FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted By :

Milind B. Kuwar

ACKNOWLEDGMENT

The project would not have been built without the constant support from DataTrained and Fliprobo teams.

References:

1. Notes and classes by DataTrained Academy

INTRODUCTION

- ## Business Problem Framing

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 travellers saying that flight ticket prices are so unpredictable. Optimal timing for airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we simulate various models for computing expected future prices and classifying whether this is the best time to buy the ticket.

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)

- ## Conceptual Background of the Domain Problem

The above problem statement clearly explains that the target variable is continuous & it's a regression problem as we need to predict the price of the flight tickets. So this can be solved by any of the below Regression Machine learning algorithms:
1. Linear Regression.
2. Decision Tree Regressor.
3. Random Forest Regressor.
I have mentioned only few. We would be performing each of the techniques later in this blog.
Main objective is to analyse and build a dynamic machine learning model that can predict the flight prices on the basis of information of the flight provided by the airlines

- ## Review of Literature

Based on the sample data we have scrapped from the websites like yatra & Indigo where we have understood the real time data & the valuation of ticket price in the market. The data set explains it is a regression problem as we need to build a model using Machine Learning in order to predict the price of ticket. Also, we have other independent features that would help to decide which all variables are important to predict the price of the variable and how do these variables describe the price of the used car.

Analytical Problem Framing

Analytical Modelling of the Problem

Methodology represents a description about the framework that is undertaken. It consists of various milestones that need to be achieved in order to fulfil the objective. We have various attributes that contribute retaining customers that aids the growth of business.

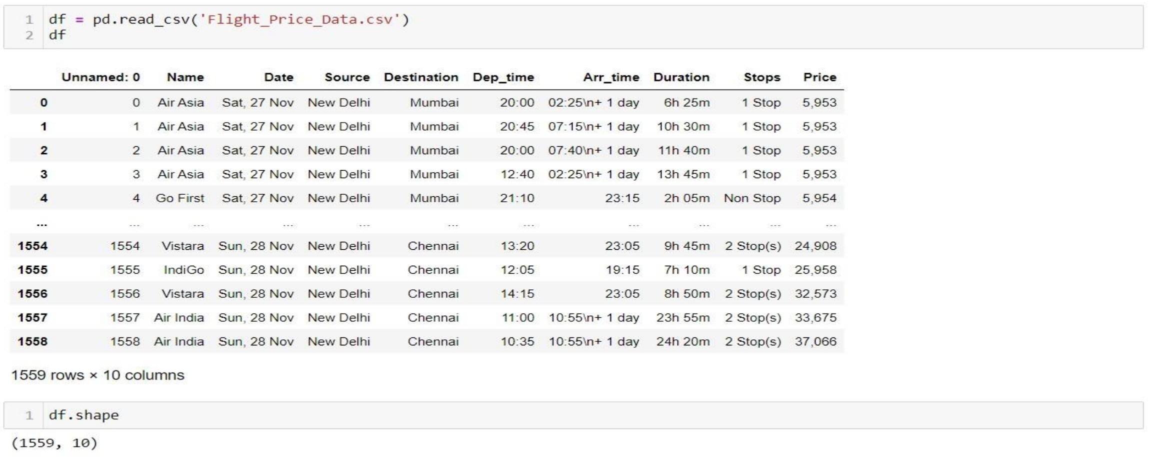
The following steps represents stepwise tasks that need to be completed:

- Data Collection
- Exploratory Data Analysis
- Data pre-processing
- Data Visualization
- Model Building
- Model Evaluation
- Saving the best model

Data Sources and their formats

The dataset contained 1559 rows and 10 columns.

Data Features : 9
Data Types : int64, float64, object



```
df = pd.read_csv('Flight_Price_Data.csv')
df
```

