FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

Deepak kr. Singh

ACKNOWLEDGMENT

• I would like to thank FlipRobo Technologies for providing me this opportunity and guidance throughout the project and all the steps that are implemented.

• I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on Flight price prediction project.

A huge thanks to my academic team “Data trained” who helped me learn.

• I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible.

• I would like to thank my project SME M/S Sapna Verma for providing the flexibility in time and also for giving us guidance in creating the project.

• I have referred to various articles in Towards Data Science and Kaggle

INTRODUCTION

• Business Problem Framing

As we all know tourism industry is changing fast and this is attracting a lot more travellers each year. The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Now-a-days flight prices are quite unpredictable. The ticket prices change frequently. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible.

Using technology it is actually possible to reduce the uncertainty of flight prices. So here we will be predicting the flight prices using efficient machine learning techniques. When booking a flight, travellers need to be confident that they're getting a good deal. The Flight Price Analysis API uses an Artificial Intelligence algorithm trained on Amadeus historical flight booking data to show how current flight prices compare to historical fares. More precisely, it shows how a current flight price sits on a distribution of historical airfare prices.

As retrieving price metrics through aggregation techniques and business intelligence tools alone could lead to incorrect conclusions – for example, in cases where insufficient data points to compute specific price statistics have – we used machine learning to forecast prices. This provides an elegant way to interpolate missing data and predict coherent prices. Moreover, we confirmed the forecast decisions using state of the art Explainable AI techniques.

• Conceptual Background of the Domain Problem

- As domestic air travel is getting more and more popular these days in India with various air ticket booking channels coming up online, travellers are trying to understand how these airline companies make decisions regarding ticket prices over time. Nowadays, airline corporations are using complex strategies and methods to assign airfare prices in a dynamic fashion. Flight prices are something unpredictable. It's more than likely that we spent hours on the internet researching flight deals, trying to figure an airfare pricing system that seems completely random every day.
  Flight price appears to fluctuate without reason and longer flights aren't always more expensive than shorter ones. But now the question is how to know proper Flight price, for that I have built a Machine learning model which can predict the Flight price. Using various features like Airline, Source, Destination, Arrival time, Departure time, Stops, Travelling date and the Price for the same travel. So using all these previously known information and analysing the data I have achieved a good model that has 82% accuracy. So let's understand what all the steps we did to reach this good accuracy.
  Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

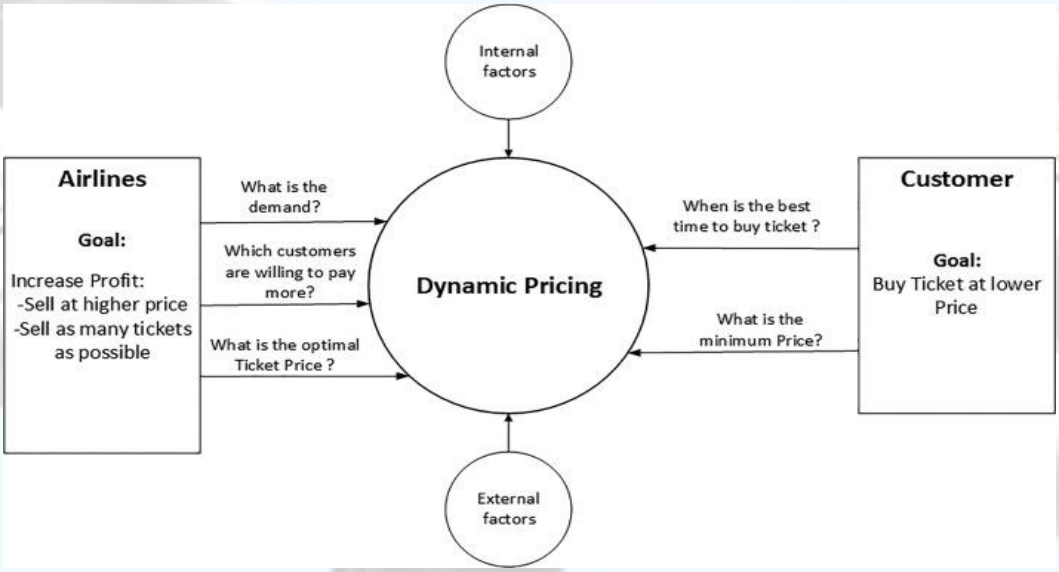
- # Review of Literature

Nowadays, airline corporations are using complex strategies and methods to assign airfare prices in a dynamic fashion. These strategies are taking into consideration several financial, marketing, commercial and social factors are closely connected with the ultimate airfare prices. Due to the high complexity of the pricing models applied by the airlines, it is very difficult for a customer to purchase an air ticket at the lowest price, since the price changes dynamically. For this reason, several techniques ready to provide the proper time to the customer to buy an air ticket by predicting the airfare price, are proposed recently.

To buy an air ticket at the most reduced cost. For this few procedures are explored to determine time and date to grab air tickets with minimum fare rate. The majority of these systems are utilizing the modern computerized system known as Machine Learning. The model guesses airfare well in advance from the known information. This framework is proposed to change various added value arrangements into included added value arrangement heading which can support to solo gathering estimation.
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.
1. Time of purchase patterns (making sure last-minute purchases are expensive)
2. Keeping the flight as full as they want it

- # Motivation for the Problem Undertaken

Flight Price Prediction project help tourists to find the right flight price based on their needs and also it gives various options and flexibility for travelling. Different features (airline, source, destination, departure and arrival timings, Journey date etc.) helps to understand the flight price variations. Using it airlines also get benefits and required passengers.



Due to the high complexity of the pricing models applied by the airlines, it is very difficult for a customer to purchase an air ticket at the lowest price, since the price changes dynamically.

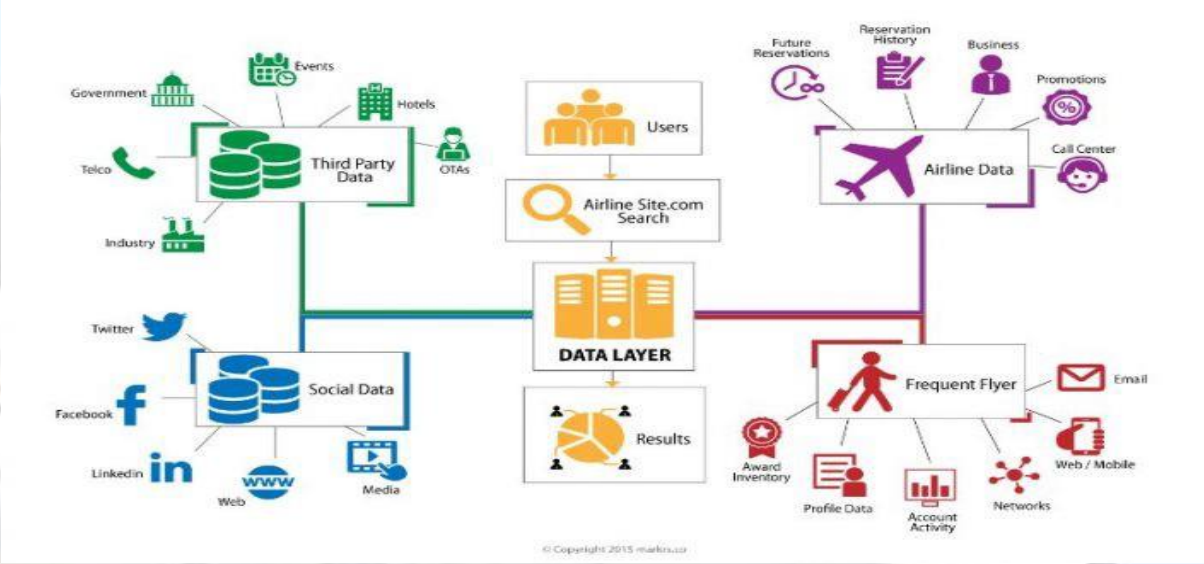
Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

First scrapped the required data from website (makemytrip.com). I have fetched data for different source and destinations and saved it to csv format.
I have Price as my target column and it was a continuous column. So clearly it is a regression problem and I have to use all regression algorithms while building the model. There was no null values in the dataset. Since we have scrapped the data from makemytrip website the raw data was not in the format, so we have use feature engineering to extract the required feature format. To get better insight on the features I have used plotting like distribution plot, bar plot, strip plot and count plot. With these plotting it was able to understand the relation between the features in better manner. skewness or outliers in the dataset not found.
Regression algorithms used while building different model then tuned the best fit model and saved the best model. At last I have predicted the Price using saved model.

