

NAME OF THE PROJECT

# **FLIGHT PRICE PREDICTION**

Submitted by:

**RANJEET JANAGOUDA**

## **ACKNOWLEDGMENT**

First and foremost, I would like to thank Flip Robo Technologies to provide me a chance to work on this project. It was a great experience to work on this project under your guidance.

I would like to present my gratitude to the following websites:

- [Zendesk](#)
- [Kaggle](#)
- [Datatrained Notes](#)
- [Sklearn.org](#)
- [Crazyegg](#)
- [kayak.com](#)
- [Youtube.com](#)

These websites were of great help and due to this, I was able to complete my project effectively and efficiently.

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

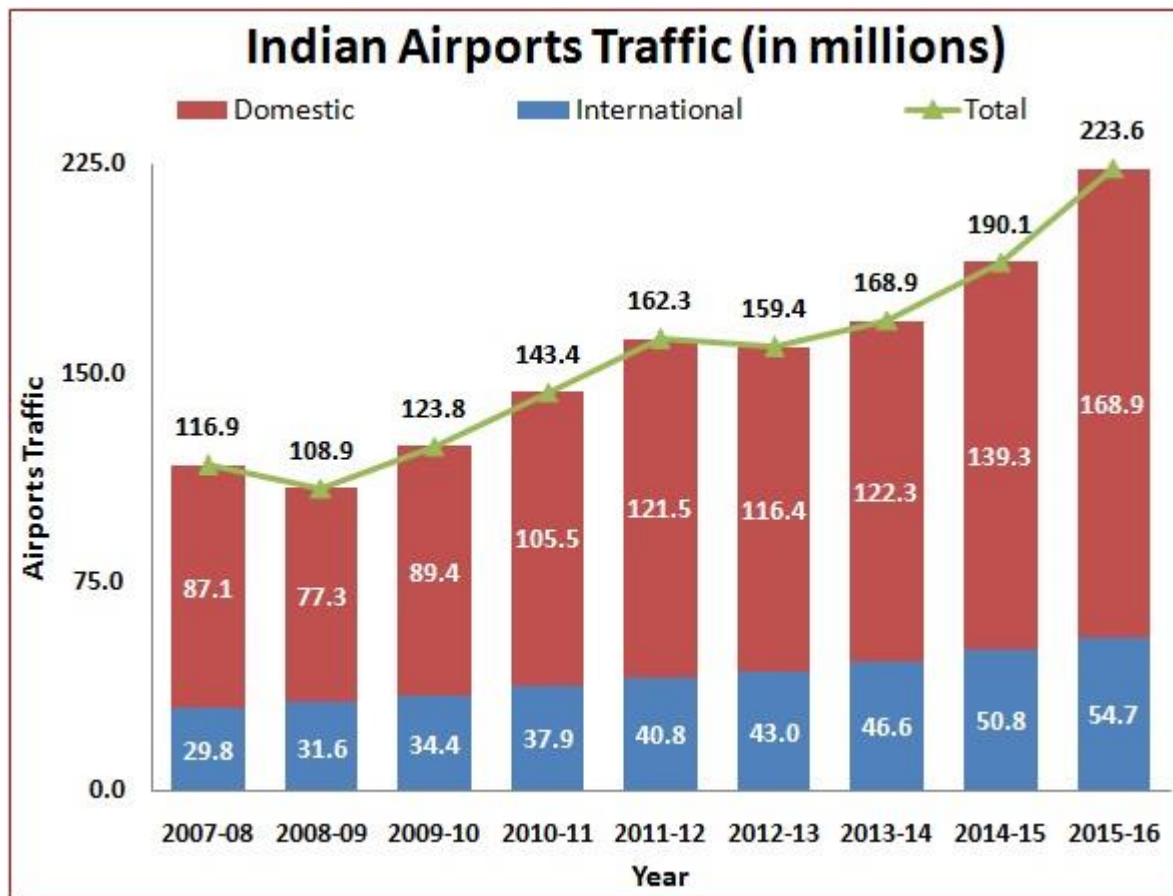
- Conceptual Background of the Domain Problem

One should know how to scrap the data from a website using the selenium as we have to create a fresh data for our project. Basic EDA concepts and regression algorithms must be known to work on this project. One should know what factors is important to predict the Flight Price and how it is going to affect ones purchase of the ticket. Why predicting the flight prices is important and how can it is going to help the person to get the cheaper ticket?

- Review of Literature

According to a report, India's civil aviation industry is on a high-growth trajectory. India aims to become the third-largest aviation market by 2020 and the largest by 2030. Indian domestic air traffic is expected to cross 100 million passengers by FY2017, compared to 81 million passengers in 2015, as per Centre for Asia Pacific Aviation (CAPA).

According to Google Trends, the search term - "Cheap Air Tickets" is most searched in India. Moreover, as the middle-class of India is exposed to air travel, consumers hunting for cheap prices increases.



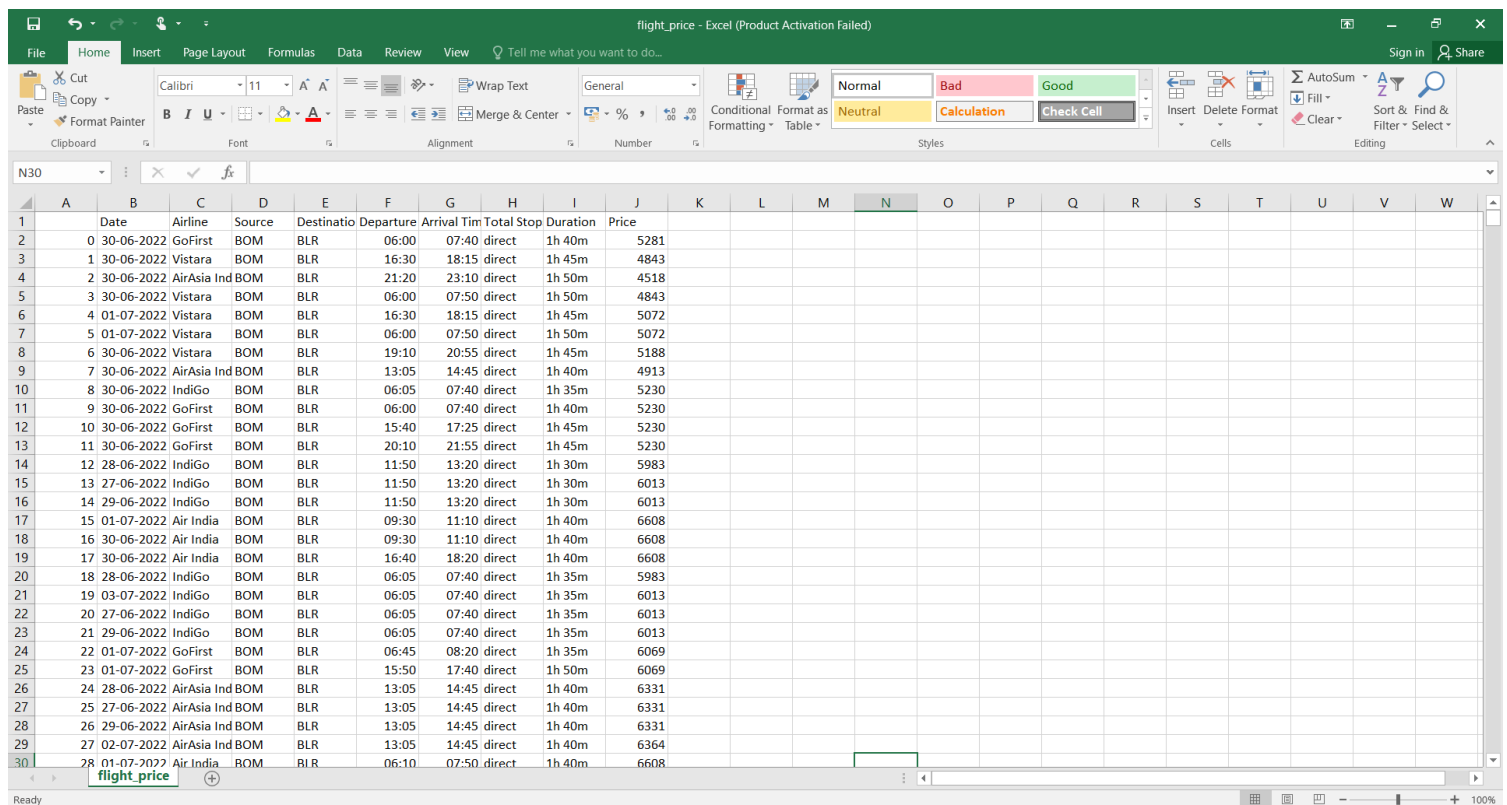
# Analytical Problem Framing

- Data Sources and their formats

In this project our first task is to scrap the data from a flight booking website. My source of data is Kayak website from where I scrap the data of different flights for different routes including departure date, arrival time, price, total stops etc.

To scrap the data, we will use the selenium. With the help of it scrap the data and save it in a csv file format for future use.