	Unnamed: 0	Name	Date	Source	Destination	Dep_time	Arr_time	Duration	Stops	Price
0	0	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:00	02:25\n+ 1 day	6h 25m	1 Stop	5,953
1	1	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:45	07:15\n+ 1 day	10h 30m	1 Stop	5,953
2	2	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:00	07:40\n+ 1 day	11h 40m	1 Stop	5,953
3	3	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	12:40	02:25\n+ 1 day	13h 45m	1 Stop	5,953
4	4	Go First	Sat, 27 Nov	New Delhi	Mumbai	21:10	23:15	2h 05m	Non Stop	5,954
...	...	...	...	...	...	...	...	...	...	...
1554	1554	Vistara	Sun, 28 Nov	New Delhi	Chennai	13:20	23:05	9h 45m	2 Stop(s)	24,908
1555	1555	IndiGo	Sun, 28 Nov	New Delhi	Chennai	12:05	19:15	7h 10m	1 Stop	25,958
1556	1556	Vistara	Sun, 28 Nov	New Delhi	Chennai	14:15	23:05	8h 50m	2 Stop(s)	32,573
1557	1557	Air India	Sun, 28 Nov	New Delhi	Chennai	11:00	10:55\n+ 1 day	23h 55m	2 Stop(s)	33,675
1558	1558	Air India	Sun, 28 Nov	New Delhi	Chennai	10:35	10:55\n+ 1 day	24h 20m	2 Stop(s)	37,066

1559 rows × 10 columns

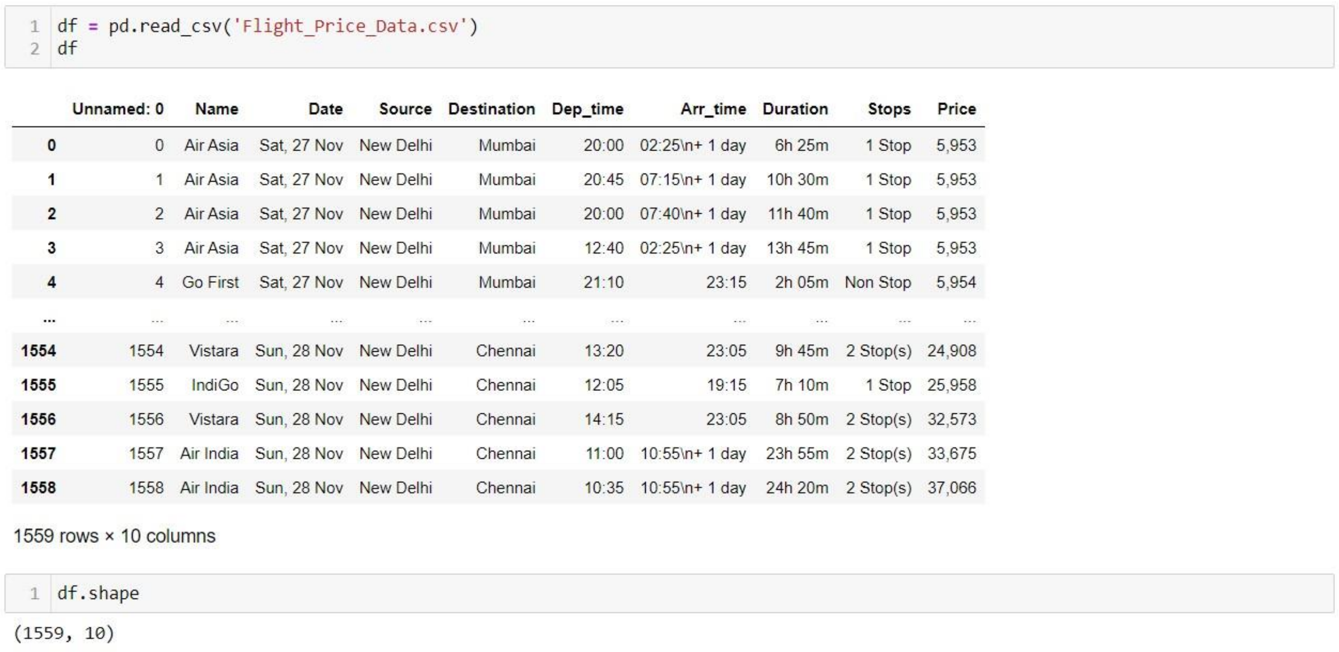
```
df.shape
```

```
(1559, 10)
```

Data Preprocessing

- ## Data Preprocessing Done

  - Checked for null values and identified no null values in dataset.
  - Outliers and skewness were checked.



```
df = pd.read_csv('Flight_Price_Data.csv')
df
```