- ## Data Sources and their formats

The dataset has been scraped from makemytrip.com website this data is only for academic use, not for any commercial. After scrapping required features the dataset is saved as csv file. Also, my dataset was having 4342 rows and 8 columns including target.



In particular datasets I have object type of data which has been changed as per our analysis
Features Information:

- Airline: The name of the airline.
- Journey_date: The date of the journey
- From: The source from which the service begins.
- To: The destination where the service ends.
- Route: The route taken by the flight to reach the destination.
- D_Time: The time when the journey starts from the source.
- A_Time: Time of arrival at the destination.

• Stops: Total stops between the source and destination.
• Price: The price of the ticket

- # Data Preprocessing Done
    - Firstly, scrapped the required data using selenium from makemytrip website.
    - Imported required libraries and imported the dataset which was in csv format.
    - Then I did all the statistical analysis like checking shape, nunique, value counts, info .
    - While checking for null values I dropped that row as it will not help our analysis.
    - I have also dropped Unnamed:0 column as I found it was the index column of csv.
    - Next as a part of feature extraction I converted the data types of date-time columns and I have extracted useful information from the raw dataset. Thinking that this data will help us more than raw data

- # Data Inputs- Logic- Output Relationships

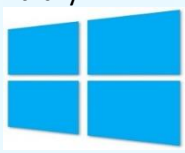
Exploratory Data Analysis (*EDA*)

    - This section shows the exploration done on the dataset, which is what motivated the use of the algorithm. The following are the questions explored in this project and for the sake of writing I will only show some of the visuals here while I will provide the codes that shows the full visualization of all the questions explored.
    - Is there a significant relationship between Flight price and distance travelled? It was used to check for this and we can see that there is a relationship between how much Flight price are and how much is its selling price of flight ticket.
    - There are two primary phases in the system:
    - 1. Training phase: The system is trained by using the data in the data set and fits a model (line/curve) based on the algorithm chosen accordingly.
    - 2. Testing phase: the system is provided with the inputs and is tested for its working. The accuracy is checked. And therefore, the data that is used to train the model or test it, has to be appropriate. The system is designed to detect and predict Flight price and hence appropriate algorithms must be used to do the two different tasks. Before the algorithms are selected for further use, different algorithms were compared for its accuracy. The well-suited one for the task was chosen.
    -
    - Steps :
    - Importing the required packages into our python environment
    - Importing the Flight price prediction data and do some EDA on it
    - Data Visualization on the Flight price prediction data
    - Feature Selection & Data Split
    - Modelling the data using the algorithms
    - Evaluating the built model using the evaluation metrics

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

Finally, we conclude which model is best suitable for the given case by evaluating each of them using the evaluation metrics provided by the scikit-learn package. This model will help to decide the Flight price prediction.

• Hardware and Software Requirements and Tools Used

- Listing down the hardware and software requirements along with the tools, libraries and packages used.
- Windows 10 64bit
- Anaconda 2021 / Python version – Python 3.9.5
- Software: Jupyter notebook, Python, Panda library, numpy library, Matplotlib library, Seaborn library
- 
- **Python**: Python is a general-purpose, and high-level programming language which is best known for its efficiency and powerful functions. Its ease to use, which makes it more accessible. Python provides data scientists with an extensive amount of tools and packages to build machine learning models. One of its special features is that we can build various machine learning with less-code.
- **Matplotlib** is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
- **Seaborn** is a library for making statistical graphics in Python. It builds on top of matplotlib and integrates closely with pandas data structures. Seaborn helps you explore and understand
- **NumPy** is a general-purpose array-processing package. it provides a high-performance multidimensional array object and tools for working with these arrays. It is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
- **Scikit-learn** provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built
- **Jupyter notebook**: The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modelling, data visualization, machine learning.

  The **Jupyter** Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. It includes data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

Importing datasets:

```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime as dt

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

Model/s Development and Evaluation

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

  Identification of possible problem-solving approaches (methods). The factors need to be found which can impact the Flight price. This can be done by analysing the various factors and the stores the respondent prefers. This will be done by checking each of the factors impacts the respondents in decision making.

  Since the data collected was not in the format we have to clean it and bring it to the proper format for our analysis. Since there was no outliers and skewness in the dataset no need to worry about that. We have dropped all the unnecessary columns in the dataset according to our understanding. Use of Pearson's correlation coefficient to check the correlation between dependent and independent features. Also I have used Standardisation to scale the data. After scaling we have to check multicollinearity using VIF. Then followed by model building with all Regression algorithms.

- ## Testing of Identified Approaches (Algorithms)

- We utilized several classic and state-of-the-art methods, including ensemble learning techniques, with a 90% - 10% split for the training and test data. To reduce the time required for training, we used about 5 thousand examples from our dataset. Linear Regression, Random Forest and Gradient Boost were our baseline methods. For most of the model implementations, the open-source Scikit-Learn package was used.
- Since Price was my target and it was a continuous column with improper format which has to be changed to continuous float datatype column, By looking into the r2 score and error values I found ExtraTreesRegressor as a best model with highest r2_score and least error values. Also to get the best model we have to run through multiple.
  - RandomForestRegressor
  - XGBRegressor
  - ExtraTreesRegressor
  - GradientBoostingRegressor
  - DecisionTreeRegressor
  - KNN
  - BaggingRegressor

- ## Run and Evaluate selected models

- **Important package required**

  Our primary packages for this project are going to be pandas for data processing, NumPy to work with arrays, matplotlib & seaborn for data visualizations, and finally scikit-learn for building an evaluating our ML model.

  - RandomForestRegressor
  - XGBRegressor
  - ExtraTreesRegressor
  - GradientBoostingRegressor
  - DecisionTreeRegressor
  - KNN
  - BaggingRegressor

Model building steps:

```
def scrap (origin, destin,trDate):
    global none
    baseDataUrl = "https://www.makemytrip.com/flight/search?itinerary="+ origin +"-"+ destin +"-"+ trDate +"&tripType=O&paxType=
    try:
        driver = webdriver.Chrome(r"C:\driver\chromedriver.exe")
        print("Requested Url: "+baseDataUrl)

        driver.get(baseDataUrl)
        print("WebPage found...")

        element_xpath = '//*[@id="left-side--wrapper"]/div[2]'
```

```
#Importing dataset
df = pd.read_csv("Flight.csv") #Reading csv file
df.head()
```

	Airline	Journey_date	From	To	Dtime	Atime	Stops	Price
0	IndiGo	05-05-2022	Goa	Ahmedabad	01:20	03:00	Non stop	6,471
1	SpiceJet	05-05-2022	Goa	Ahmedabad	07:50	09:30	Non stop	6,471
2	Go First	05-05-2022	Goa	Ahmedabad	21:00	22:30	Non stop	6,471
3	IndiGo	05-05-2022	Goa	Ahmedabad	00:10	07:40	1 stop via Mumbai	6,471
4	IndiGo	05-05-2022	Goa	Ahmedabad	20:15	21:55	Non stop	12,771

Pre-processing and EDA

```
#Checking shape of my dataset
df.shape
```

```
(4342, 8)
```

```
#Checking all column names
df.columns
```

```
Index(['Airline', 'Journey_date', 'From', 'To', 'Dtime', 'Atime', 'Stops',
       'Price'],
      dtype='object')
```

Maximum no. of flight as per company we see that Indigo has maximum flight count

```
#Checking the value counts of Airline
df.Airline.value_counts()
```

```
IndiGo                      1809
Go First                     545
Vistara                      521
SpiceJet                     487
Air India                    444
AirAsia                      329
Multiple Airlines             71
Alliance Air                  35
SpiceJet, IndiGo              30
IndiGo, SpiceJet              19
Star Air                       8
Air India, SpiceJet            8
Air India, AirAsia             6
SpiceJet, Go First             6
Alliance Air, IndiGo           4
IndiGo, Alliance Air           4
Vistara, SpiceJet              4
Go First, SpiceJet             4
SpiceJet, Air India            3
SpiceJet, Alliance Air         2
AirAsia, Air India             1
SpiceJet, Vistara              1
AirAsia, SpiceJet              1
Name: Airline, dtype: int64
```

```
#Checking valuecount of Journey_year column
df.Journey_year.value_counts()
```

```
2022    4342
Name: Journey_year, dtype: int64
```

Checking Z-Score:

```
from scipy.stats import zscore

z_score = zscore(df[['Price','From','To']])
abs_zscore = np.abs(z_score)

filtering_entry = (abs_zscore < 3).all(axis=1)

df = df[filtering_entry]

# the data now seems much better than before.

df.describe()
```

	Airline	From	To	Stops	Price	Journey_day	Dhour	DMin	AHour	AMin
count	4342.000000	4342.000000	4342.000000	4342.000000	4342.000000	4342.0	4342.000000	4342.000000	4342.000000	4342.000000
mean	10.060341	2.543759	2.613772	0.870797	587.989389	5.0	12.672962	27.394058	13.258637	29.094887
std	5.673164	1.581626	1.549706	2.620271	217.019789	0.0	6.299805	17.453587	6.728187	17.707489
min	0.000000	0.000000	0.000000	0.000000	0.000000	5.0	0.000000	0.000000	0.000000	0.000000
25%	8.000000	1.000000	1.000000	0.000000	533.000000	5.0	7.000000	10.000000	8.000000	15.000000
50%	10.000000	3.000000	3.000000	0.000000	666.000000	5.0	14.000000	30.000000	14.000000	30.000000
75%	13.000000	4.000000	4.000000	1.000000	739.000000	5.0	18.000000	40.000000	19.000000	45.000000
max	22.000000	5.000000	5.000000	26.000000	947.000000	5.0	23.000000	55.000000	23.000000	55.000000

Encoding:

Label Encoding:

```
# Separating categorical columns in df_1
cat_col=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        cat_col.append(i)
print(cat_col)
```

```
['Airline', 'From', 'To', 'Stops', 'Price']
```

```
# LabelEncoder

from sklearn.preprocessing import LabelEncoder

num= LabelEncoder()

label_col = ['Airline', 'From', 'To', 'Stops', 'Price']

for x in label_col:
    df[x] = num.fit_transform(df[x].astype(str))

df.head()
```

	Airline	From	To	Stops	Price	Journey_day	Dhour	DMin	AHour	AMin
0	10	2	0	0	807	5	1	20	3	0
1	14	2	0	0	807	5	7	50	9	30
2	8	2	0	0	807	5	21	0	22	30

Scaling the data using standard scaler:

```
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
X = pd.DataFrame(scaler.fit_transform(x), columns=x.columns)
X.head()
```

	Airline	From	To	Stops	Dhour	DMin	AHour	AMin
0	-0.010637	-0.343837	-1.686819	-0.332369	-1.853122	-0.423690	-1.524901	-1.643273
1	0.694518	-0.343837	-1.686819	-0.332369	-0.900602	1.295352	-0.633027	0.051121
2	-0.363215	-0.343837	-1.686819	-0.332369	1.321945	-1.569718	1.299365	0.051121
3	-0.010637	-0.343837	-1.686819	0.049315	-2.011875	-0.996704	-0.930319	0.615918
4	-0.010637	-0.343837	-1.686819	-0.332369	1.163192	-0.710197	1.150719	1.463115

Checking for multicolinearity issue using VIF:

```
from statsmodels.stats.outliers_influence import variance_inflation_factor
vif=pd.DataFrame()
vif["vif_Features"]=[variance_inflation_factor(X.values, i) for i in range(X.shape[1])]
vif["Features"]=X.columns
vif
```

	vif_Features	Features
0	1.042265	Airline
1	1.052243	From
2	1.044459	To
3	1.032483	Stops
4	1.143451	Dhour
5	1.029570	DMin
6	1.151157	AHour
7	1.016721	AMin

Now Building the model as per the data given

Models:

RandomForestRegressor:

```
RFR=RandomForestRegressor()
RFR.fit(X_train,y_train)
pred=RFR.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 29.65750427773456
mean_squared_error: 31591.02110057777
mean_absolute_error: 103.15642464378803
root_mean_squared_error: 177.73863142428482
```

DecisionTreeRegressor:

```
DTR=DecisionTreeRegressor()
DTR.fit(X_train,y_train)
pred=DTR.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 0.1651115977107519
mean_squared_error: 37495.082968742114
mean_absolute_error: 107.16853963381209
root_mean_squared_error: 193.6364711740588
```

KNN:

```
knn=KNN()
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 0.2157904306792776
mean_squared_error: 35219.08171910975
mean_absolute_error: 117.6096699923254
root_mean_squared_error: 187.66747645532445
```

ExtraTreeRegressor:

```
ETR=ExtraTreesRegressor()
ETR.fit(X_train,y_train)
pred=ETR.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 0.26375560809119813
mean_squared_error: 33064.95153627447
mean_absolute_error: 101.90641000621278
root_mean_squared_error: 181.83770658550023
```

Gradient Boosting Regressor:

```
GBR=GradientBoostingRegressor()
GBR.fit(X_train,y_train)
pred=GBR.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 0.3008831437059485
mean_squared_error: 31397.543024570532
mean_absolute_error: 118.02001670451159
root_mean_squared_error: 177.19351857381955
```

Bagging Regressor:

```
BG=BaggingRegressor()
BG.fit(X_train,y_train)
pred=BG.predict(X_test)
print('R2_score:',r2_score(y_test,pred))
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 0.2688236799138286
mean_squared_error: 32837.342944563265
mean_absolute_error: 105.67662633946594
root_mean_squared_error: 181.21076939454582
```

XGB Regressor:

```
XGB=XGBRegressor()
XGB.fit(X_train,y_train)
pred=XGB.predict(X_test)
R2_score = r2_score(y_test,pred)*100
print('R2_score:',R2_score)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print('root_mean_squared_error:',np.sqrt(metrics.mean_squared_error(y_test,pred)))
```

```
R2_score: 27.094742323931786
mean_squared_error: 32741.96500905094
mean_absolute_error: 103.37346166074963
root_mean_squared_error: 180.94740951185497
```

- Key Metrics for success in solving problem under consideration

- Using the sklearn.metrics I have calculated Adjusted R2 squared ,Mean Absolute Error (MAE),Mean Squared Error (MSE),Root Mean Squared Error (RMSE)
- Using Hyper-parameter : model parameters are estimated from data automatically and model hyper-parameters are set manually and are used in processes to help estimate model and Grid search is a basic method for hyper-parameter tuning. It performs an exhaustive search on the hyper-parameter set specified by users.

Hyper Parameter Tuning:

```
#importing necessary libraries
from sklearn.model_selection import GridSearchCV
```

```
parameter = {'max_features':['auto','sqrt','log2'],
             'min_samples_split':[1,2,3,4],
             'n_estimators':[20,40,60,80,100],
             'min_samples_leaf':[1,2,3,4,5],
              'n_jobs':[-2,-1,1,2]}
```

- # Visualizations

Mention all the plots made along with their pictures and what were the inferences and observations obtained:

Data visulation:

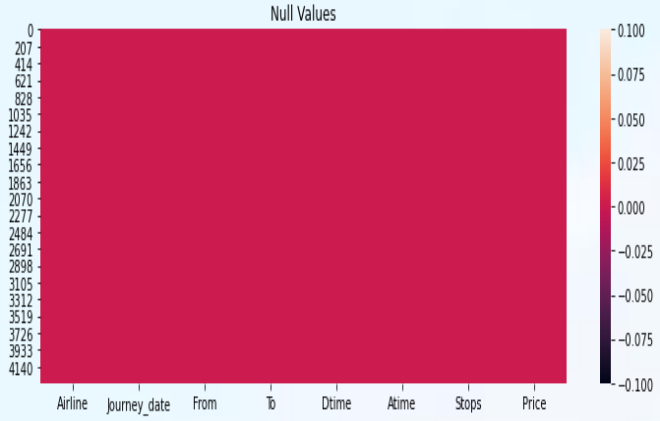
```
# checking for categorical columns
categorical_columns=[]
for i in df.dtypes.index:
    if df.dtypes[i]=='object':
        categorical_columns.append(i)
print(categorical_columns)
```