It contains 9 columns and 1658 rows. There are so many factors which can be used for the prediction of flight price. Dataset contain both numerical as well as categorical data.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
	Date	Airline	Source	Destination	Departure	Arrival Time	Total Stop	Duration	Price														
1	0 30-06-2022	GoFirst	BOM	BLR	06:00	07:40	direct	1h 40m	5281														
2	1 30-06-2022	Vistara	BOM	BLR	16:30	18:15	direct	1h 45m	4843														
3	2 30-06-2022	AirAsia Ind	BOM	BLR	21:20	23:10	direct	1h 50m	4518														
4	3 30-06-2022	Vistara	BOM	BLR	06:00	07:50	direct	1h 50m	4843														
5	4 01-07-2022	Vistara	BOM	BLR	16:30	18:15	direct	1h 45m	5072														
6	5 01-07-2022	Vistara	BOM	BLR	06:00	07:50	direct	1h 50m	5072														
7	6 30-06-2022	Vistara	BOM	BLR	19:10	20:55	direct	1h 45m	5188														
8	7 30-06-2022	AirAsia Ind	BOM	BLR	13:05	14:45	direct	1h 40m	4913														
9	8 30-06-2022	IndiGo	BOM	BLR	06:05	07:40	direct	1h 35m	5230														
10	9 30-06-2022	GoFirst	BOM	BLR	06:00	07:40	direct	1h 40m	5230														
11	10 30-06-2022	GoFirst	BOM	BLR	15:40	17:25	direct	1h 45m	5230														
12	11 30-06-2022	GoFirst	BOM	BLR	20:10	21:55	direct	1h 45m	5230														
13	12 28-06-2022	IndiGo	BOM	BLR	11:50	13:20	direct	1h 30m	5983														
14	13 27-06-2022	IndiGo	BOM	BLR	11:50	13:20	direct	1h 30m	6013														
15	14 29-06-2022	IndiGo	BOM	BLR	11:50	13:20	direct	1h 30m	6013														
16	15 01-07-2022	Air India	BOM	BLR	09:30	11:10	direct	1h 40m	6608														
17	16 30-06-2022	Air India	BOM	BLR	09:30	11:10	direct	1h 40m	6608														
18	17 30-06-2022	Air India	BOM	BLR	16:40	18:20	direct	1h 40m	6608														
19	18 28-06-2022	IndiGo	BOM	BLR	06:05	07:40	direct	1h 35m	5983														
20	19 03-07-2022	IndiGo	BOM	BLR	06:05	07:40	direct	1h 35m	6013														
21	20 27-06-2022	IndiGo	BOM	BLR	06:05	07:40	direct	1h 35m	6013														
22	21 29-06-2022	IndiGo	BOM	BLR	06:05	07:40	direct	1h 35m	6013														
23	22 01-07-2022	GoFirst	BOM	BLR	06:45	08:20	direct	1h 35m	6069														
24	23 01-07-2022	GoFirst	BOM	BLR	15:50	17:40	direct	1h 50m	6069														
25	24 28-06-2022	AirAsia Ind	BOM	BLR	13:05	14:45	direct	1h 40m	6331														
26	25 27-06-2022	AirAsia Ind	BOM	BLR	13:05	14:45	direct	1h 40m	6331														
27	26 29-06-2022	AirAsia Ind	BOM	BLR	13:05	14:45	direct	1h 40m	6331														
28	27 02-07-2022	AirAsia Ind	BOM	BLR	13:05	14:45	direct	1h 40m	6364														
29	28 01-07-2022	Air India	BOM	BLR	06:10	07:50	direct	1h 40m	6608														
30																							

- Libraries Used

I am using different libraries to explore the dataset.

1. Pandas – It is used to load and store the dataset. We can discuss the dataset with the pandas different attributes like .info, .columns, .shape

2. Seaborn – It is used to plot the different types of plots like catplot, lineplot, countplot and more to have a better visualization of the dataset.
3. Matplotlib.pyplot – It helps to give a proper description to the plotted graph by seaborn and make our graph more informative.
4. Numpy – It is the library to perform the numerical analysis to the dataset
5. Sklearn – It is used to for machine learning algorithms. It contains so many attributes & algorithms.

## Load the Dataset

### Importing the libraries

```
In [2]: import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

### Loading the Dataset

```
In [3]: df=pd.read_csv(r'F:\Data Trained\flight_price.csv') #importing the data
df.head()
```

```
Out[3]:
```

	Unnamed: 0	Date	Airline	Source	Destination	Departure Time	Arrival Time	Total Stops	Duration	Price
0	0	30-06-2022	GoFirst	BOM	BLR	06:00	07:40	direct	1h 40m	5281
1	1	30-06-2022	Vistara	BOM	BLR	16:30	18:15	direct	1h 45m	4843
2	2	30-06-2022	AirAsia India	BOM	BLR	21:20	23:10	direct	1h 50m	4518
3	3	30-06-2022	Vistara	BOM	BLR	06:00	07:50	direct	1h 50m	4843
4	4	01-07-2022	Vistara	BOM	BLR	16:30	18:15	direct	1h 45m	5072

We have successfully load our dataset for the further processes.

## Checking the Attributes

- First & last five rows of both the dataset
- Shape of the datasets
- Columns present in the datasets

- Brief info about the datasets
- Null values present in both the dataset
- Unique values in each column
- Datatypes of each column

```
In [4]: ► df.shape #shape of the dataset
```

```
Out[4]: (1658, 10)
```

Dataset has 1652 rows and 10 columns

```
In [5]: ► df.dtypes #datatypes of each column
```

```
Out[5]: Unnamed: 0      int64
Date      object
Airline    object
Source     object
Destination object
Departure Time object
Arrival Time object
Total Stops object
Duration   object
Price      int64
dtype: object
```

```
In [6]: ► df.nunique() #unique value in each column
```

```
Out[6]: Unnamed: 0      1658
Date      7
Airline    12
Source     5
Destination 10
Departure Time 193
Arrival Time 189
Total Stops 3
Duration   115
Price      613
dtype: int64
```

```
In [7]: ► df.info() #a brief info about the dataset
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1658 entries, 0 to 1657
Data columns (total 10 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Unnamed: 0            1658 non-null   int64
1   Date                  1658 non-null   object
2   Airline               1658 non-null   object
3   Source               1658 non-null   object
4   Destination          1658 non-null   object
5   Departure Time       1658 non-null   object
6   Arrival Time         1658 non-null   object
7   Total Stops          1658 non-null   object
8   Duration              1658 non-null   object
9   Price                1658 non-null   int64
dtypes: int64(2), object(8)
memory usage: 129.7+ KB
```

#### Checking the null values

```
In [8]: ► df.isnull().sum()
```

```
Out[8]: Unnamed: 0      0
Date                0
Airline             0
Source              0
Destination         0
Departure Time      0
Arrival Time       0
Total Stops        0
Duration            0
Price              0
dtype: int64
```

No null values present in the dataset

Now we have checked the attributes for the dataset and get a rough idea about the dataset like the no of rows & columns, datatype & null values in the dataset.

## Dealing with the Null Values

In the dataset null values are present, so we have to handled them for better model learning. As we have categorical data so we have to handled it accordingly. We will use the most occurring value in the column to fill the null value.



### Checking the null values

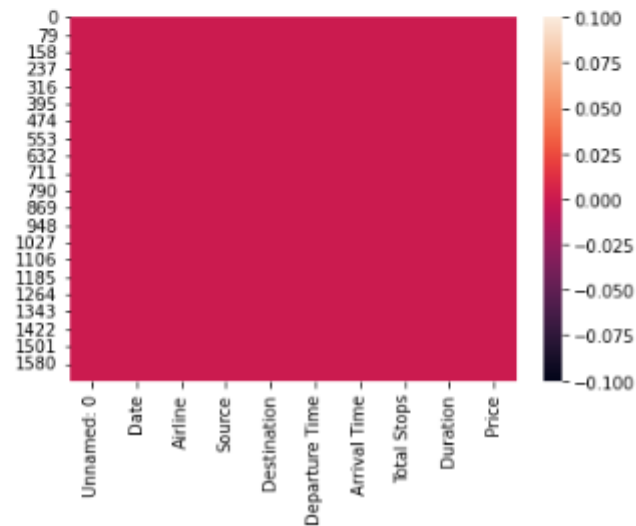
```
In [8]: df.isnull().sum()
```

```
Out[8]: Unnamed: 0      0  
Date      0  
Airline    0  
Source     0  
Destination 0  
Departure Time 0  
Arrival Time 0  
Total Stops 0  
Duration   0  
Price      0  
dtype: int64
```

No null values present in the dataset

