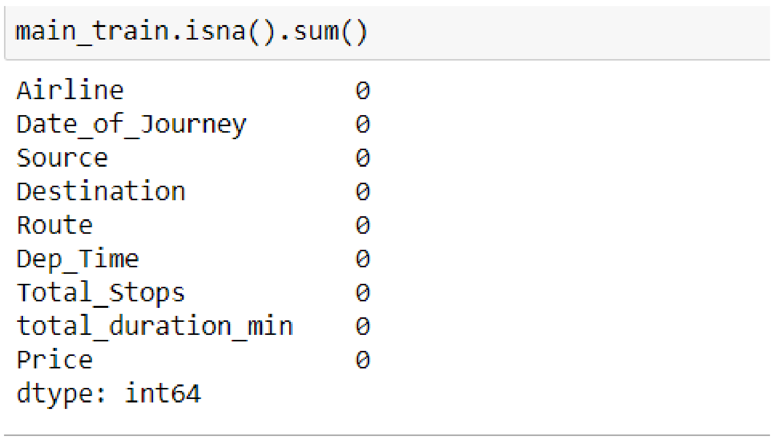
As our attribute ROUTE is MCAR because it is independent of the other variables and the total stop is related and only single row so does not affect that much.



```
main_train[main_train.Route.isnull()]
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Total_Stops	total_duration_
9038	Air India	06-05-2019	Delhi	Cochin	NaN	09:45	NaN	1

Finally, we got the all attributes without null values.

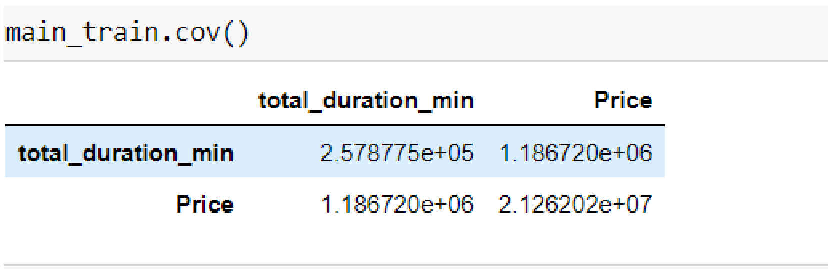


```
main_train.isna().sum()
```

```
Airline               0
Date_of_Journey       0
Source                0
Destination           0
Route                 0
Dep_Time              0
Total_Stops           0
total_duration_min    0
Price                 0
dtype: int64
```

REMOVING CONSTANT OR DUPLICATE REMOVAL:

We need to remove duplicate values to get more accurate, so here first I tried to display confusion matrix for this removal of duplicate values between the Price attribute and the total_duration_min attribute.



```
main_train.cov()
```

	total_duration_min	Price
total_duration_min	2.578775e+05	1.186720e+06
Price	1.186720e+06	2.126202e+07

But we cannot take these values we need only integer or float value data frame so we implemented variancethreshold from sklearn to get unique values.

```
main_train.nunique()
```

```
Airline                 12
Date_of_Journey         40
Source                   5
Destination              6
Route                  128
Dep_Time               222
Total_Stops              5
total_duration_min     367
Price                 1870
dtype: int64
```

- ➔ This is the count of unique values in each attribute.
- ➔ Now we are going to find the number of unique datapoints in airline attribute and the starting place and the destination of the particular flight and the number of stops in between.

```
for col in main_train.columns:
    if main_train[col].nunique()<= 12:
        print('--------------------------------------------')
        print(main_train[col].value_counts())
        print('--------------------------------------------')
```

```
--------------------------------------------
Jet Airways                          3849
IndiGo                               2053
Air India                            1750
Multiple carriers                    1196
SpiceJet                              818
Vistara                               479
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
--------------------------------------------
--------------------------------------------
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       696
Chennai      381
Name: Source, dtype: int64
--------------------------------------------
--------------------------------------------
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     696
Kolkata       381
Name: Destination, dtype: int64
--------------------------------------------
--------------------------------------------
```

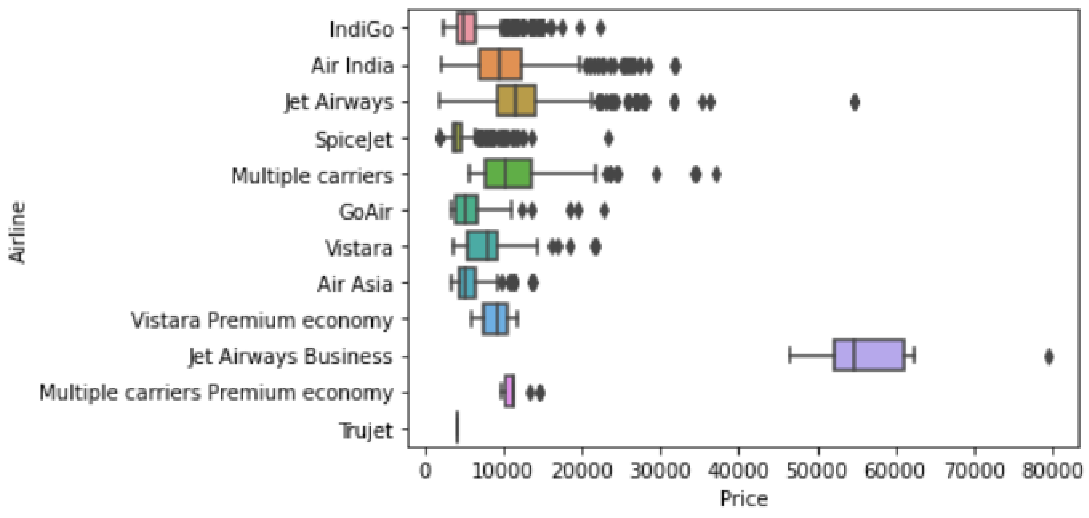
```
-------------------------------------------
1 stop       5625
non-stop     3491
2 stops      1519
3 stops        45
4 stops         1
Name: Total_Stops, dtype: int64
-------------------------------------------
```

➔ Now we are going to merge the airlines attribute with low occurrence in dataset to ensure model gets maximum information.