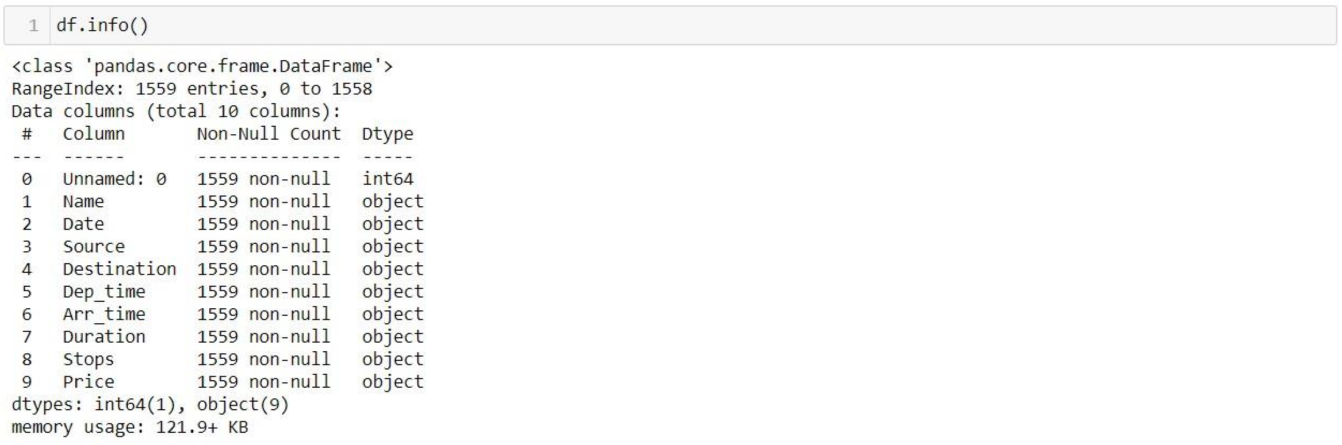
	Unnamed: 0	Name	Date	Source	Destination	Dep_time	Arr_time	Duration	Stops	Price
0	0	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:00	02:25\n+ 1 day	6h 25m	1 Stop	5,953
1	1	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:45	07:15\n+ 1 day	10h 30m	1 Stop	5,953
2	2	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	20:00	07:40\n+ 1 day	11h 40m	1 Stop	5,953
3	3	Air Asia	Sat, 27 Nov	New Delhi	Mumbai	12:40	02:25\n+ 1 day	13h 45m	1 Stop	5,953
4	4	Go First	Sat, 27 Nov	New Delhi	Mumbai	21:10	23:15	2h 05m	Non Stop	5,954
...	...	...	...	...	...	...	...	...	...	...
1554	1554	Vistara	Sun, 28 Nov	New Delhi	Chennai	13:20	23:05	9h 45m	2 Stop(s)	24,908
1555	1555	IndiGo	Sun, 28 Nov	New Delhi	Chennai	12:05	19:15	7h 10m	1 Stop	25,958
1556	1556	Vistara	Sun, 28 Nov	New Delhi	Chennai	14:15	23:05	8h 50m	2 Stop(s)	32,573
1557	1557	Air India	Sun, 28 Nov	New Delhi	Chennai	11:00	10:55\n+ 1 day	23h 55m	2 Stop(s)	33,675
1558	1558	Air India	Sun, 28 Nov	New Delhi	Chennai	10:35	10:55\n+ 1 day	24h 20m	2 Stop(s)	37,066