```
In [9]: sns.heatmap(df.isnull())
```

```
Out[9]: <AxesSubplot:>
```



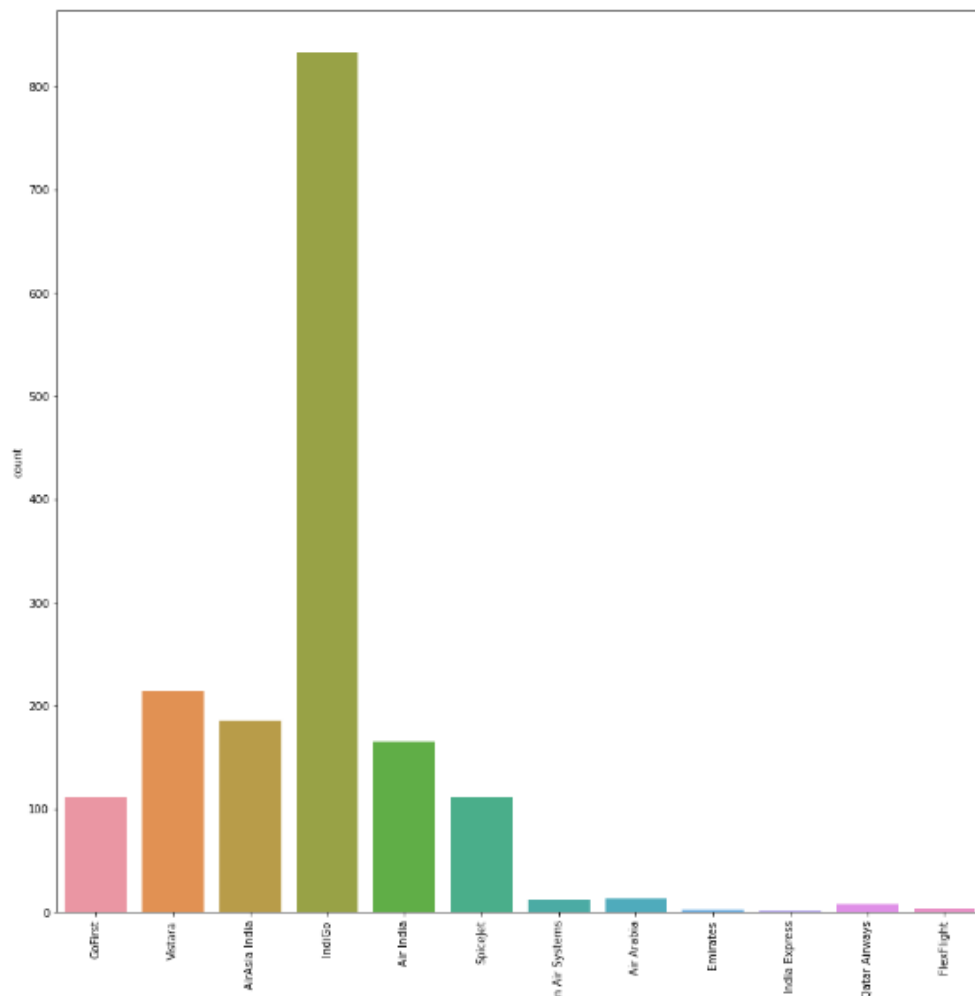
# EXPLORATORY DATA ANALYSIS

## Univariate Analysis

Let's understand each variable one by one and try to interpret about them.

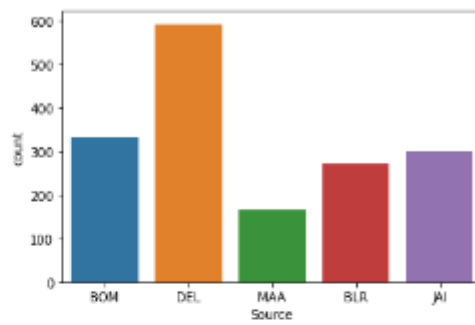
```
In [13]: plt.figure(figsize=(15,15))
sns.countplot(df['Airline'])
plt.xticks(rotation=90)
```

```
Out[13]: (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'GoFirst'),
  Text(1, 0, 'Vistara'),
  Text(2, 0, 'AirAsia India'),
  Text(3, 0, 'IndiGo'),
  Text(4, 0, 'Air India'),
  Text(5, 0, 'SpiceJet'),
  Text(6, 0, 'Hahn Air Systems'),
  Text(7, 0, 'Air Arabia'),
  Text(8, 0, 'Emirates'),
  Text(9, 0, 'Air India Express'),
  Text(10, 0, 'Qatar Airways'),
  Text(11, 0, 'FlexFlight')])
```



```
In [14]: sns.countplot(df['Source'])
```

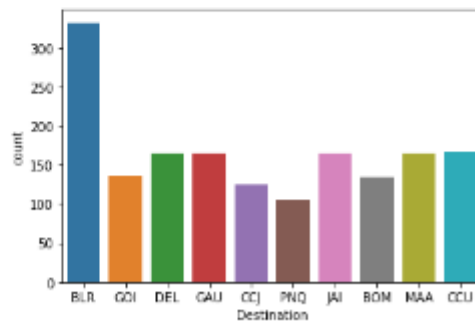
```
Out[14]: <AxesSubplot:xlabel='Source', ylabel='count'>
```



Contains more of the source of flight is Delhi

```
In [15]: sns.countplot(df['Destination'])
```

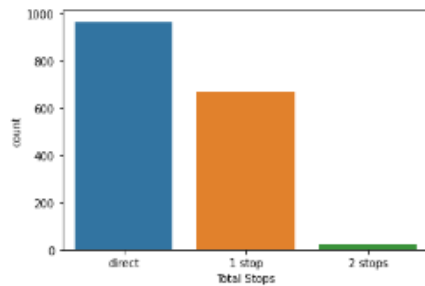
```
Out[15]: <AxesSubplot:xlabel='Destination', ylabel='count'>
```



Favourite Destination is Bengaluru

```
In [17]: sns.countplot(df['Total Stops'])
```

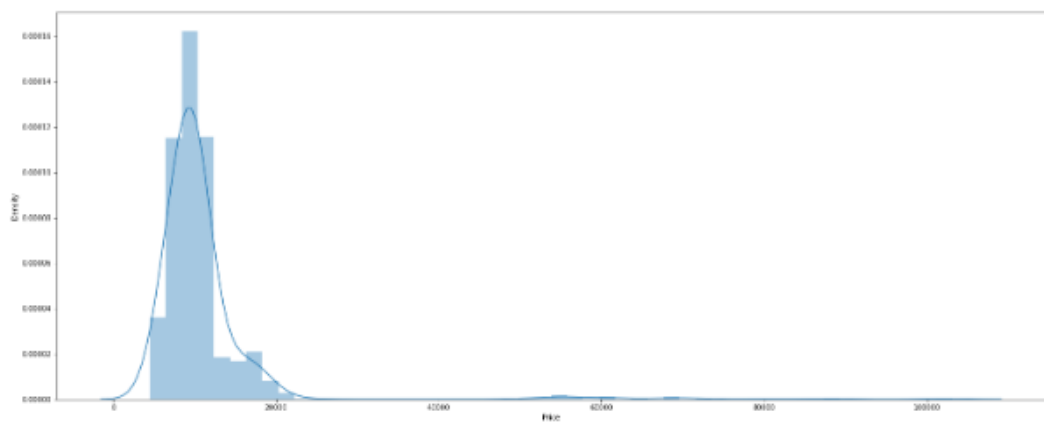
```
Out[17]: <AxesSubplot:xlabel='Total Stops', ylabel='count'>
```



Most of the flights are direct

```
In [11]: plt.figure(figsize=(25,10))  
sns.distplot(df['Price'])
```

```
Out[11]: <AxesSubplot:xlabel='Price', ylabel='Density'>
```



We have the price index less than 20000 a very low ratio is towards the higher side

## Observations

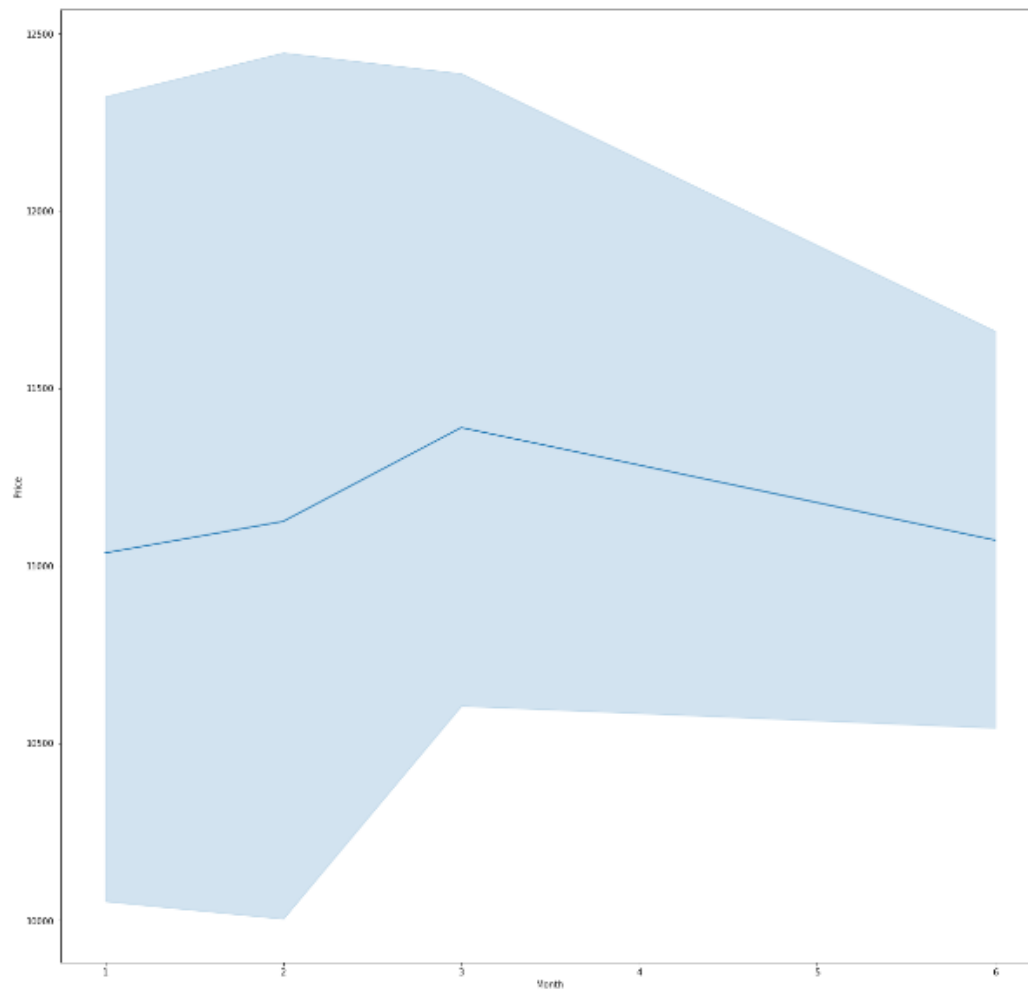
- We have a very large number of flights of IndiGo airline
- Contains more of the source of flight is Delhi
- Favourite Destination is Bengaluru
- Most of the flights are direct
- We have the price index less than 20000 a very low ratio is towards the higher side

## Bivariate Analysis

Let's understand each variable relation with the target variable and interpret how target variable vary with the inputs.