```
main_train['Airline'] = np.where(main_train['Airline'].isin(["Multiple carriers Premium economy",
                                                             "Jet Airways Business",
                                                             "Vistara Premium economy",
                                                             "Trujet"]),"Other Airlines",main_train['Airline'])
```

➔ Date pre-processing is done, we cleaned the data and now we are going to visualize the data between different attributes in the dataset with different plots available in matplot library.

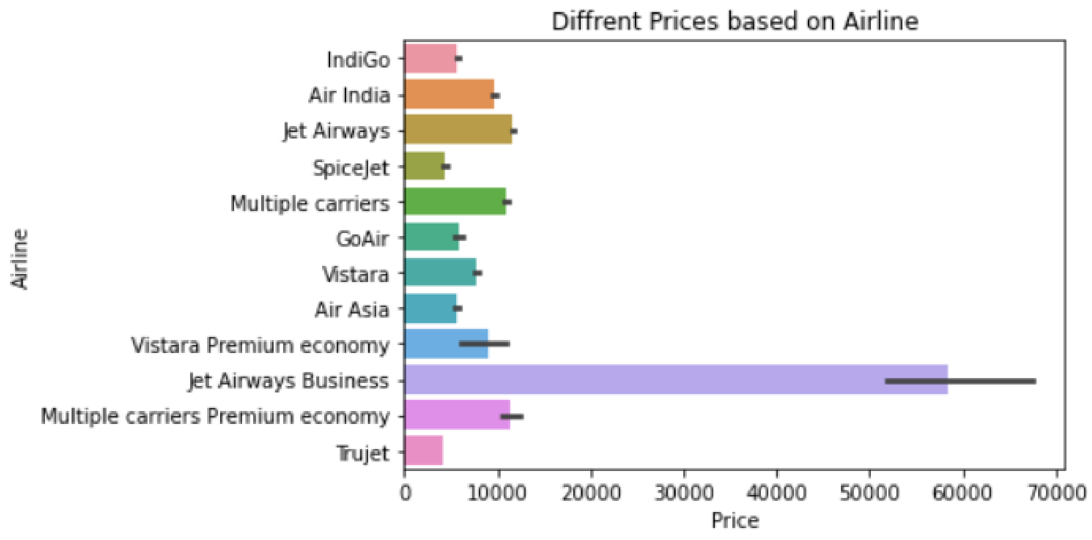
```
g=sns.boxplot(x=df['Price'],y=df['Airline'])
```



➔ Box plot between price and airline attribute.

```
g=sns.barplot(x=df['Price'],y=df['Airline'])
g.set_title('Diffrent Prices based on Airline')
```

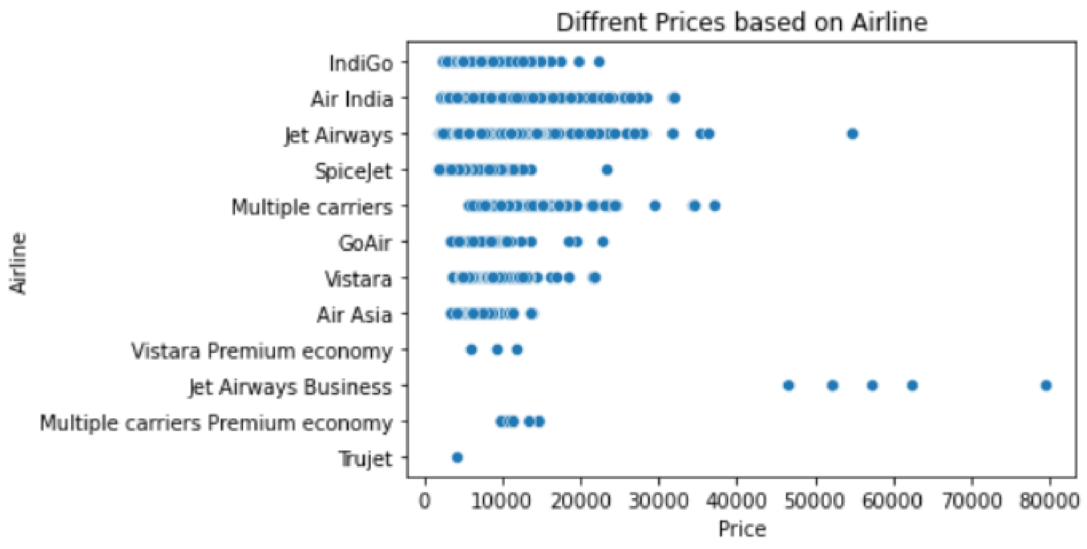
```
Text(0.5, 1.0, 'Diffrent Prices based on Airline')
```



➔ Bar plot between price and airline attribute.

```
g=sns.scatterplot(x=df['Price'],y=df['Airline'])
g.set_title('Diffrent Prices based on Airline')
```

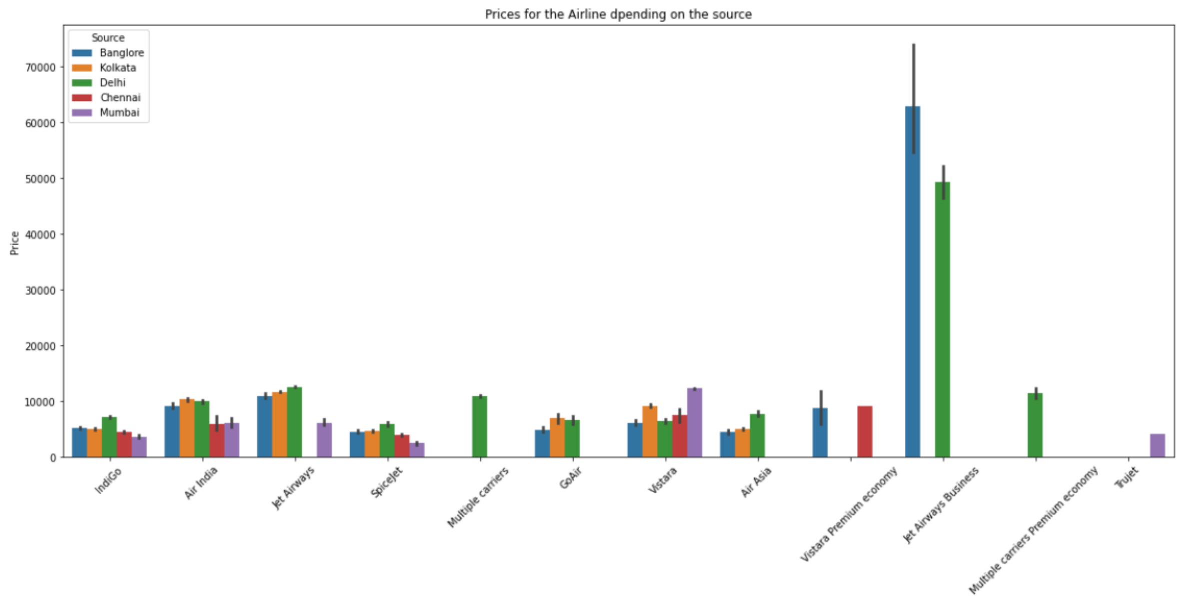
```
Text(0.5, 1.0, 'Diffrent Prices based on Airline')
```



➔ Scatter plot between price and airline attribute.