1559 rows × 10 columns

```
df.shape
```

```
(1559, 10)
```

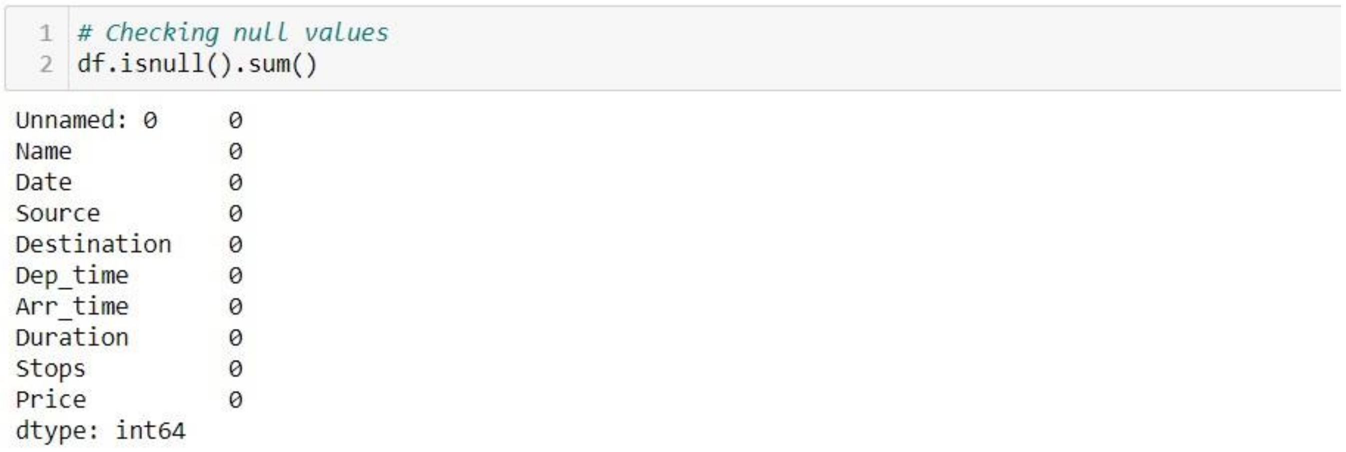
Name: Name of the airline used for traveling
Date : Date at which a person travelled
Source: Starting location of flight
Destination: Ending location of flight
Departure Time: Departure time of flight from starting location
Arrival Time: Arrival time of flight at destination
Duration: Duration of flight in hours/minutes
Stops: Number of total stops flight took before landing at the destination.
Price: Price of the flight.(Target Variable)



```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1559 entries, 0 to 1558
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Unnamed: 0   1559 non-null   int64 
 1   Name         1559 non-null   object
 2   Date         1559 non-null   object
 3   Source       1559 non-null   object
 4   Destination  1559 non-null   object
 5   Dep_time     1559 non-null   object
 6   Arr_time     1559 non-null   object
 7   Duration     1559 non-null   object
 8   Stops        1559 non-null   object
 9   Price        1559 non-null   object
dtypes: int64(1), object(9)
memory usage: 121.9+ KB
```

We have 1559 rows & 9 columns in the dataset. We have maximum columns as categorical, which is an• independent feature.



```
# Checking null values
df.isnull().sum()
```

```
Unnamed: 0     0
Name           0
Date           0
Source         0
Destination    0
Dep_time       0
Arr_time       0
Duration       0
Stops          0
Price          0
dtype: int64
```

For cleaning the data firstly the data was checked for inconsistencies by checking the standard deviation. On finding higher std. values it could be guessed that there were outliers present in the data. In order to get a better understanding of the data, we plotted distribution of data. We noticed that the dataset had some outliers, which was negligible.

- ## Hardware and Software Requirements and Tools Used

  Listing down the hardware and software requirements along with the tools, libraries and packages used. Describe all the software tools used along with a detailed description of tasks done with those tools.

2. Lenovo ideapad145 laptop
3. Jupyter Notebook - The Jupyter Notebook is an interactive environment for running code in the browser. It is a great tool for exploratory data analysis and is widely used by data scientists.
4. MS powerpoint - For preparing presentation of project.
5. MS word - For preparing report
6. Matplotlib - For data visualization to produce high quality plots, charts and graphs.
7. Pandas - Python library for data wrangling and analysis. It is built around a data structure called DataFrame.
8. Numpy - NumPy is one of the fundamental packages for scientific computing in Python.

Model/s Development and Evaluation

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

  I have use statistical models to find insights given set of data. Predictions are assessed differently depending on its type: Based on the data structure of Target variable, Model type was identified

  First and important step was to identify the model which was done by analysing the target variable ‘Price. Best R2 score was achieved using Random Forest Regressor.

- ## Testing of Identified Approaches (Algorithms)