```
In [18]: plt.figure(figsize=(20,20))
sns.lineplot(df['Month'],df['Price'])
```

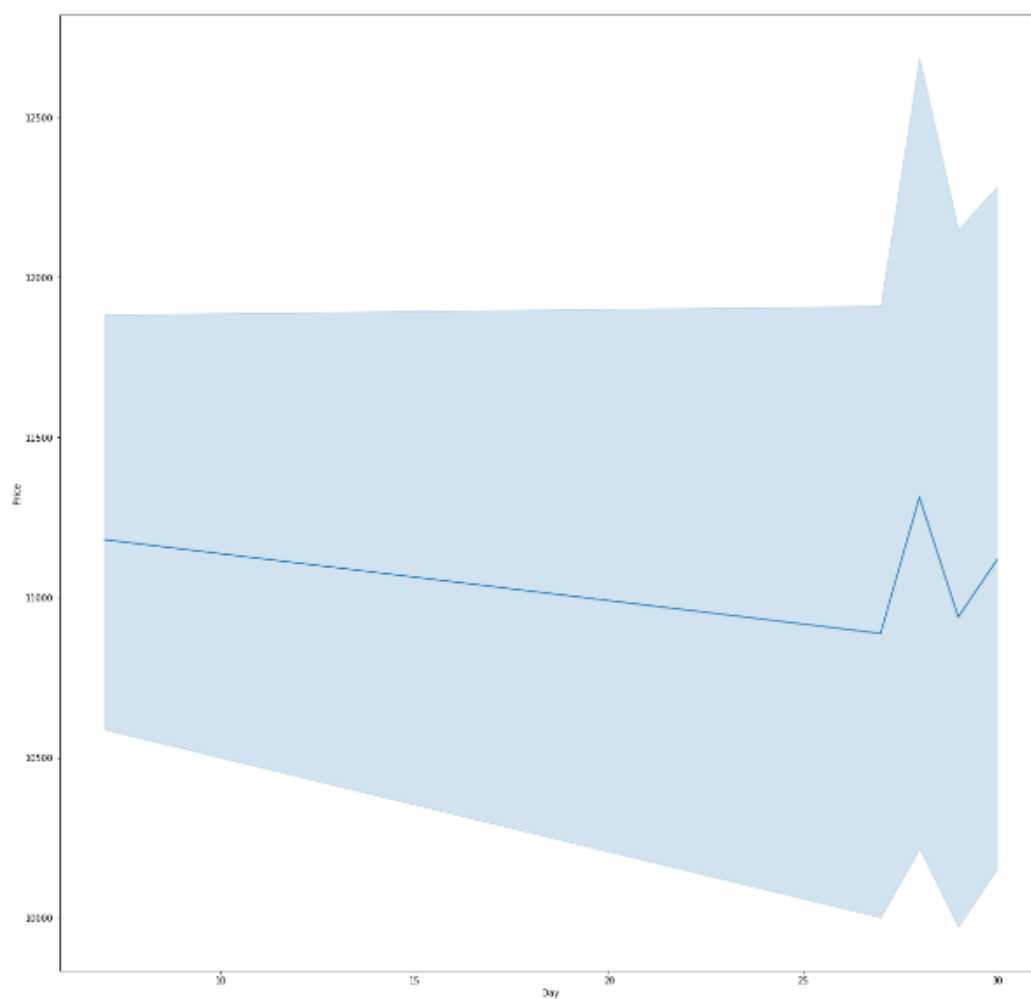
```
Out[18]: <AxesSubplot:xlabel='Month', ylabel='Price'>
```



Prices will increase with the month when we move toward the departure date prices will go up

```
In [19]: plt.figure(figsize=(20,20))
sns.lineplot(df['Day'],df['Price'])
```

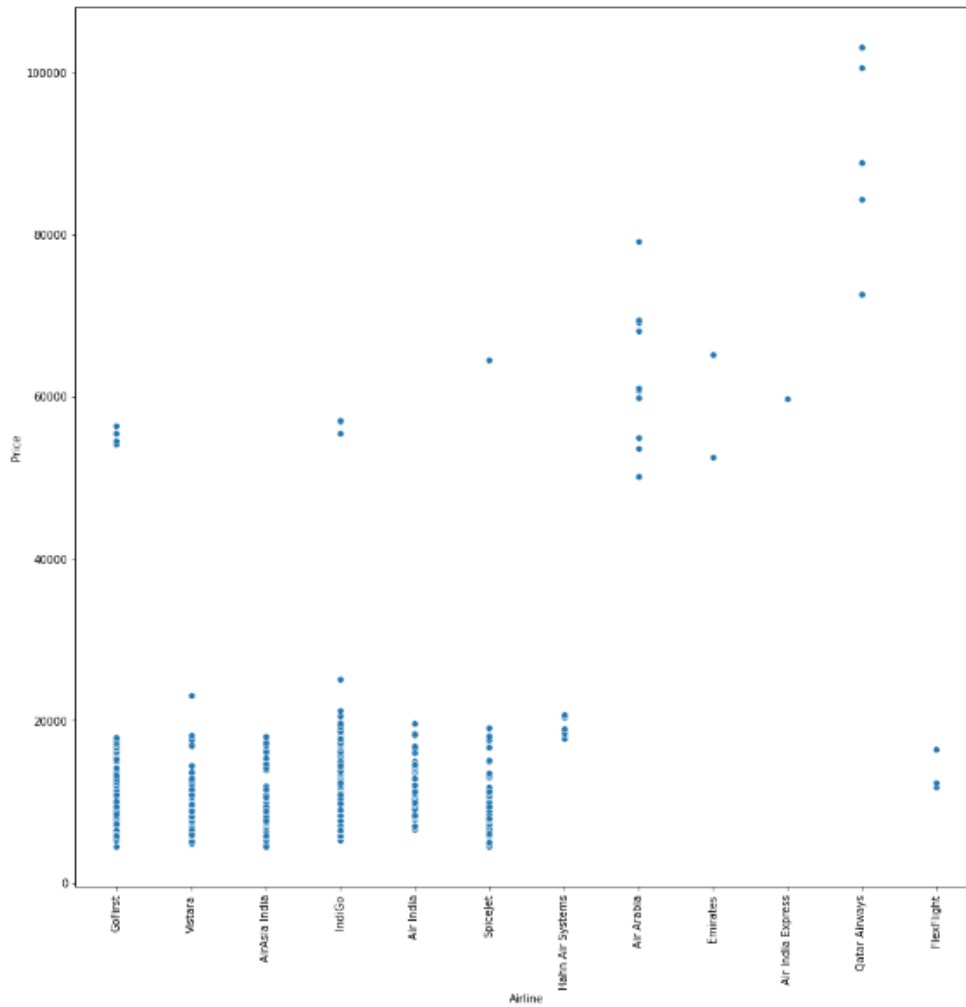
```
Out[19]: <AxesSubplot: xlabel='Day', ylabel='Price'>
```



If we book near the departure date then price will be high and drastically tends to higher side

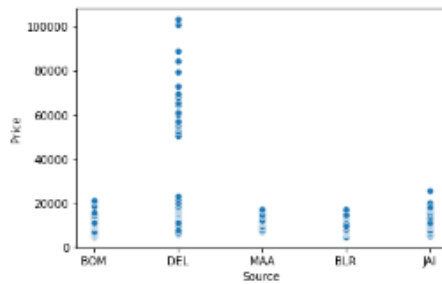
```
In [22]: plt.figure(figsize=(15,15))
sns.scatterplot(df['Airline'],df['Price'])
plt.xticks(rotation=90)
```

```
Out[22]: ([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11],
[Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, ''),
Text(0, 0, '))])
```