```
plt.figure(figsize=(20,8))
g=sns.barplot(x=df['Airline'],y=df['Price'],hue=df['Source'])
plt.xticks(rotation=45)
#g.set_xticklabels(labels=df['Airline'],rotation=40)
g.set_title('Prices for the Airline dpending on the source')
```

```
Text(0.5, 1.0, 'Prices for the Airline dpending on the source')
```



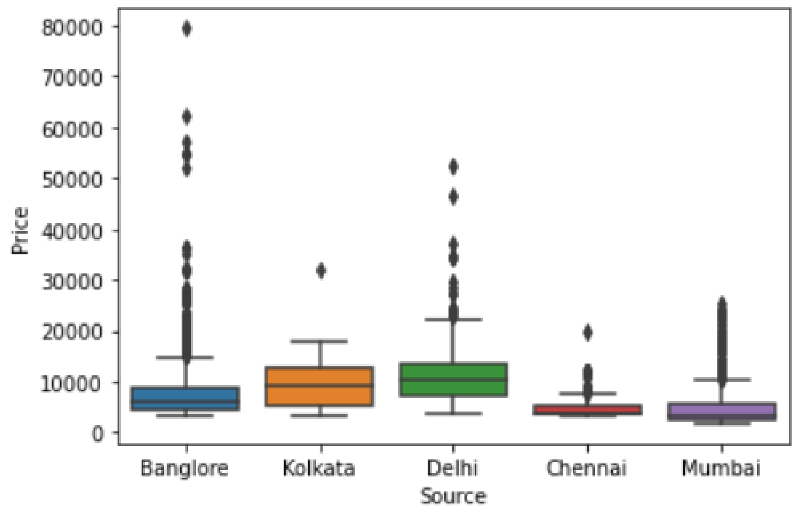
➔ Bar plot between the price and airline with the prices for the airline depending on the source.

CHECKING OUTLIERS:

An outlier is a data point that varies greatly from other results. Outliers can spoil and deceive the training process of machine learning models, resulting in less accurate models and eventually bad performance. Outliers in data can be observed using number of techniques and one of those popular techniques is VISUALIZATION TECHNIQUE. To find outliers, we simply plot the box plot.

```
sns.boxplot(x=df['Source'],y=df['Price'])
```

```
<AxesSubplot:xlabel='Source', ylabel='Price'>
```



➔ The Outliers are the points that are outside of the minimum and maximum values, as we can see from the above screen shot.

SPLIT DATA

```
# Converting the main_train into X and y so that we can pass it

# X --> contains the dataframe without the target i.e price
X = main_train.drop('Price',axis=1)

# y --> contains only the target value
y = main_train['Price']
```

In the above code we are dividing the data so that we can pass it onto train_test_split function.

We have used x,y variables where x is the dataframe without the target to be

```
#splitting the data first into two part -- doing a 70:30 split i.e 30% data fed to intermediate test data set
from sklearn.model_selection import train_test_split
X_train,X_inter_test,y_train,y_inter_test = train_test_split(X,y,test_size=0.3,random_state=0 , shuffle = False)
```

```
#
X_val,X_test,y_val,y_test = train_test_split(X_inter_test,y_inter_test,test_size=0.5,random_state=0 , shuffle = Fal
```

```
X_train.shape , X_val.shape , X_test.shape
```

predicted and y is only the values of the target.

Here we are splitting the data first into two part -- doing a 70:30 split i.e 30% data fed to intermediate test data set

EDA:

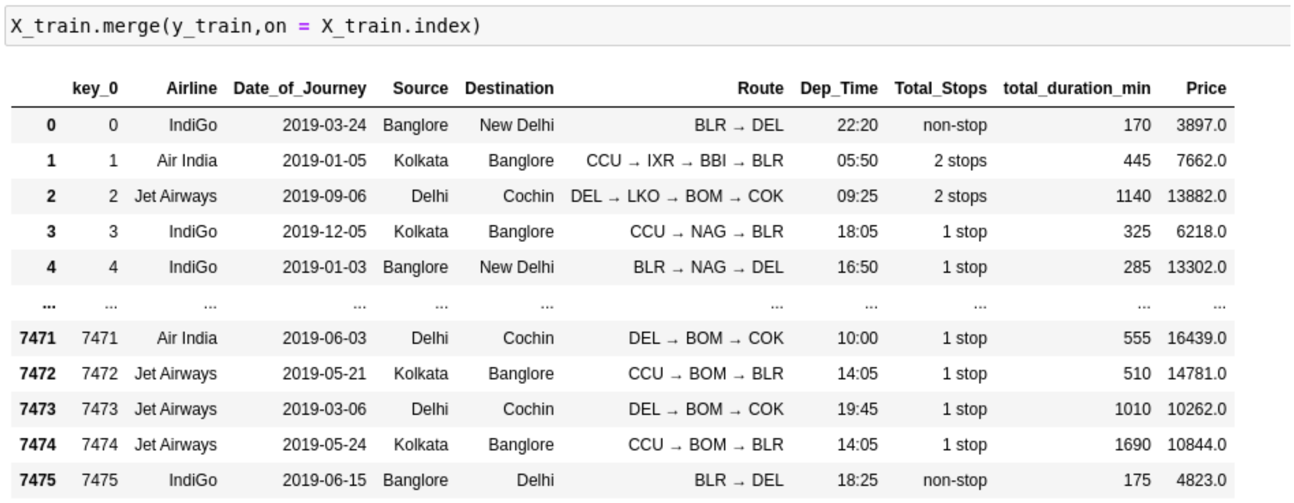
Exploratory Data Analysis, or EDA, is an important step in any Data Analysis or Data Science project.
EDA is the process of investigating the dataset to discover patterns, and anomalies (outliers), and form hypotheses based on our understanding of the dataset.

```
# distribution of the target column
# right skewed distribution
# outliers present but not that significant , hence
sns.distplot(y_train)
```

In the above code we plot the distribution using sns.distplot.