```
['Airline', 'From', 'To', 'Stops', 'Price']
```

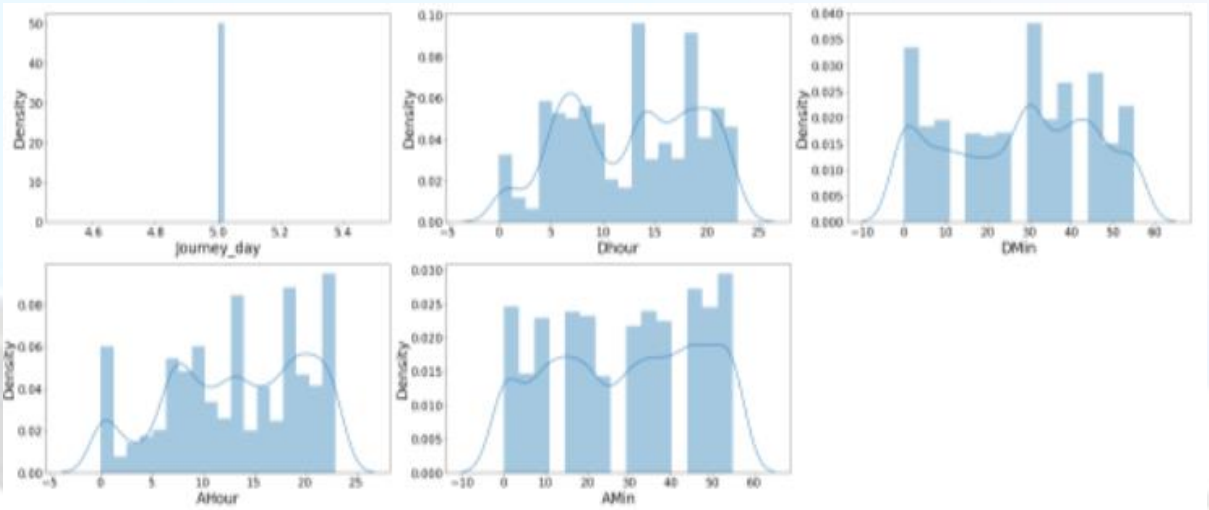
```
# Now checking for numerical columns
numerical_columns=[]
for i in df.dtypes.index:
    if df.dtypes[i]!='object':
        numerical_columns.append(i)
print(numerical_columns)
```