```
In [23]: sns.scatterplot(df['Source'],df['Price'])
```

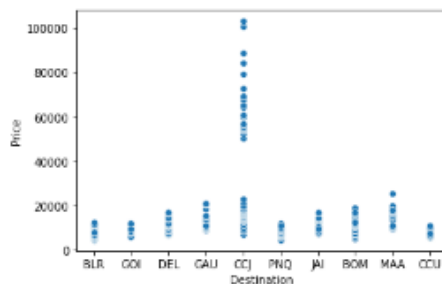
```
Out[23]: <AxesSubplot:xlabel='Source', ylabel='Price'>
```



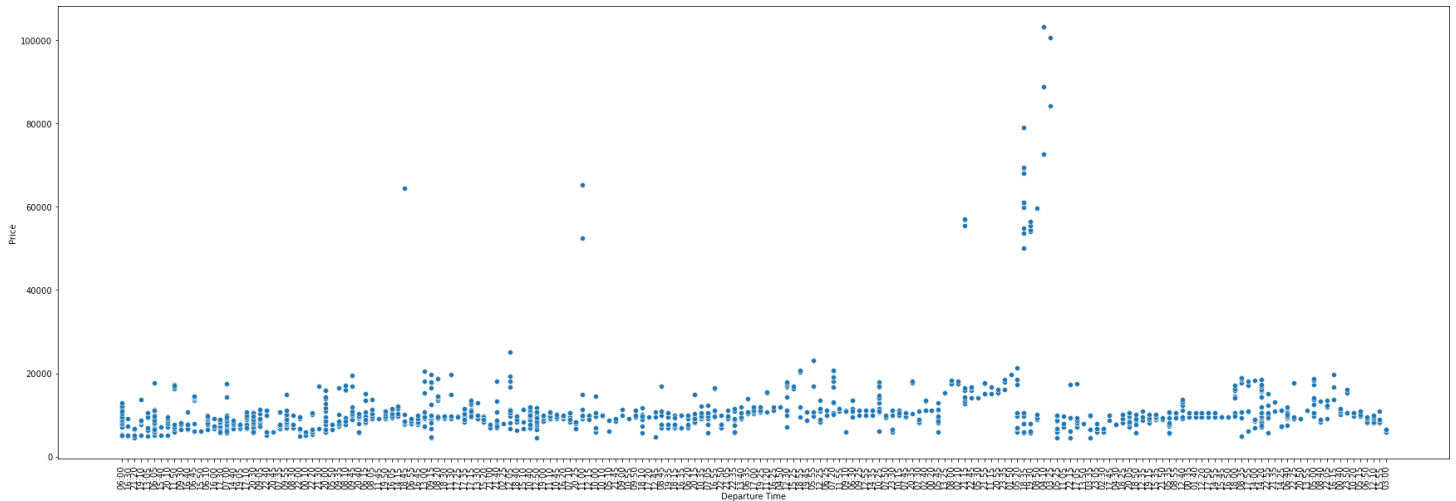
Flights taking off from Delhi Airport are expensive

```
In [24]: sns.scatterplot(df['Destination'],df['Price'])
```

```
Out[24]: <AxesSubplot:xlabel='Destination', ylabel='Price'>
```



Flights to Calicut are expensive



## Observation

- Prices will increase with the month when we move toward the departure date prices will go up
- If we book near the departure date then price will be high and drastically tends to higher side
- 1 stop flights are expensive than others



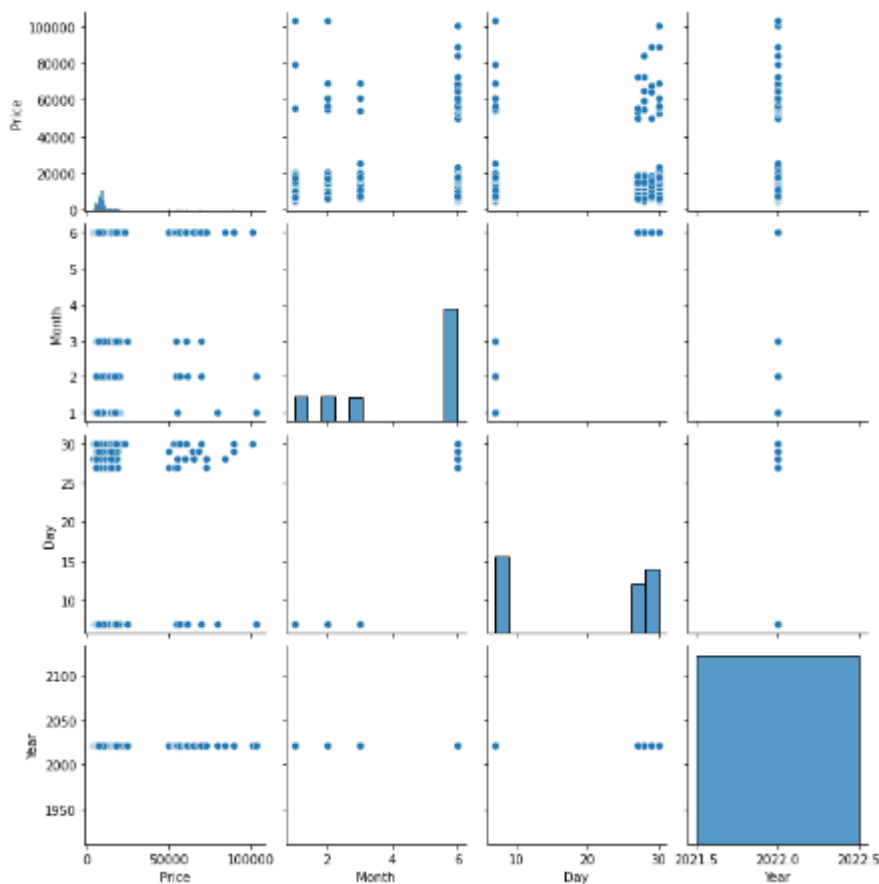
- Qatar airways is the most expensive airline followed by Air Arabia
- Flights taking off from Delhi Airport are expensive
- Flights to Calicut are expensive
- Morning flights are expensive than night ones

## Multivariate Analysis

Using the pairplot function plot each column relation with each other to have a better understanding.

```
In [25]: sns.pairplot(df)
```

```
Out[25]: <seaborn.axisgrid.PairGrid at 0x15e750418b0>
```



## Label Encoding & Correlation

As we have some categorical data we have to encoded those columns for machine learning model. We will use Label Encoder from sklearn.preprocessing.

Find the skewness of each column as well for each column, if it is out of acceptable range then we have to scale the skewness also.

```
In [4]: from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for i in df.columns:
    if df[i].dtypes=="object":
        df[i]=le.fit_transform(df[i].astype(str))
df.head()
```

```
In [13]: df.describe()
```

```
In [14]: corr=df.corr()
```

[illegible]

### Checking the skewness

```
In [18]: df.skew()

Out[18]: Airline      -0.823673
Source        0.055134
Destination   0.095113
Departure Time 0.124377
Arrival Time  -0.045236
Total Stops   -0.359391
Duration      0.728345
Price         6.487139
Month         -0.591159
Day           -0.380465
dtype: float64
```

Very less skewness is present in the dataset except the price column

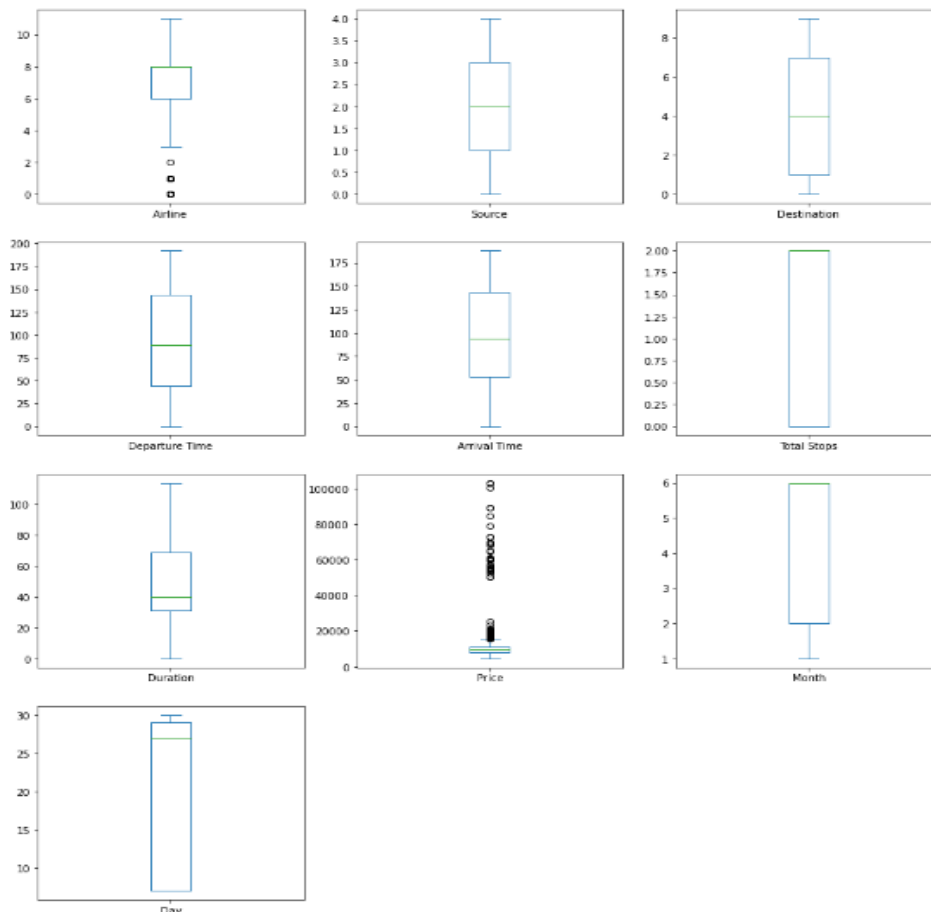
We have encoded our categorical data and find the correlation between each column. As skewness is acceptable so there is no need to scale the skewness.

## Removing the Outliers

### Plotting the outliers