```
X_train.merge(y_train,on = X_train.index)
```

We merge the values in x and y using merge() function and the we get the output like below



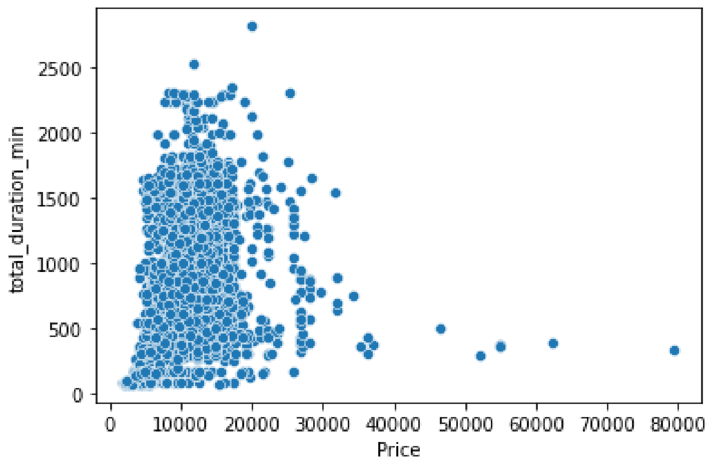
```
X_train.merge(y_train,on = X_train.index)
```

	key_0	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Total_Stops	total_duration_min	Price
0	0	IndiGo	2019-03-24	Banglore	New Delhi	BLR → DEL	22:20	non-stop	170	3897.0
1	1	Air India	2019-01-05	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	2 stops	445	7662.0
2	2	Jet Airways	2019-09-06	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	2 stops	1140	13882.0
3	3	IndiGo	2019-12-05	Kolkata	Banglore	CCU → NAG → BLR	18:05	1 stop	325	6218.0
4	4	IndiGo	2019-01-03	Banglore	New Delhi	BLR → NAG → DEL	16:50	1 stop	285	13302.0
...	...	...	...	...	...	...	...	...	...	...
7471	7471	Air India	2019-06-03	Delhi	Cochin	DEL → BOM → COK	10:00	1 stop	555	16439.0
7472	7472	Jet Airways	2019-05-21	Kolkata	Banglore	CCU → BOM → BLR	14:05	1 stop	510	14781.0
7473	7473	Jet Airways	2019-03-06	Delhi	Cochin	DEL → BOM → COK	19:45	1 stop	1010	10262.0
7474	7474	Jet Airways	2019-05-24	Kolkata	Banglore	CCU → BOM → BLR	14:05	1 stop	1690	10844.0
7475	7475	IndiGo	2019-06-15	Banglore	Delhi	BLR → DEL	18:25	non-stop	175	4823.0

```
#No direct trend
sns.scatterplot(data=X_train.merge(y_train,on = X_train.index), x="Price", y="total_duration_min")
```

And again we plot the scatterplot using scatterplot() function for the price and total duration from the merged file and we got the output like below

```
<AxesSubplot:xlabel='Price', ylabel='total_duration_min'>
```



```
X_train.Route.nunique()
```

```
122
```

We got the unique values using nunique() function

<u>**Index.nunique()**</u> function return number of unique elements in the object. It returns a scalar value which is the count of all the unique values in the Index. By default the NaN values are not included in the count. If dropna parameter is set to be False then it includes NaN value in the count.

```
# cheapest / costliest --> flights per route
# flights which provides services to most of the routes
X_train.merge(y_train,on = X_train.index).groupby(['Route','Airline'])['Price'].agg(['min']).unstack().T.idxmin()
```

```
BLR → BOM → UDR → DEL                (min, Air India)
BLR → CCU → BBI → DEL                (min, Air India)
BLR → CCU → BBI → HYD → DEL          (min, Air India)
BLR → CCU → DEL                      (min, Air India)
BLR → CCU → GAU → DEL                (min, Air India)
BLR → COK → DEL                      (min, Air India)
BLR → DEL                             (min, SpiceJet)
BLR → GAU → DEL                      (min, Air India)
BLR → GOI → DEL                         (min, IndiGo)
BLR → HBX → BOM → AMD → DEL          (min, Air India)
BLR → HBX → BOM → BHO → DEL          (min, Air India)
BLR → HYD → DEL                         (min, IndiGo)
BLR → HYD → VGA → DEL                (min, Air India)
BLR → IDR → DEL                         (min, IndiGo)
BLR → LKO → DEL                         (min, IndiGo)
BLR → MAA → DEL                      (min, Air India)
BLR → NAG → DEL                         (min, IndiGo)
BLR → PNQ → DEL                       (min, SpiceJet)
BLR → STV → DEL                         (min, IndiGo)
BLR → TRV → COK → DEL                (min, Air India)
```

In the above code we have used merge() function and groupby() function to merge the x and y datas and grouped it by route ,airline.

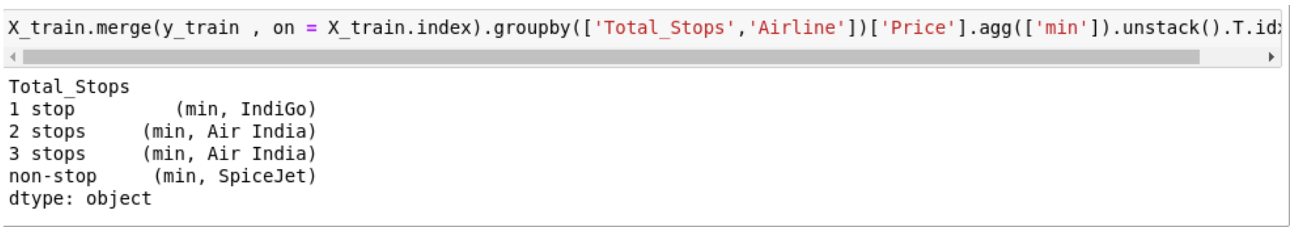
<u>**groupby()**</u>

In Pandas **dataframe.groupby()** function is used to split the data into groups based on some criteria. pandas' objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names.

Agg ():

Series.agg () is used to pass a function or list of function to be applied on a series or even each element of series separately.

In case of list of function, multiple results are returned by agg() method.



```
X_train.merge(y_train , on = X_train.index).groupby(['Total_Stops','Airline'])['Price'].agg(['min']).unstack().T.id
```

```
Total_Stops
1 stop          (min, IndiGo)
2 stops      (min, Air India)
3 stops      (min, Air India)
non-stop      (min, SpiceJet)
dtype: object
```

We have grouped the data like non-stop and with stops using the above code snippet using the groupby () and agg ().

Feature Engineering:

In this step we mainly work on the data set and do some transformation like creating different bins of particular columns, clean the messy data so that it can be used in our ML model. This step is very important because for a high prediction score you need to continuously make changes in it.