```
['Journey_day', 'Dhour', 'DMin', 'AHour', 'AMin']
```

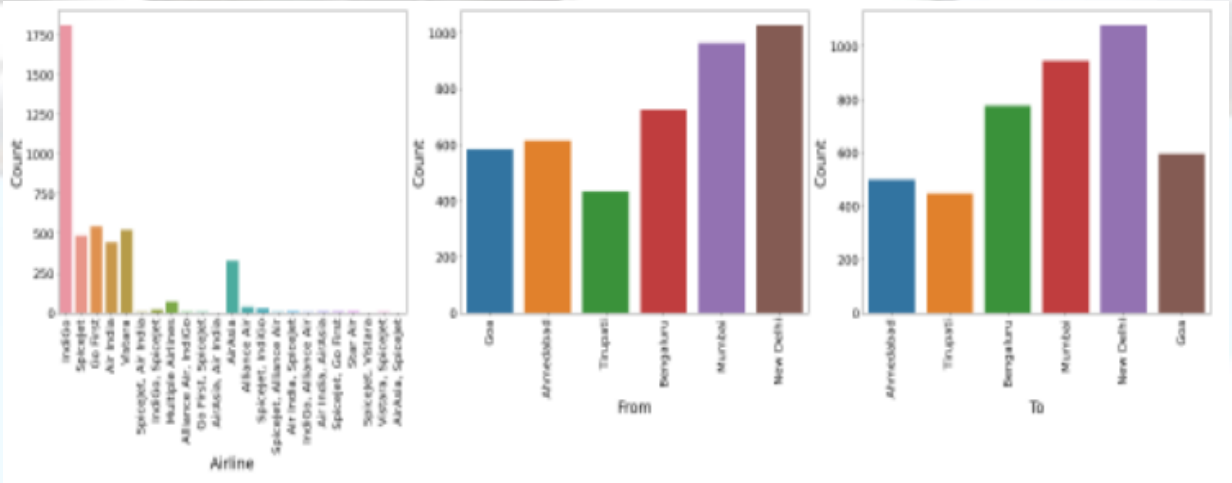
Check null value:



- Univariate analysis for numerical columns:

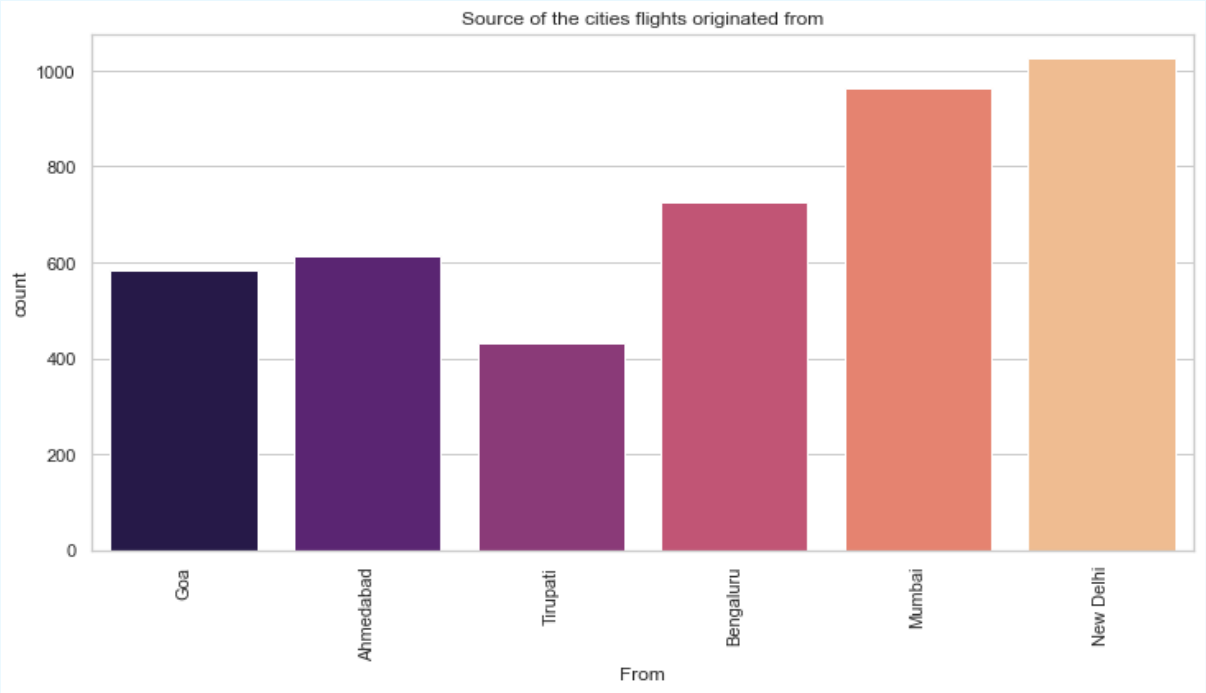


Univariate Analysis for categorical columns:



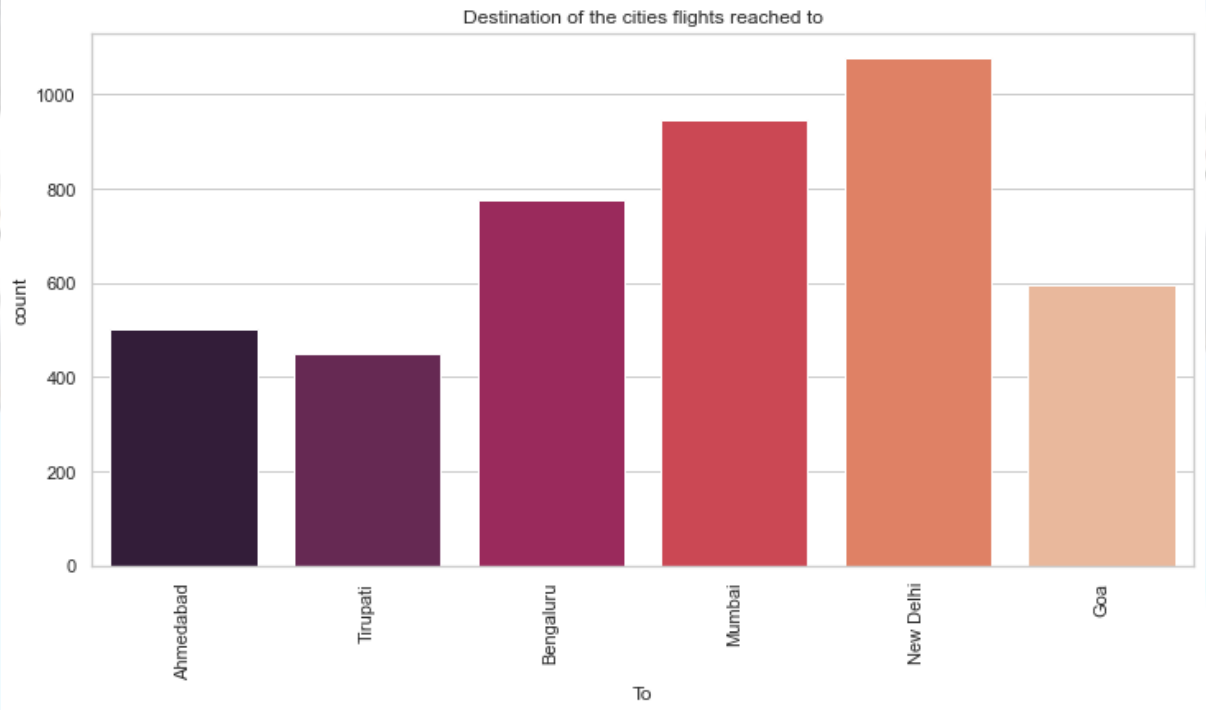
Flights per day Source city:

New Delhi has maximum count for source which means maximum passengers are choosing New Delhi as there source.



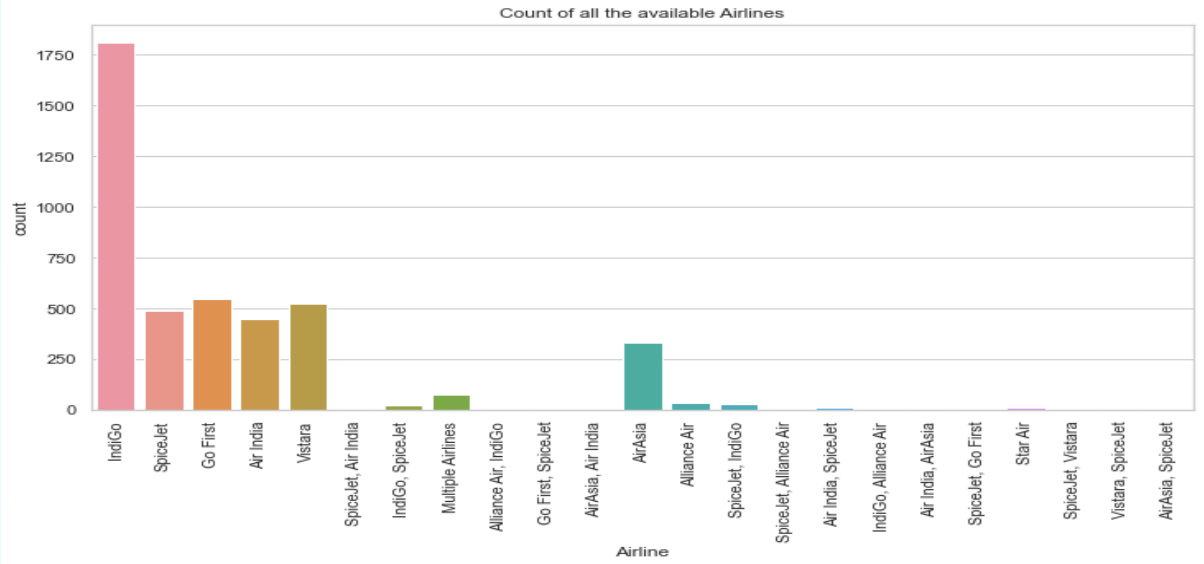
Flights per day Destination city:

New Delhi has maximum count for Destination which means maximum passengers are choosing New Delhi as there Destination.

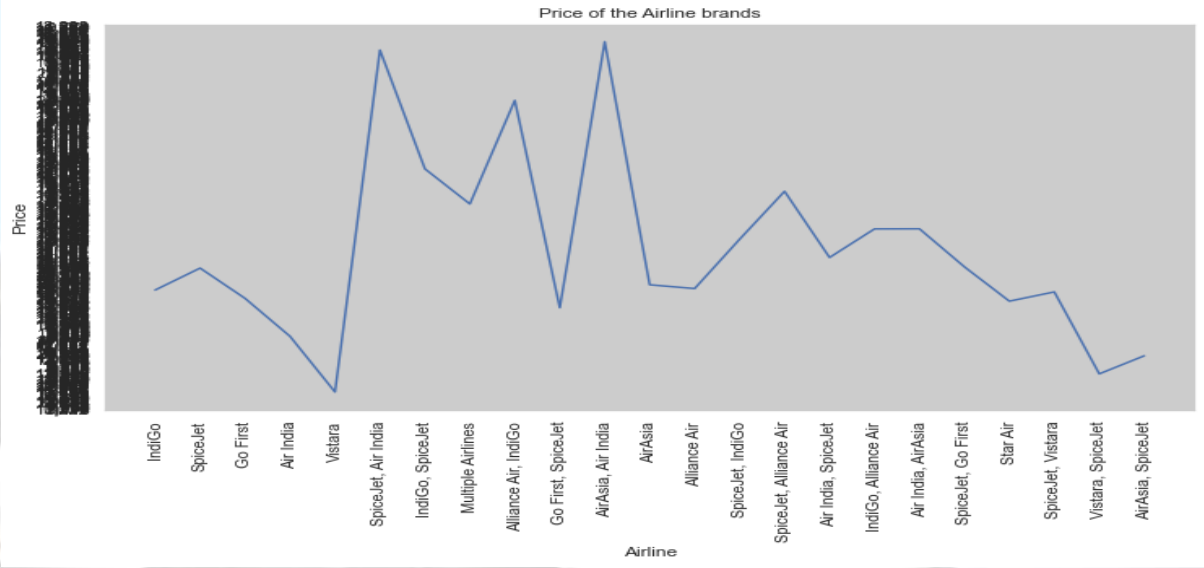


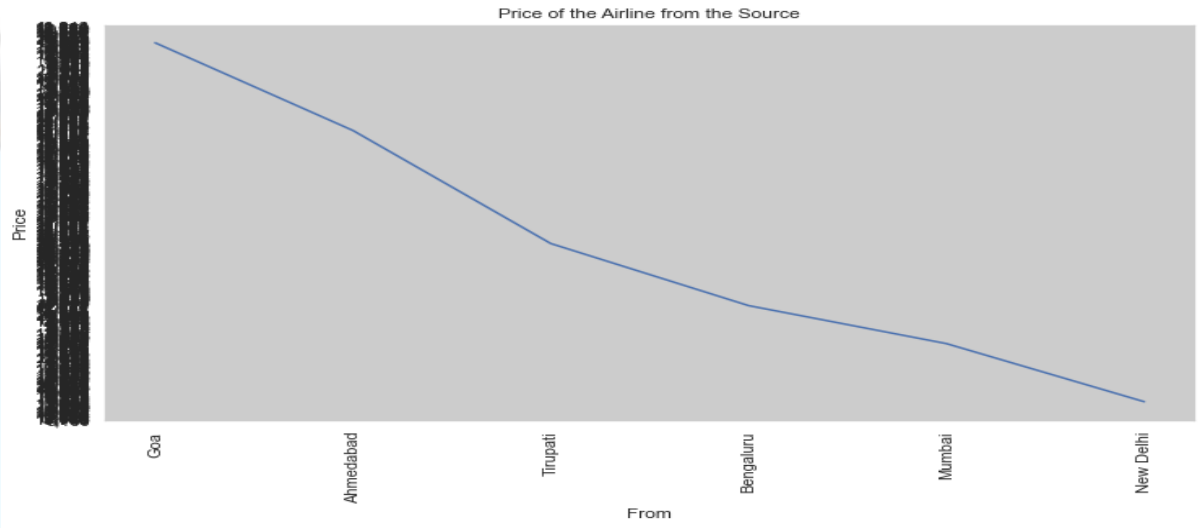
Numbers of airline companies and their flights:

Indigo has maximum count which means most of the passengers preferred Indigo.



Price as per Airline Brand & source city:

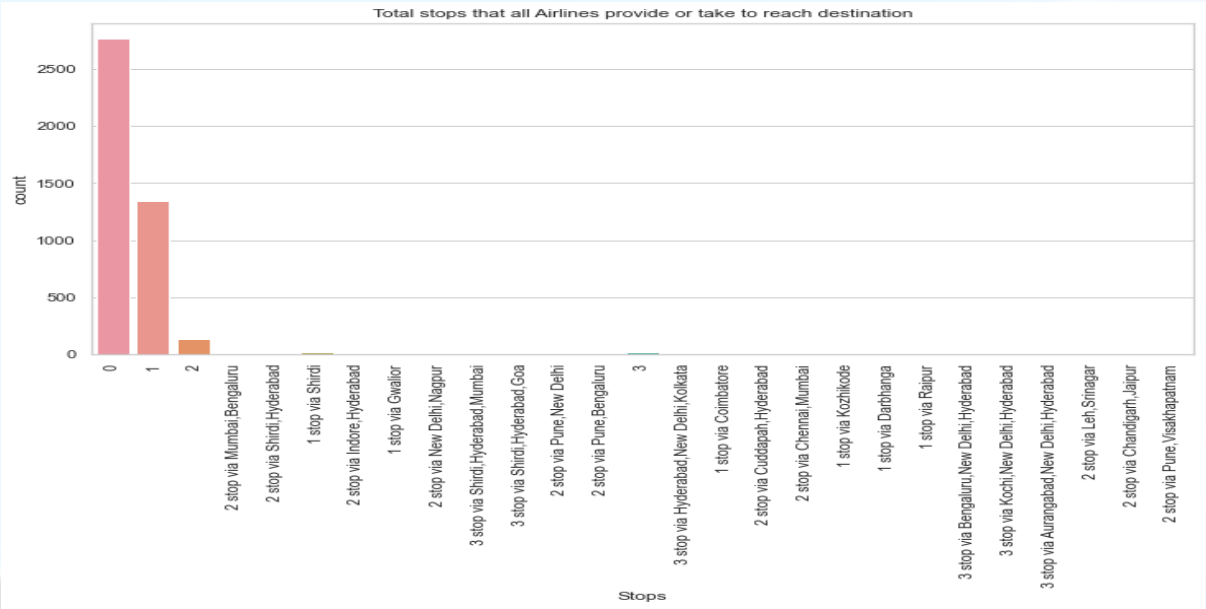