```
In [19]: df.plot(kind='box',subplots=True,layout=(4,3),figsize=(15,18))

Out[19]: Airline      AxesSubplot(0.125,0.71587;0.227941x0.16413)
Source        AxesSubplot(0.398529,0.71587;0.227941x0.16413)
Destination   AxesSubplot(0.672859,0.71587;0.227941x0.16413)
Departure Time AxesSubplot(0.125,0.518913;0.227941x0.16413)
Arrival Time  AxesSubplot(0.398529,0.518913;0.227941x0.16413)
Total Stops   AxesSubplot(0.672859,0.518913;0.227941x0.16413)
Duration      AxesSubplot(0.125,0.321957;0.227941x0.16413)
Price         AxesSubplot(0.398529,0.321957;0.227941x0.16413)
Month         AxesSubplot(0.672859,0.321957;0.227941x0.16413)
Day           AxesSubplot(0.125,0.125;0.227941x0.16413)
dtype: object
```



### Removing the outliers

```
In [20]: from scipy.stats import zscore
z=np.abs(zscore(df))
threshold=3
print(np.where(z>3))
df_new=df[(z<3).all(axis=1)]
df=df_new
df.shape

(array([[889, 890, 891, 892, 893, 894, 895, 896, 897, 898, 899, 900, 901,
        902, 903, 904, 905, 906, 907, 908, 909, 910, 911, 912, 913, 914,
        915, 916, 917, 918, 919, 920, 921], dtype=int64), array([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7], dtype=int64))

Out[20]: (1625, 10)
```

We have some outliers present in the dataset, so let's handle them also. As the outliers in the dataset will affect our ML model. We need to remove all the outliers present in the dataset.

There is something called zscore which indicates how many standard deviations away an element is from the mean. We consider the points as outliers whose zscore is above 3 or less than -3. So we need to remove all such points from our dataset.

Using the threshold, we have removed all the points where the zscore is greater than 3. Now the total number of rows after removing the outliers are 1625.

## MODEL BUILDING

We will import important libraries for the building the ML model and defining the different models for our easiness.

Finding the best random state for the train test split.

### Model Building

```
In [22]: df=df[['Airline', 'Source', 'Destination', 'Departure Time', 'Arrival Time',  
            'Duration', 'Total Stops', 'Month', 'Day','Price']]
```

```
In [23]: x=df.iloc[:, :-1]  
        y=df.iloc[:, -1]
```

```
In [24]: #Importing the different machine Learning models  
  
from sklearn.linear_model import LinearRegression  
from sklearn.metrics import mean_squared_error, mean_absolute_error  
from sklearn.model_selection import train_test_split  
from sklearn.ensemble import RandomForestRegressor  
from sklearn.svm import SVR  
from sklearn.tree import DecisionTreeRegressor  
from sklearn.neighbors import KNeighborsRegressor  
from sklearn.metrics import r2_score
```

```
In [25]: # defining the different models  
  
lg=LinearRegression()  
rdr=RandomForestRegressor()  
svr=SVR()  
dtr=DecisionTreeRegressor()  
knr=KNeighborsRegressor()
```

### Finding the best random state

```
In [26]: model=[lg,rdr,svr,dtr,knr]  
maxAcc=0  
maxRS=0  
for i in range(40,60):  
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=.20)  
    lg.fit(x_train,y_train)  
    pred=lg.predict(x_test)  
    acc=r2_score(y_test,pred)  
    if acc>maxAcc:  
        maxAcc=acc  
        maxRS=i  
print('Best Accuracy score is', maxAcc , 'on random state', maxRS)  
  