```python
from sklearn.preprocessing import LabelEncoder
Airline_encoder = LabelEncoder()
Airline_encoder.fit(X_train['Airline'])
X_train['Airline_enc'] = Airline_encoder.transform(X_train['Airline'])
X_train[['Airline_enc','Airline']]
```

	Airline_enc	Airline
0	3	IndiGo
1	1	Air India
2	4	Jet Airways
3	3	IndiGo
4	3	IndiGo
...	...	...
7471	1	Air India
7472	4	Jet Airways
7473	4	Jet Airways
7474	4	Jet Airways
7475	3	IndiGo

7476 rows × 2 columns

To convert categorical text data into model-understandable numerical data, we use the Label Encoder class. So, all we have to do, to label encode a column is import the LabelEncoder class from the sklearn library, fit and transform the column of the data, and then replace the existing text data with the new encoded data.

```python
X_val['Airline_enc'] = Airline_encoder.transform(X_val['Airline'])
X_test['Airline_enc'] = Airline_encoder.transform(X_test['Airline'])
```

Now that all our data is numerical after label encoding so we split the data into test and train.

```python
# function will delete a column from train , val and test
def thanos_snap(col ,traindf = X_train,valdf = X_val,testdf = X_test):
    traindf.drop(col, axis =1,inplace=True)
    valdf.drop(col,axis=1 , inplace=True)
    testdf.drop(col,axis=1 , inplace=True)

    return traindf,valdf ,testdf
```

This function drops the particular required column whenever column name is passed as shown below.

```
X_train ,X_val, X_test = thanos_snap(['Date_of_Journey'])
```

```
#extract day,month and year

X_train['day_of_Journey'] = X_train['Date_of_Journey'].dt.day
X_train['month_of_Journey'] = X_train['Date_of_Journey'].dt.month
X_train['year_of_Journey'] = X_train['Date_of_Journey'].dt.year

X_val['day_of_Journey'] = X_val['Date_of_Journey'].dt.day
X_val['month_of_Journey'] = X_val['Date_of_Journey'].dt.month
X_val['year_of_Journey'] = X_val['Date_of_Journey'].dt.year


X_test['day_of_Journey'] = X_test['Date_of_Journey'].dt.day
X_test['month_of_Journey'] = X_test['Date_of_Journey'].dt.month
X_test['year_of_Journey'] = X_test['Date_of_Journey'].dt.year
```

In the column ‘Date_of_Journey’, we can see the date format is given as dd/mm/yyyy and as you can see the datatype is given as object So there is two ways to tackle this column, either convert the column into Timestamp or divide the column into date,Month ,Year.

Here, i am splitting the columns

```
# label enceode route as well
route_encoder = LabelEncoder()

route_encoder.fit(X_train['Route'])

X_train['Route_enc'] = route_encoder.transform(X_train['Route'])
```

The ‘Route’ columns mainly tell us that how many cities they have taken to reach from source to destination. This column is very important because based on the route they took will directly affect the price of the flight So We split the Route column to extract the information.

Feature Selection:

Feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction.

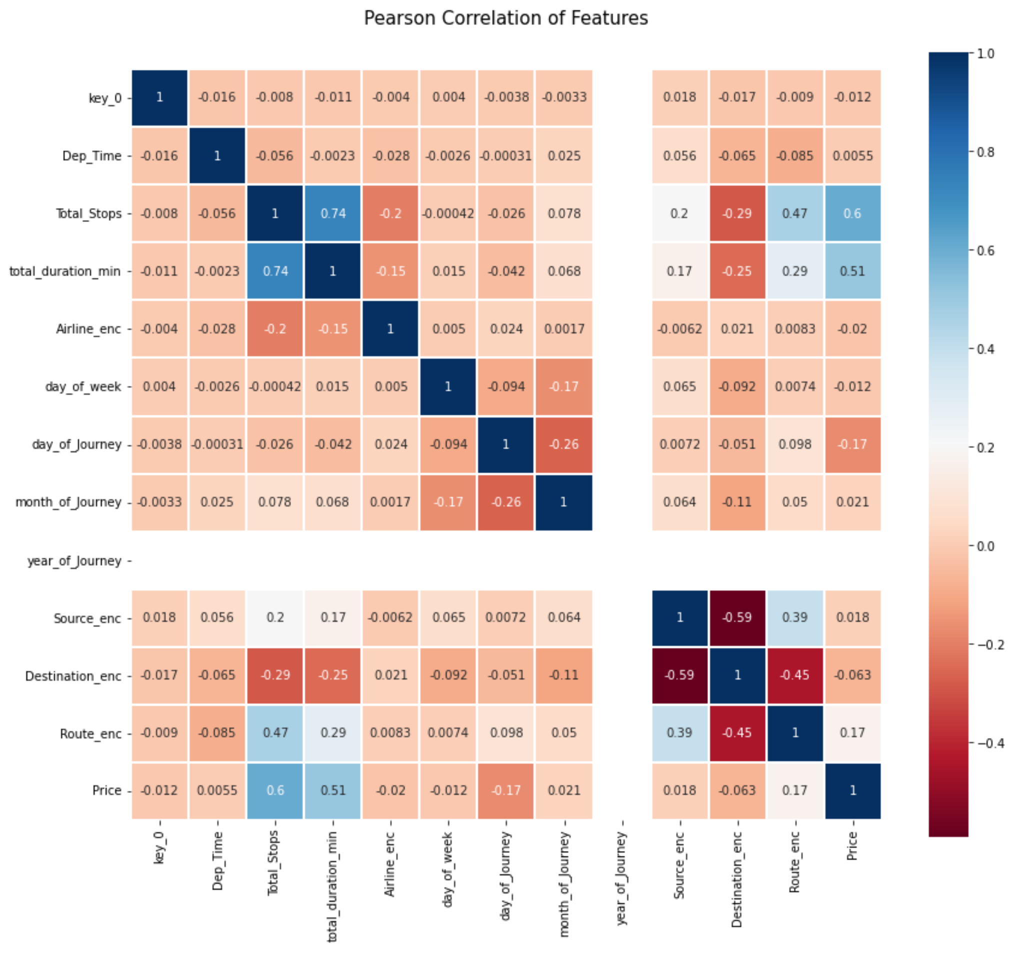
```
#Feature Selection

#corrmat = X_train.merge(y_train , on = X_train.index).corr()
# plt.subplots(figsize=(12,9))
# sns.heatmap(corrmat, vmax=0.9, square=True)


colormap = plt.cm.RdBu
plt.figure(figsize=(14,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(X_train.merge(y_train , on = X_train.index ).corr(),linewidths=0.1,vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True)
```

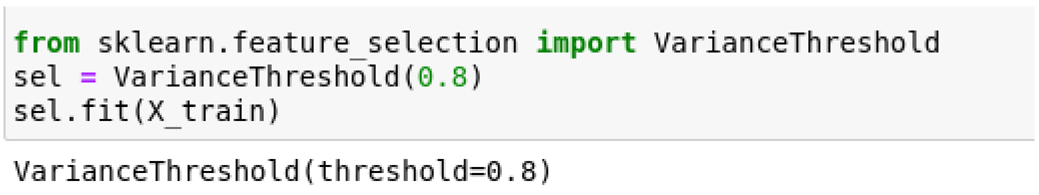
We have plotted heatmap using the above code and we got the output like below

Sns.heatmap():

This is an Axes-level function and will draw the heatmap into the currently-active Axes if none is provided to the x argument. Part of this Axes space will be taken and used to plot a colormap, unless cbar is False or a separate Axes is provided to x.