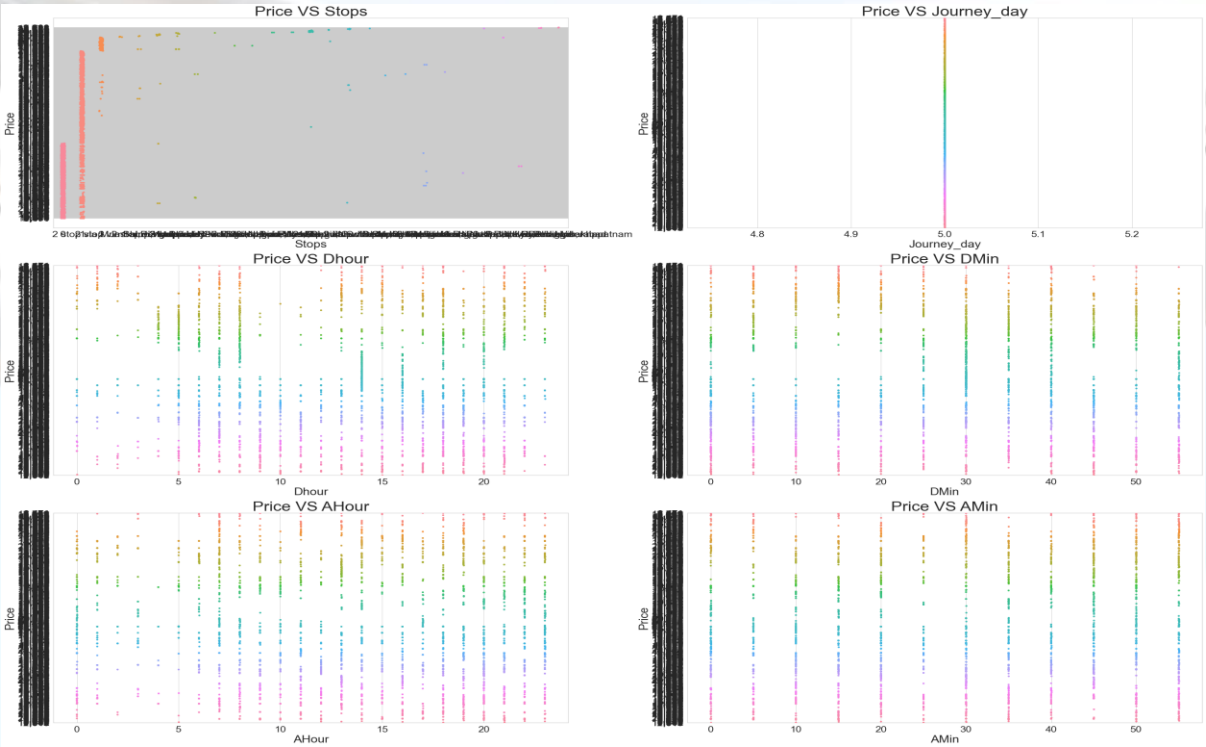
For Multiple Airlines the Price is high in Vistara compared to other Airlines.
Taking Goa as Source costs highest Price Compared to other Source points.
Taking New-Delhi as Destination costs highest Price Compared to other Destination points

Business Class" has a lot of additional benefits. An Economy class doesn't have much benefits and hence people prefer this more as it's relatively cheaper.

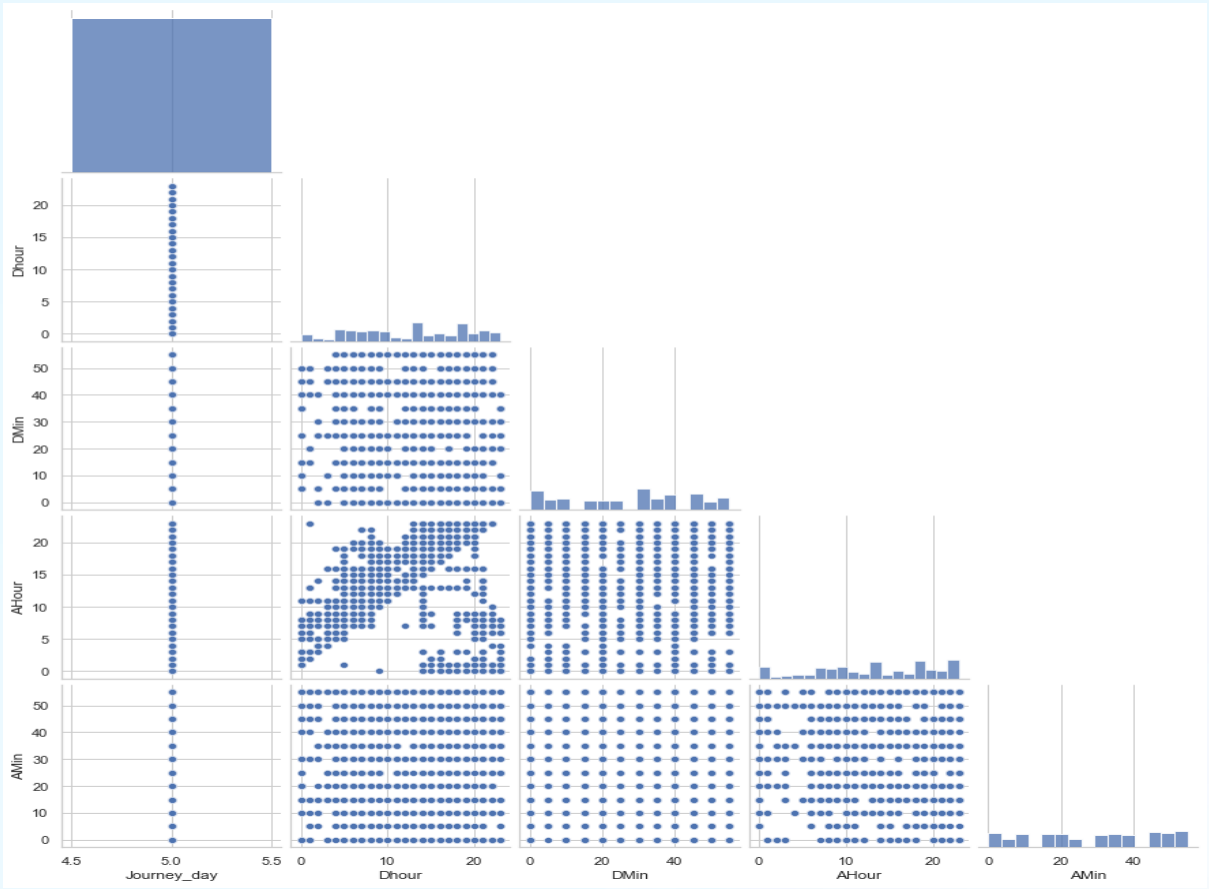
Plot shows us the number of stops given by flights
Majority of flights have "Non-stop or one stop" and this shows that flights have reached directly to the destination with more amount of time for halt.



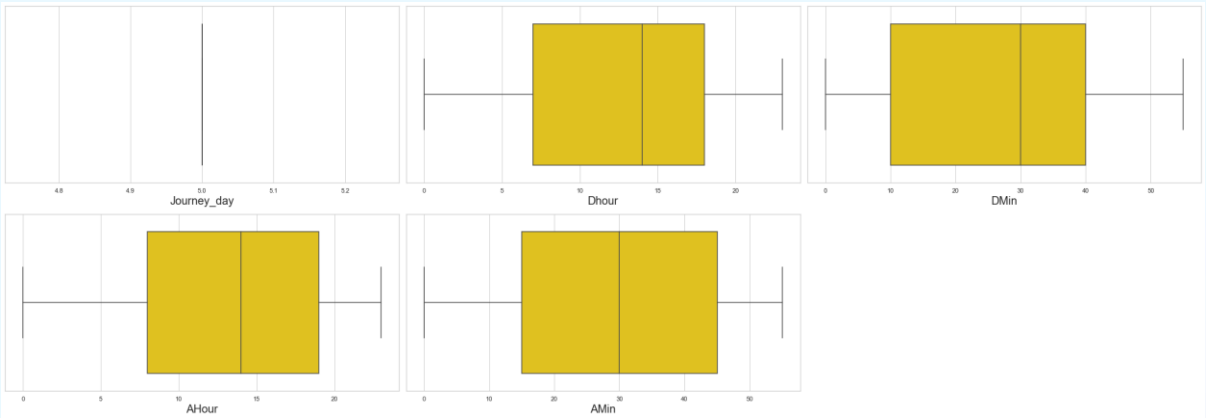
Bivariate Analysis:



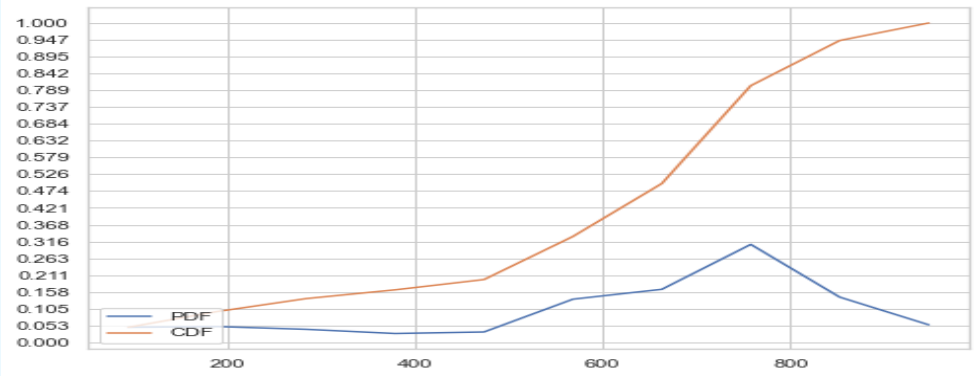
Multivariate Analysis: Pricing of flight per day, varies with time to time



Checking for outliers:

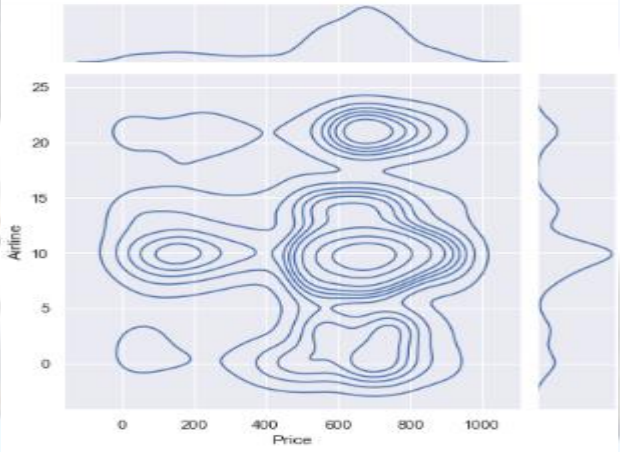


CDF / PDF:



Airline vs price view:

```
sns.set_style("darkgrid")
sns.jointplot("Price","Airline",data=df,kind="kde")
plt.show()
```



Checking correlation using heat map:

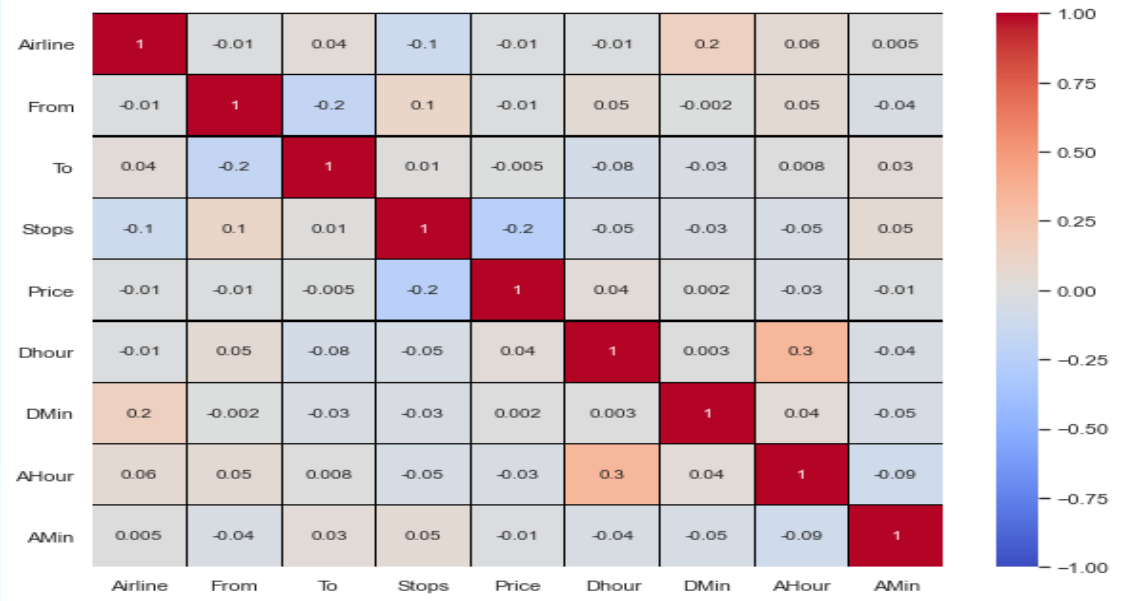
```
#Droping Journey_date column
df = df.drop(["Journey_day"],axis=1)
```