Best Accuracy score is 0.3999963569827626 on random state 49
```

```
In [27]: x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=49,test_size=.20)
```

## Regression Algorithms

We have use five different regression algorithms to find the best model for our problem.

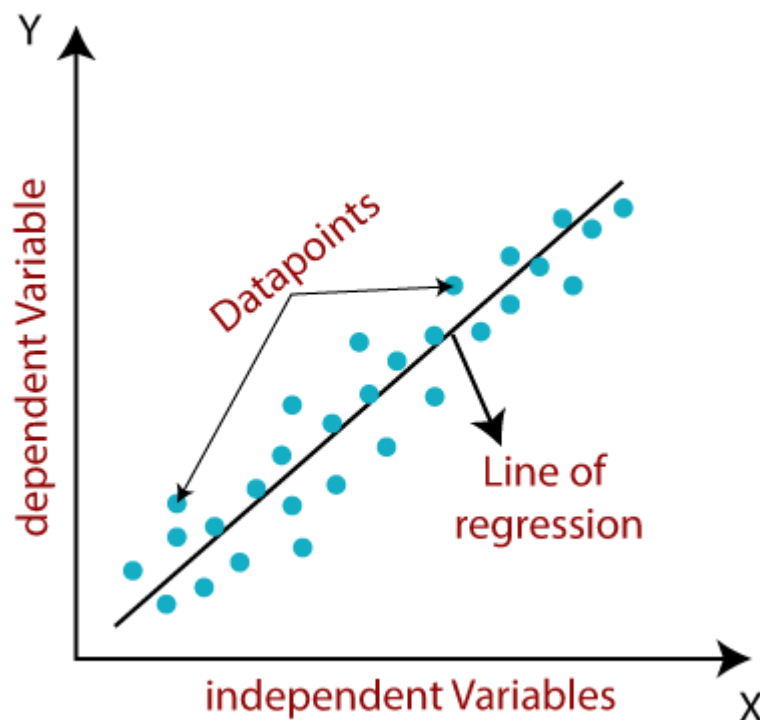
### 1. Linear Regression

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

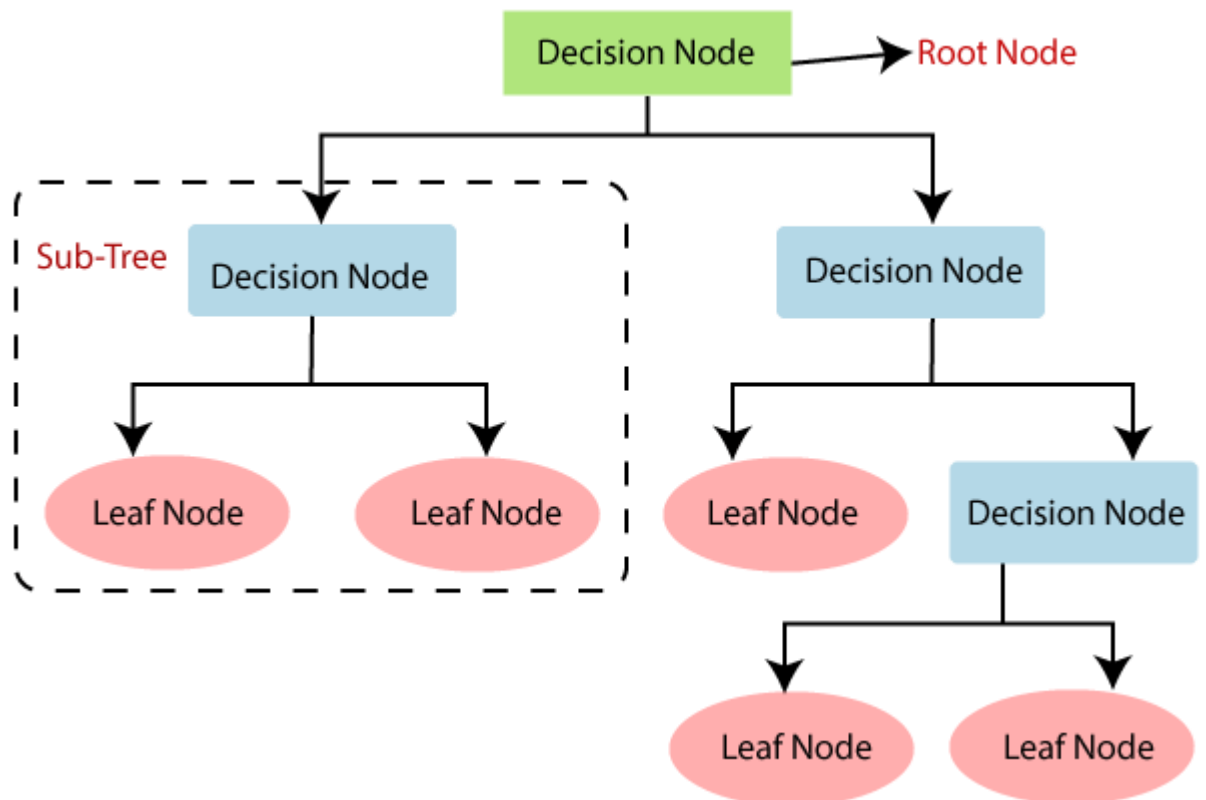
regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

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



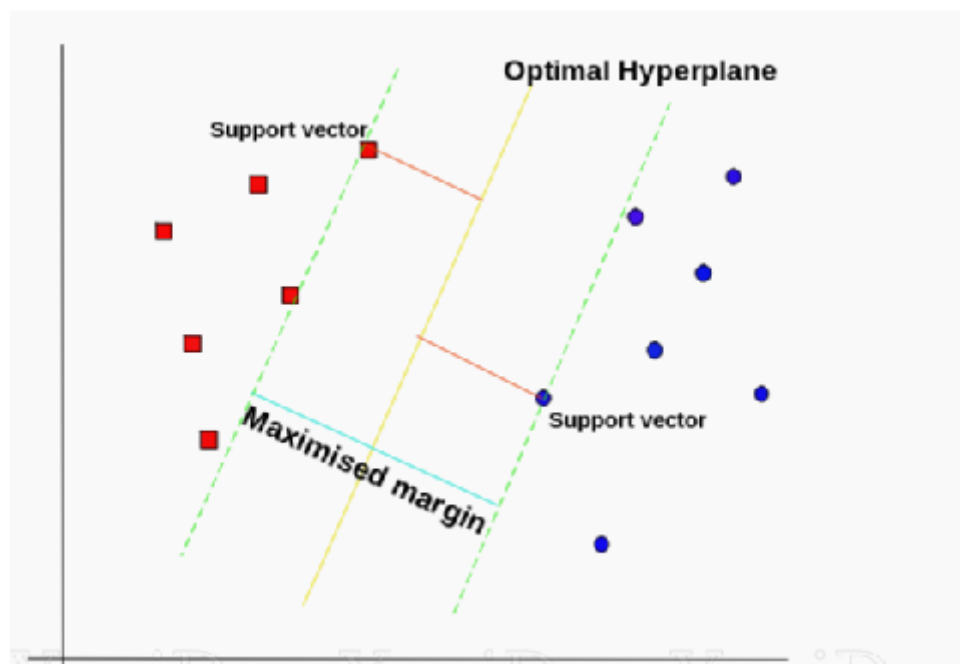
## 2. Decision Tree Regressor

Decision Tree can be used both in classification and regression problem. Decision trees are predictive models that use a set of binary rules to calculate a target value. Each individual tree is a fairly simple model that has branches, nodes and leaves. 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.



### 3. Support Vector Regressor

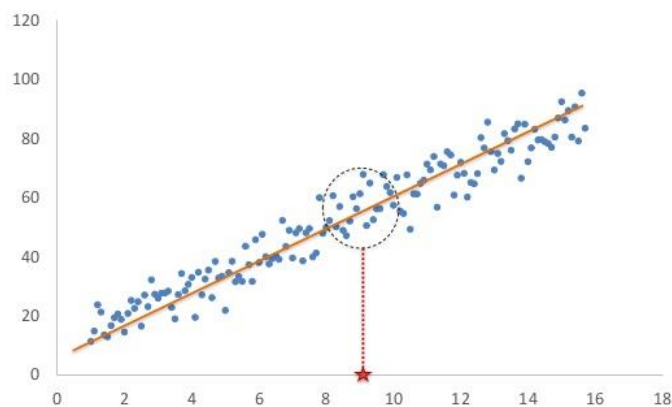
Support Vector Regression is a supervised learning algorithm that is used to predict discrete values. Support Vector Regression uses the same principle as the SVMs. The basic idea behind SVR is to find the best fit line. In SVR, the best fit line is the hyperplane that has the maximum number of points.



#### 4. K Neighbor Regression

KNN regression is a non-parametric method that, in an intuitive manner, approximates the association between independent variables and the continuous outcome by averaging the observations in the same *neighbourhood*. The size of the neighbourhood needs to be set by the analyst or can be chosen using cross-validation (we will see this later) to select the size that minimises the mean-squared error.

#### kNN Regression

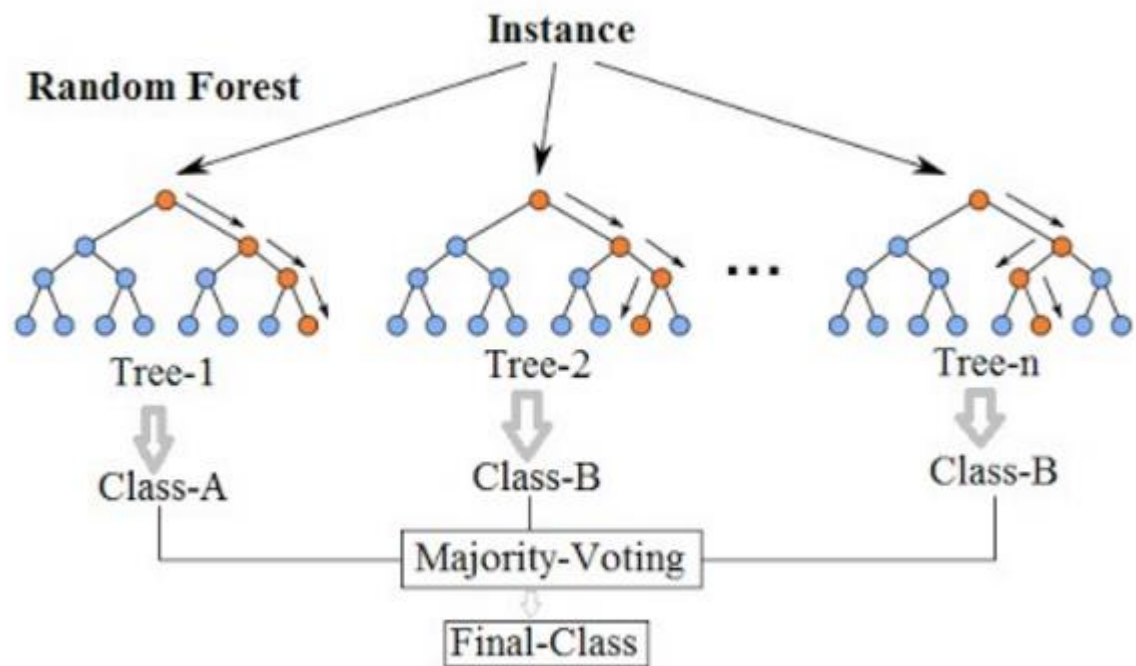


#### 5. Random Forest Regression

A random forest is a machine learning technique that's used to solve regression and classification problems. It utilizes ensemble learning, which is a technique that combines many classifiers to provide solutions to complex problems. Random Forest works in two-phase first is to create the random forest by combining N decision tree, and second is to make predictions for each tree created in the first phase.



## Random Forest Simplified



## 6. Lasso Regularisation

It modifies the over-fitted or under-fitted models by adding the penalty equivalent to the sum of the absolute values of coefficients. Lasso regression also performs coefficient minimization, but instead of squaring the magnitudes of the coefficients, it takes the true values of coefficients. This means that the coefficient sum can also be 0, because of the presence of negative coefficients.

Consider the cost function for the Lasso:

$$\text{Cost function} = \text{Loss} + \lambda \times \sum ||w||$$

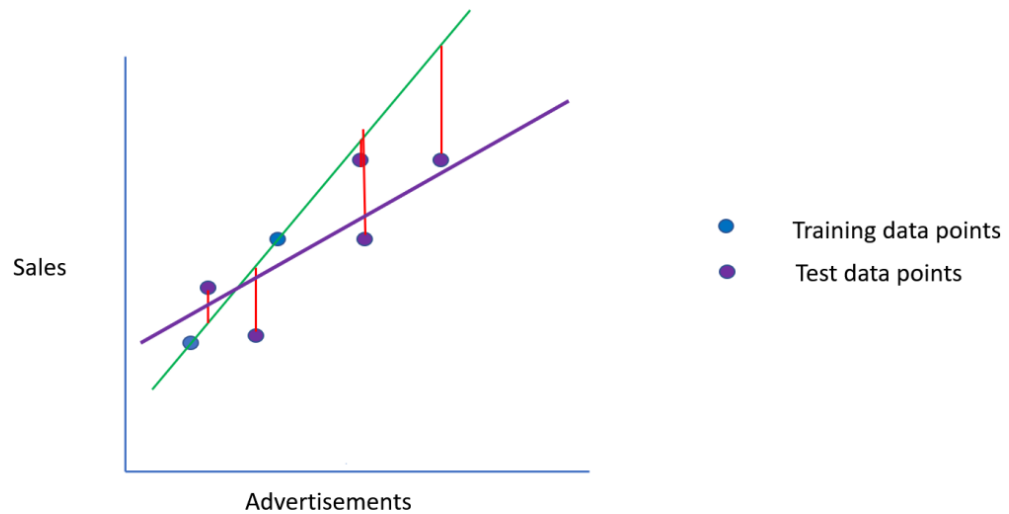
Where Loss=sum of squared residuals

$\lambda$  = Penalty error

$w$  = slope of curve/line

We can control the coefficient values by controlling the penalty terms. LASSO regression converts coefficients of less important features to zero, which indeed helps in feature selection, and it shrinks the coefficients of remaining features to reduce the model complexity, hence avoiding overfitting.

# Lasso Regression



## 7. Ridge Regularisation

It modifies the over-fitted or under fitted models by adding the penalty equivalent to the sum of the squares of the magnitude of coefficients. This means that the mathematical function representing our machine learning model is minimized and coefficients are calculated. The magnitude of coefficients is squared and added. Ridge Regression performs regularization by shrinking the coefficients present.

The Cost function for Ridge:

$$\text{Cost function} = \text{Loss} + \lambda \times \sum ||w||^2$$

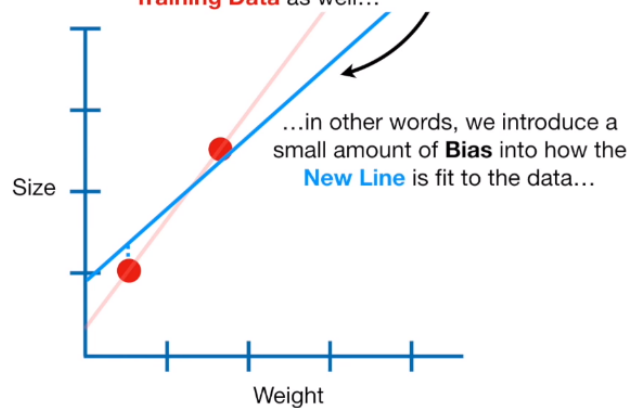
Where Loss=sum of squared residuals

$\lambda$  = Penalty error

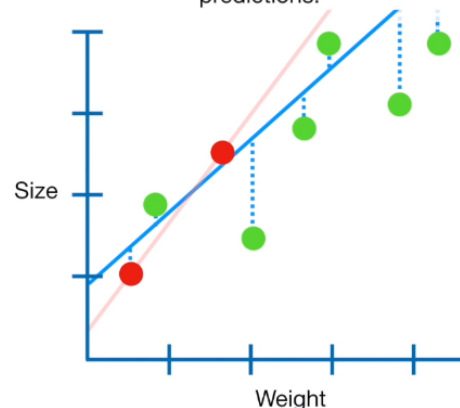
w= slope of curve/line

Ridge regression shrinks the coefficients as it helps to reduce the model complexity and multi collinearity.

The main idea behind **Ridge Regression** is to find a **New Line** that doesn't fit the **Training Data** as well...



In other words, by starting with a slightly worse fit, **Ridge Regression** can provide better long term predictions.



### Libraries Used for Regression Models

- Linear Regression
  - from sklearn.linear\_model import LinearRegression
- Decision Tree Regressor
  - from sklearn.tree import DecisionTreeRegressor
- Support Vector Regressor
  - from sklearn.svm import SVR
- Kneighbor Regressor
  - from sklearn.neighbors import KNeighborsRegressor
- Random Forest Regressor
  - from sklearn.ensemble import RandomForestRegressor
- Lasso Regularization
  - from sklearn.linear\_model import Lasso
- Ridge Regularization
  - from sklearn.linear\_model import Ridge

### Model Accuracy:

MODEL	ACCURACY
Linear Regression	0.3999963569827626
Decision Tree Regressor	0.7719937778616296
Support Vector Regressor	-0.01887564129226993
Kneighbor Regressor	0.5116525830272465
Random Forest Regressor	0.8644094305461366
Lasso	0.3999359120075674
Ridge	0.399907003614267

#### Linear Regression