```
Out[60]: <AxesSubplot:title={'center':'Pearson Correlation of Features'}>
```





```
from sklearn.feature_selection import VarianceThreshold
sel = VarianceThreshold(0.8)
sel.fit(X_train)
```

```
VarianceThreshold(threshold=0.8)
```

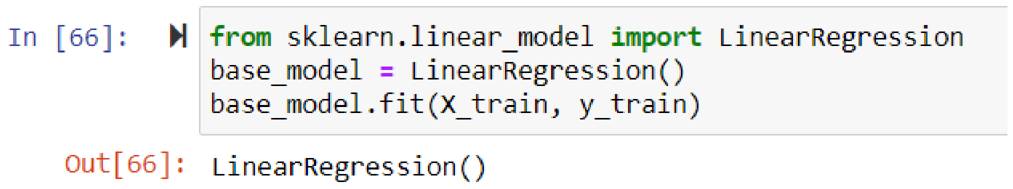
We have fixed the variance threshold value using variance Theshold().

Variance threshold:

This feature selection algorithm looks only at the features (X), not the desired outputs (y), and can thus be used for unsupervised learning.

MODELLING

⇨ **Creating base model**



```
In [66]:  from sklearn.linear_model import LinearRegression
          base_model = LinearRegression()
          base_model.fit(X_train, y_train)

Out[66]:  LinearRegression()
```

Linear Regression

To determine the correlation between two continuous variables, simple linear regression analysis is used. One of the two variables is the predictor variable of which value is to be found. It gives the statistical relationship not the deterministic relationship between two variables. Linear regression algorithm gives the best fit line to the given data for which the prediction error is minimum. Gradient descent and cost function are the two major factors to understand linear regression. The equation for linear regression is:

$$y(pred) = b0+b1 * x \quad (1)$$

The value of coefficients b1 and b0 are chosen so that the error value is as small as possible. The square of predicted and actual value difference gives the error. To deal with the negative values, the mean square error is taken (MSE). Here b0 gives the positive or negative relationship between the x and y, whereas b1 is called bias. The accuracy of the regression problem is measured in terms of R-squared, MAE and MSE.

Mean Square Error

```
In [99]:  from sklearn.metrics import mean_squared_error
          mean_squared_error(y_val, y_predict, squared=False)

Out[99]:  3309.889496745601
```

The Mean Squared Error (MSE) of an estimator measures the average of error squares i.e., the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better. The MSE is the second moment of the error (about the origin) and thus incorporates both the variance of the estimator and its bias.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \bar{y}_i)^2$$

Mean Absolute Error

Absolute error refers to the magnitude of difference between the prediction of an observation and the true value of that observation. MAE takes the average of absolute errors for a group of predictions and observations as a measurement of the magnitude of errors for the entire group.

$$\text{MAE} = \frac{1}{n} \sum_{j=1}^{n} |y_j - y_j|$$

⇨ Tree based model

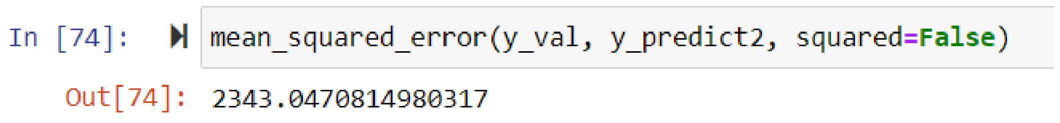
```
In [71]:  from sklearn.tree import DecisionTreeRegressor
          regressor = DecisionTreeRegressor(random_state=0 , max_depth=5)
```

Decision Tree

Decision Tree is a decision-making tool that uses a flowchart-like tree structure or is a model of decisions and all of their possible results, including outcomes, input costs, and utility. Decision-tree algorithm falls under the category of supervised learning algorithms. It works for both continuous as well as categorical output variables.

Decision Tree Regressor

Decision tree regression observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output. Continuous output means that the output/result is not discrete, i.e., it is not represented just by a discrete, known set of numbers or values.



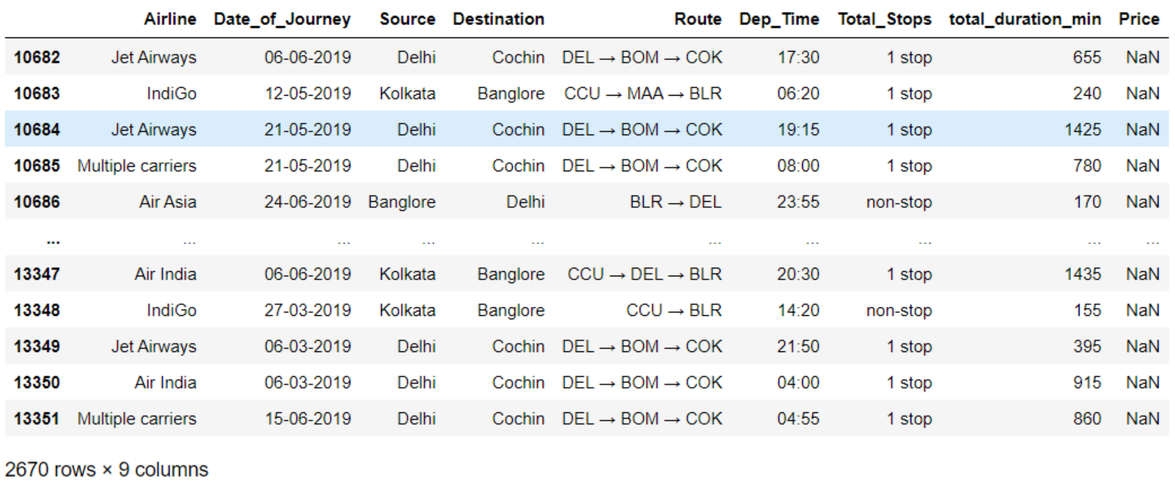
```
In [74]:  mean_squared_error(y_val, y_predict2, squared=False)