```
corr_matrix = df.corr()
corr_matrix["Price"].sort_values(ascending=False)
```

```
Price      1.000000
Dhour      0.038150
DMin       0.001540
To        -0.005361
Airline   -0.010147
AMin      -0.010629
From      -0.011739
AHour     -0.032207
Stops     -0.235309
Name: Price, dtype: float64
```

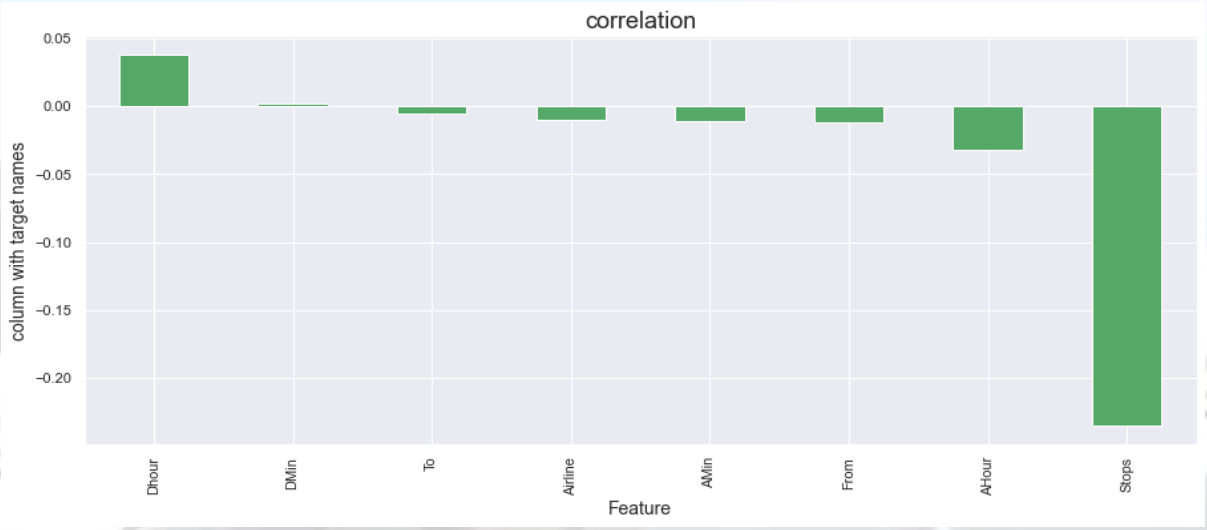
Correlations of all the pair of features.To get better visualization on the correlation of features,let me plot it using heat map.

Heat map:



Correlation:

There is no multi-collinearity issue in any features



- # Interpretation of the Results
  - In this research, two experiments were performed, the first experiment was collecting data using all the variables available in the dataset after pre-processing, while the second experiment was conducted using most important variables and the goal of this is to be able to improve the model's performance using fewer variables.
  - There requirement of train and test and building of many models to get accuracy.
  - There are multiple of matric which decide the best fit model like as : R-squared ,RMSE value, VIF, CDF & PDF Z-score and etc.

Database helped in making perfect model and will help in understanding Indian market of Flight price.

Saving the model and Predictions:

New Model

```
Best_mod=ExtraTreesRegressor(max_features='auto',min_samples_leaf=2,min_samples_split=2,n_estimators=80,n_jobs=1)
Best_mod.fit(X_train,y_train)
pred=Best_mod.predict(X_test)
print('R2_Score:',r2_score(y_test,pred)*100)
print('mean_squared_error:',metrics.mean_squared_error(y_test,pred))
print('mean_absolute_error:',metrics.mean_absolute_error(y_test,pred))
print("RMSE value:",np.sqrt(metrics.mean_squared_error(y_test, pred)))
```

```
R2_Score: 29.34568003118606
mean_squared_error: 31731.062284093347
mean_absolute_error: 104.17916475975738
RMSE value: 178.1321483733168
```

Saving New Model:

```
# Saving the model using .pkl
import joblib
joblib.dump(Best_mod,"Flight_Price.pkl")
```

```
['Flight_Price.pkl']
```

Predicting Flight Price for test dataset using Saved model of trained dataset:

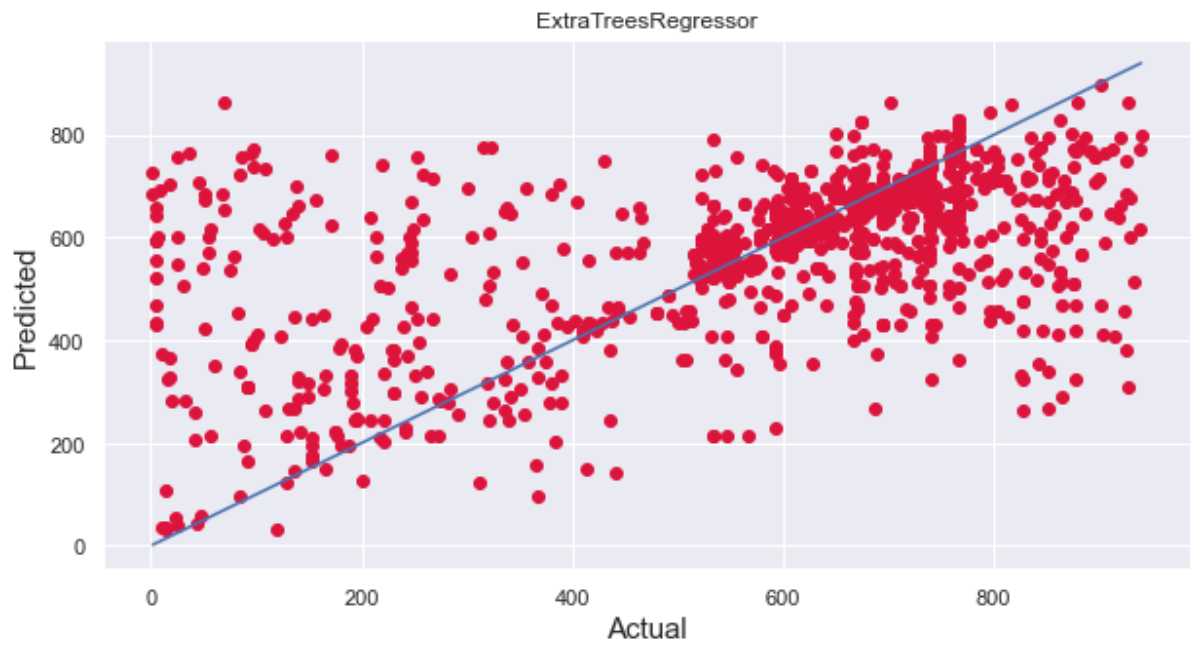
```
# Loading the saved model
model=joblib.load("Flight_Price.pkl")

#Prediction
prediction = model.predict(X_test)
prediction
```

```
array([712.53      , 665.21583333, 693.325     , ..., 597.883125  ,
       653.6025    , 475.92125   ])
```

```
pd.DataFrame([model.predict(X_test)[:],y_test[:]],index=["Predicted","Actual"])
```

	0	1	2	3	4	5	6	7	8	9	...	1293	1294
Predicted	712.53	665.215833	693.325	864.380417	766.0	618.43	766.0	211.540208	685.4675	623.5	...	564.013735	649.212083
Actual	806.00	563.000000	669.000	70.000000	766.0	56.00	766.0	218.000000	380.0000	632.0	...	213.000000	702.000000

Final Prediction:



Plotting Actual vs Predicted, To get better insight. Bule line is the actual line and red dots are the predicted values

CONCLUSION

• Key Findings and Conclusions of the Study

In this project report, we have used machine learning algorithms to predict the flight prices. We have mentioned the step by step procedure to analyse the dataset and finding the correlation between the features. Thus we can select the features which are correlated to each other and are independent in nature. These feature set were then given as an input to seven algorithms and a hyper parameter tuning was done to the best model and the accuracy has been improved. Hence we calculated the performance of each model using different performance metrics and compared them based on those metrics. Then we have also saved the best model and predicted the flight price.

In this machine learning in python project there is only one module namely, User. User can login with valid credentials in order to access the web application. A traveller can access this module to get the future price prediction of individual airlines. The prediction will help a traveller to decide a specific airline as per his/her budget.
Strategies are taking into consideration several financial, marketing, commercial and social factors are closely connected with the ultimate airfare prices. Due to the high complexity of the pricing models applied by the airlines, it is very difficult for a customer to purchase an air ticket at the lowest price.

• Learning Outcomes of the Study in respect of Data Science

- The dataset was quite interesting to handle as it contains all types of data in it and it was self scrapped from makemytrip website using selenium. Improvement in computing technology has made it possible to examine social information that cannot previously be captured, processed and analysed. New analytical techniques of machine learning can be used in flight price research. The power of visualization has helped us in understanding the data by graphical representation it has made me to understand what data is trying to say. Data cleaning is one of the most important steps to remove unrealistic values and null values. This study is an exploratory attempt to use seven machine learning algorithms in estimating flight price prediction, and then compare their results.
- To conclude, the application of machine learning in predicting flight price is still at an early stage. We hope this study has moved a small step ahead in providing some methodological and empirical contributions to crediting online platforms, and presenting an alternative approach to the valuation of flight price. Future direction of research may consider incorporating additional used flight data from a larger economical background with more features.

- # Limitations of this work and Scope for Future Work

- First drawback is scrapping the data as it is a fluctuating process.
- Followed by raw data which is not in format to analyse.
- Also, this study will not cover all Regression algorithms instead, it is focused on the chosen algorithm, starting from the basic ensembling techniques to the advanced ones.
- About Flight industry: It is the huge industry and the data given is quite small but for getting a start and for the decision making the data is quite sufficient. In this industry there is always open opportunities to get start. There are different raw data are available for real estate and applying data science to it, will move this industry at a next level.
- In this machine learning in python project there is only one module namely, Used. Single entries of current or previous data can be made. This training set is used to train the algorithm for accurate predictions.