```
In [28]: > lg.fit(x_train,y_train)
pred1=lg.predict(x_test)
acc=r2_score(y_test,pred1)
print("Accuracy Score: ",acc)

Accuracy Score: 0.3999963569827626
```

#### Decision Tree Regressor

```
In [29]: > dtr.fit(x_train,y_train)
pred2=dtr.predict(x_test)
acc=r2_score(y_test,pred2)
print("Accuracy Score: ",acc)

Accuracy Score: 0.7719937778616296
```

#### Support Vector Regressor

```
In [30]: > svr.fit(x_train,y_train)
pred3=svr.predict(x_test)
acc=r2_score(y_test,pred3)
print("Accuracy Score: ",acc)

Accuracy Score: -0.01887564129226993
```

#### KNN Regression

```
In [31]: > knr.fit(x_train,y_train)
pred4=knr.predict(x_test)
acc=r2_score(y_test,pred4)
print("Accuracy Score: ",acc)

Accuracy Score: 0.5116525830272465
```

#### Random Forest Regression

```
In [32]: > rdr.fit(x_train,y_train)
pred5=rdr.predict(x_test)
acc=r2_score(y_test,pred5)
print("Accuracy Score: ",acc)

Accuracy Score: 0.8644094305461366
```

#### Lasso Regularization

```
In [33]: > from sklearn.linear_model import Lasso
ls=Lasso()
ls.fit(x_train,y_train)
pred6=ls.predict(x_test)
acc=r2_score(y_test,pred6)
print("Accuracy Score: ",acc)

Accuracy Score: 0.3999359120075674
```

#### Ridge Regularization

```
In [34]: > from sklearn.linear_model import Ridge
rg=Ridge()
rg.fit(x_train,y_train)
pred7=rg.predict(x_test)
acc=r2_score(y_test,pred7)
print("Accuracy Score: ",acc)

Accuracy Score: 0.399907003614267
```

Hence, we are getting the best accuracy score through the Random Forest Classifier Model. We will go ahead with this to find the cross val score and hypermeter tuning.

## Cross Val Score & Hypermeter Tuning

Cross-validation provides information about how well a classifier generalizes, specifically the range of expected errors of the classifier. Cross Val Score tells how the model is generalized at a particular cross validation.

At CV=6 we get the best results i.e. the Random Forest Classifier more generalized at cv=6, so we calculate the hyper parameters at this value.

We will find which parameters of random forest classifier are the best for our model. We will do this using Grid Search CV method & also calculate the accuracy score at those best parameters.

### Cross Val Score

```
In [36]: from sklearn.model_selection import cross_val_score
for i in range(3,7):
    cr=cross_val_score(dtr,x,y,cv=i)
    cr_mean=cr.mean()
    print("at cv= ", i)
    print('cross val score = ',cr_mean*100)

at cv= 3
cross val score = -195.90701276676333
at cv= 4
cross val score = -89.12894509058178
at cv= 5
cross val score = -128.80149369635566
at cv= 6
cross val score = -209.47089388457152
```

### Hypermeter Tuning

```
In [38]: from sklearn.model_selection import GridSearchCV
# creating parameters
param={'criterion':['squared_error','absolute_error','poisson'],
       'max_features':['sqrt','log2'],
       'max_depth':[2,3,4,5]}

GCV=GridSearchCV(rdr,param,cv=6,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

```
Out[38]: {'criterion': 'squared_error', 'max_depth': 2, 'max_features': 'sqrt'}
```

```
In [40]: GCV_pred=GCV.best_estimator_.predict(x_test) #predicting the price using the best parameters of RDR Model
r2_score(y_test,GCV_pred)
```

```
Out[40]: 0.4221434482352
```

```
In [41]: #FOR DTR

from sklearn.model_selection import GridSearchCV
# creating parameters
param={'criterion':['squared_error','absolute_error','poisson','friedman_mse'],
       'max_features':['sqrt','log2','auto'],
       'max_depth':[2,3,4,5],
       'splitter':['best','random']}

GCV=GridSearchCV(dtr,param,cv=6,scoring='accuracy')
GCV.fit(x_train,y_train)
GCV.best_params_
```

```
Out[41]: {'criterion': 'squared_error',
          'max_depth': 2,
          'max_features': 'sqrt',
          'splitter': 'best'}
```

```
In [42]: GCV_pred=GCV.best_estimator_.predict(x_test) #predicting the price using the best parameters of DTR Model
r2_score(y_test,GCV_pred)
```

```
Out[42]: 0.3451241549083278
```

Just for double check let's find out the accuracy of Decision Tree model for its best parameters. But we are getting the higher accuracy for Random Forest Model, so we will go ahead with Random Forest Model and save it for our future predictions.

## Saving the Model

Saving the best model – Random Forest Regression in this case for future predictions. Let's see what are the actual test data and what our model predicts.

### Saving the model

```
In [44]: import pickle
          filename='flight_price_model.pkl'
          pickle.dump(rdr, open(filename, 'wb'))
```

### Conclusion

```
In [45]: a=np.array(y_test)
          pred=np.array(pred5)
          Flight_Price=pd.DataFrame({'Actual':a, 'Predicted':pred})
          Flight_Price
```

```
Out[45]:
```

	Actual	Predicted
0	11076	11076.00
1	7964	8181.72
2	16686	16992.98
3	7984	8510.42
4	10695	10490.31
...	...	...
320	9077	10249.75
321	7332	8432.69
322	10495	10689.36
323	10695	10596.91
324	9840	10102.58

325 rows × 2 columns

Hence, we see our model predicting the values of price more or less near to actual value

Hence up to some good extensions our model predicted so well

# CONCLUSION

## Conclusion of the Study

The results of this study suggest following outputs which might be useful for the client to evaluate the price on the basis of new data:

- There are lot of things that is going to decide the price of a flight. As we see above in our visualizations, a lot of things affect the price like departure date, your destination, your starting point & many more. One needs to analyse every aspect to have good hands on the prediction of the price.
- With the machine learning it become easier to predict the price but yes it is not 100% accurate, it provides an idea and accordingly we can analyse when one should buy the ticket at a cheaper rate.
- Learning Outcomes of the Study in respect of Data Science
  - As our first step to collect the fresh or new data for the project in accordance with today's trend, scraping of data is not an easy task but yes every difficulty teaches something & I learn so much exceptions handling and error handling during the first phase of project.
  - Second phase is to create the machine learning model to help the customer to get the cheaper flight in this fluctuating rates, which is an interesting part. Data pre-processing, null values handling, data cleaning and data engineering handled in a different way every time which is a good learning part.
  - Data visualizations is more interesting as it provides the better understanding of data. We visualize the data to find the answer of such questions like how the target variable varies with other independent variables.