Out[74]:  2343.0470814980317
```

The lost data

So, in this point we have

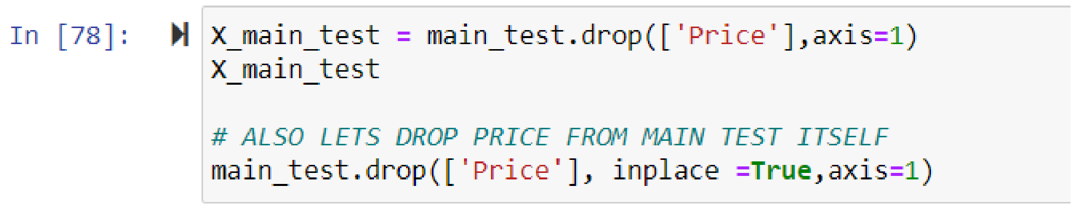
⇨ A model i.e., regressor
⇨ main_test
⇨ some encoders to map i.e., airline_enc etc

We have to predict the price using the model and fill it up in the price column



	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Total_Stops	total_duration_min	Price
10682	Jet Airways	06-06-2019	Delhi	Cochin	DEL → BOM → COK	17:30	1 stop	655	NaN
10683	IndiGo	12-05-2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	1 stop	240	NaN
10684	Jet Airways	21-05-2019	Delhi	Cochin	DEL → BOM → COK	19:15	1 stop	1425	NaN
10685	Multiple carriers	21-05-2019	Delhi	Cochin	DEL → BOM → COK	08:00	1 stop	780	NaN
10686	Air Asia	24-06-2019	Banglore	Delhi	BLR → DEL	23:55	non-stop	170	NaN
...	...	...	...	...	...	...	...	...	...
13347	Air India	06-06-2019	Kolkata	Banglore	CCU → DEL → BLR	20:30	1 stop	1435	NaN
13348	IndiGo	27-03-2019	Kolkata	Banglore	CCU → BLR	14:20	non-stop	155	NaN
13349	Jet Airways	06-03-2019	Delhi	Cochin	DEL → BOM → COK	21:50	1 stop	395	NaN
13350	Air India	06-03-2019	Delhi	Cochin	DEL → BOM → COK	04:00	1 stop	915	NaN
13351	Multiple carriers	15-06-2019	Delhi	Cochin	DEL → BOM → COK	04:55	1 stop	860	NaN

2670 rows × 9 columns



```
In [78]:  X_main_test = main_test.drop(['Price'],axis=1)
          X_main_test

          # ALSO LETS DROP PRICE FROM MAIN TEST ITSELF
          main_test.drop(['Price'], inplace =True,axis=1)
```

⇨ We have to bring the main_test in same format as that of the X_train
⇨ Now repeat the pre-processing
⇨ No EDA

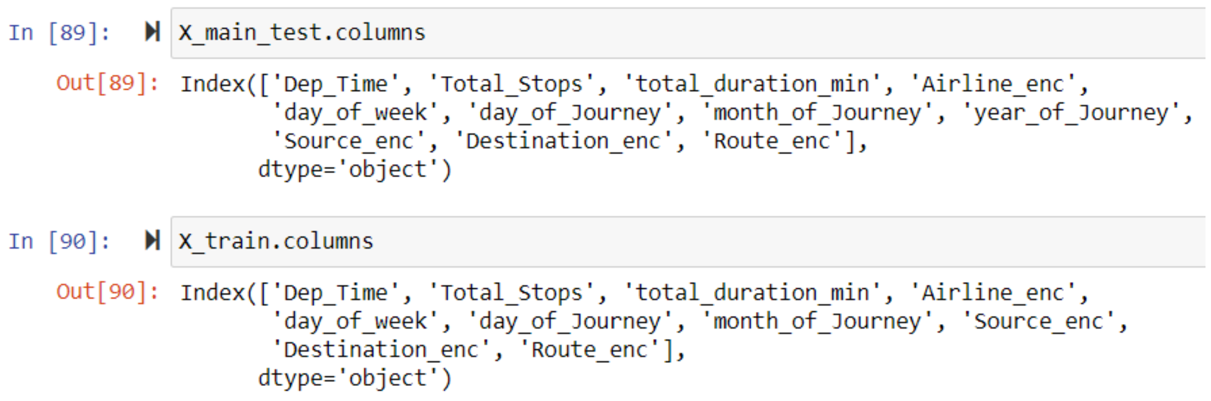
```
In [77]:  main_test.isna().sum()

Out[77]: Airline                0
         Date_of_Journey        0
         Source                 0
         Destination            0
         Route                  0
         Dep_Time               0
         Total_Stops            0
         total_duration_min     0
         Price               2670
         dtype: int64
```

➔Drop price column as we are going to predict

```
In [78]:  X_main_test = main_test.drop(['Price'],axis=1)
          X_main_test

          # ALSO LETS DROP PRICE FROM MAIN TEST ITSELF
          main_test.drop(['Price'], inplace =True,axis=1)
```

```
In [87]:  X_main_test.info()

          <class 'pandas.core.frame.DataFrame'>
          Int64Index: 2670 entries, 10682 to 13351
          Data columns (total 16 columns):
           #   Column              Non-Null Count  Dtype
          ---  ------              --------------  -----
           0   Airline             2670 non-null   object
           1   Date_of_Journey     2670 non-null   datetime64[ns]
           2   Source              2670 non-null   object
           3   Destination         2670 non-null   object
           4   Route               2670 non-null   object
           5   Dep_Time            2670 non-null   int32
           6   Total_Stops         2670 non-null   int32
           7   total_duration_min  2670 non-null   int64
           8   Airline_enc         2670 non-null   int32
           9   day_of_week         2670 non-null   int64
           10  day_of_Journey      2670 non-null   int64
           11  month_of_Journey    2670 non-null   int64
           12  year_of_Journey     2670 non-null   int64
           13  Source_enc          2670 non-null   int32
           14  Destination_enc     2670 non-null   int32
           15  Route_enc           2670 non-null   int32
          dtypes: datetime64[ns](1), int32(6), int64(5), object(4)
          memory usage: 292.0+ KB
```

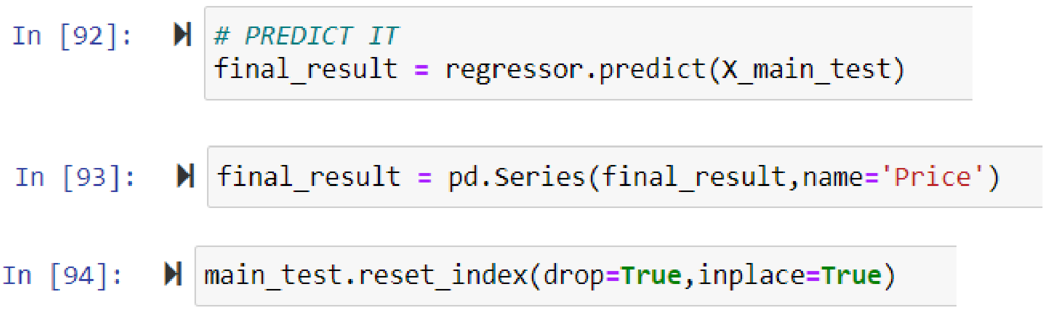
➔ Total info of the main_test will be given here



```
In [89]:  X_main_test.columns

Out[89]: Index(['Dep_Time', 'Total_Stops', 'total_duration_min', 'Airline_enc',
               'day_of_week', 'day_of_Journey', 'month_of_Journey', 'year_of_Journey',
               'Source_enc', 'Destination_enc', 'Route_enc'],
              dtype='object')

In [90]:  X_train.columns

Out[90]: Index(['Dep_Time', 'Total_Stops', 'total_duration_min', 'Airline_enc',
               'day_of_week', 'day_of_Journey', 'month_of_Journey', 'Source_enc',
               'Destination_enc', 'Route_enc'],
              dtype='object')
```

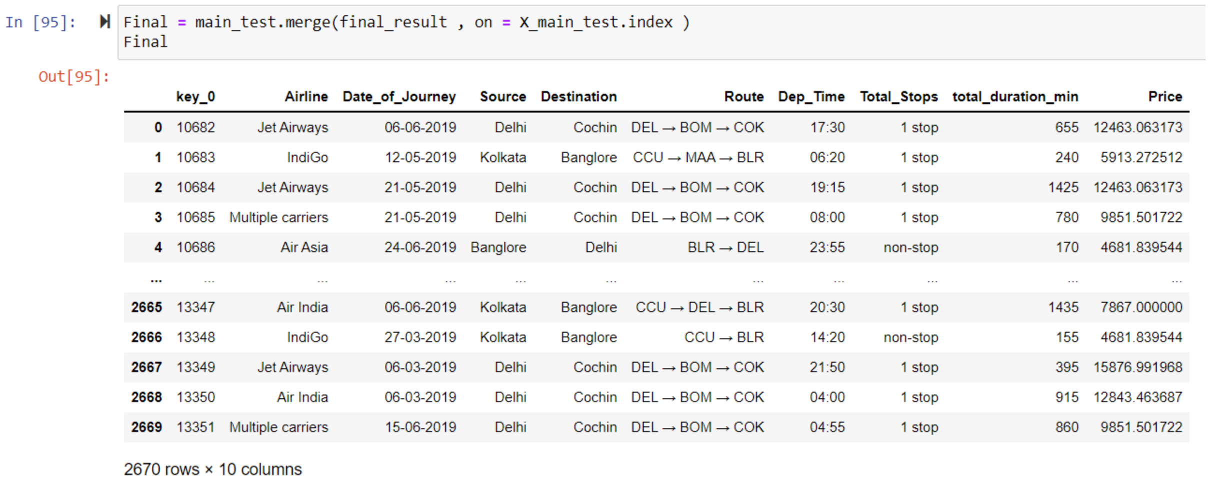
➔ All the columns in train set will be output here

Predicting



```
In [92]:  # PREDICT IT
          final_result = regressor.predict(X_main_test)

In [93]:  final_result = pd.Series(final_result,name='Price')

In [94]:  main_test.reset_index(drop=True,inplace=True)
```

Here we are predicting the price using regressor and resetting the index of main test so that we can merge price series with it.



```
In [95]:  Final = main_test.merge(final_result , on = X_main_test.index )
          Final

Out[95]:
```

	key_0	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Total_Stops	total_duration_min	Price
0	10682	Jet Airways	06-06-2019	Delhi	Cochin	DEL → BOM → COK	17:30	1 stop	655	12463.063173
1	10683	IndiGo	12-05-2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	1 stop	240	5913.272512
2	10684	Jet Airways	21-05-2019	Delhi	Cochin	DEL → BOM → COK	19:15	1 stop	1425	12463.063173
3	10685	Multiple carriers	21-05-2019	Delhi	Cochin	DEL → BOM → COK	08:00	1 stop	780	9851.501722
4	10686	Air Asia	24-06-2019	Banglore	Delhi	BLR → DEL	23:55	non-stop	170	4681.839544
...	...	...	...	...	...	...	...	...	...	...
2665	13347	Air India	06-06-2019	Kolkata	Banglore	CCU → DEL → BLR	20:30	1 stop	1435	7867.000000
2666	13348	IndiGo	27-03-2019	Kolkata	Banglore	CCU → BLR	14:20	non-stop	155	4681.839544
2667	13349	Jet Airways	06-03-2019	Delhi	Cochin	DEL → BOM → COK	21:50	1 stop	395	15876.991968
2668	13350	Air India	06-03-2019	Delhi	Cochin	DEL → BOM → COK	04:00	1 stop	915	12843.463687
2669	13351	Multiple carriers	15-06-2019	Delhi	Cochin	DEL → BOM → COK	04:55	1 stop	860	9851.501722

2670 rows × 10 columns

We can see here; we predicted the price of each airline which are null values previously. There are 2670 null values in price column in main_test before our prediction, after using the model.

References:

1. https://dalspace.library.dal.ca/bitstream/handle/10222/79128/Wang-Zhenbang-MCSc-CSCI-April-2020.pdf?sequence=1
2. https://www.researchgate.net/publication/335936877_A_Framework_for_Airfare_Price_Prediction_A_Machine_Learning_Approach
3. http://www.ijstr.org/final-print/dec2019/Predicting-The-Price-Of-A-Flight-Ticket-With-The-Use-Of-Machine-Learning-Algorithms.pdf
4. https://medium.com/geekculture/flight-price-prediction-using-machine-learning-f18b4bdc70e6
5. https://www.ijert.org/a-survey-on-flight-pricing-prediction-using-machine-learning
6. https://medium.com/swlh/predicting-airfare-price-using-machine-learning-techniques-bf3a13ad07d1
7. https://github.com/rishabdhar12/Flight-Price-Prediction
8. https://www.analyticsvidhya.com/blog/2021/06/flight-price-prediction-a-regression-analysis-using-lazy-prediction/
9. https://www.kaggle.com/vinayshaw/airfare-price-prediction