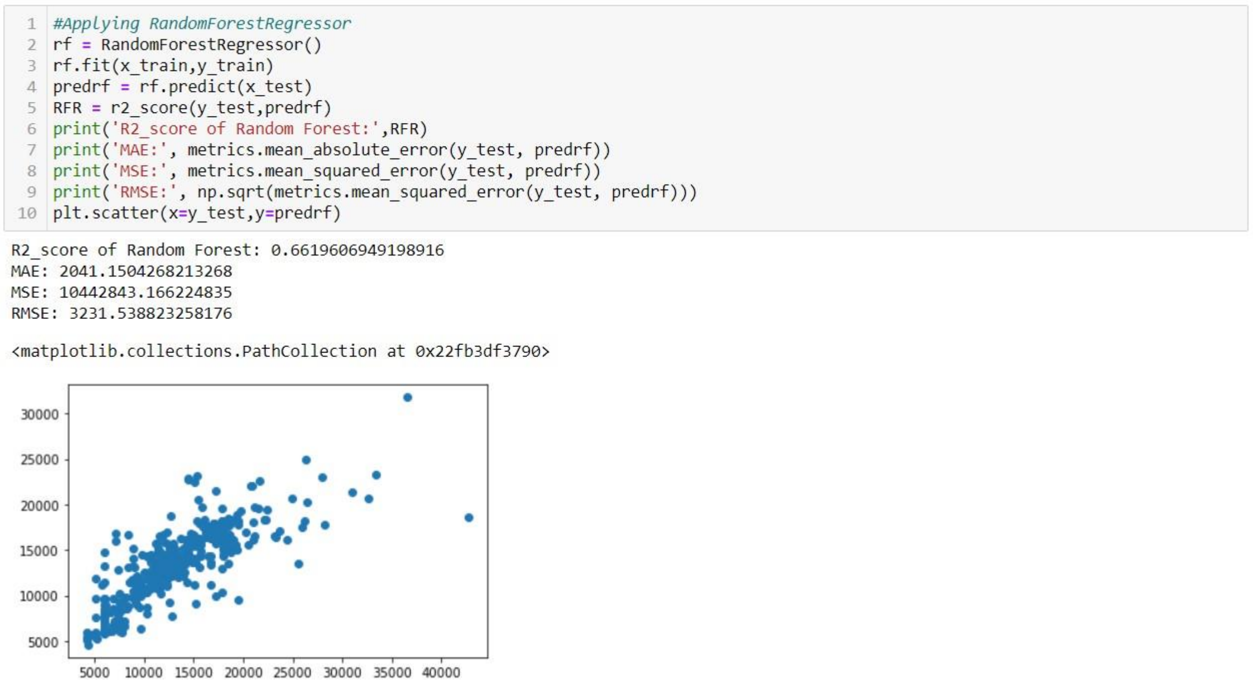
  Build Models based on learning it was a supervised classification problem.
  I built 5 models to evaluate performance of each of them:

  1. **Random Forest Regressor**
  2. **Decision Tree Regressor**
  3. **KNeighbors Regressor**
  4. **Gradient Boosting Regressor**
  5. **BaggingRegressor**
  6. **AdaBoost Regressor**

- **Run and Evaluate selected models**

Describing all the algorithms used along with the snapshot of their code and what were the results observed over different evaluation metrics.

Random Forest Regressor



```
#Applying RandomForestRegressor
rf = RandomForestRegressor()
rf.fit(x_train,y_train)
predrf = rf.predict(x_test)
RFR = r2_score(y_test,predrf)
print('R2_score of Random Forest:',RFR)
print('MAE:', metrics.mean_absolute_error(y_test, predrf))
print('MSE:', metrics.mean_squared_error(y_test, predrf))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predrf)))
plt.scatter(x=y_test,y=predrf)
```

```
R2_score of Random Forest: 0.6619606949198916
MAE: 2041.1504268213268
MSE: 10442843.166224835
RMSE: 3231.538823258176

<matplotlib.collections.PathCollection at 0x22fb3df3790>
```

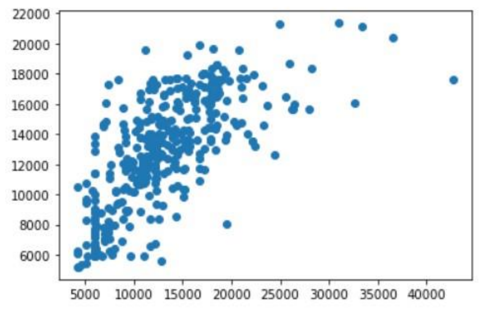
It uses ensemble learning method for regression. Ensemble learning method is a technique that combines predictions from multiple machine learning algorithms to make a more accurate prediction based on majority voting/averaging.

KNN Regressor

```
#Applying KNeighborsRegressor
KNN = KNeighborsRegressor(n_neighbors = 5)
KNN.fit(x_train,y_train)
predknn = KNN.predict(x_test)
KNNR = r2_score(y_test,predknn)
print('R2_score of KNN :',KNNR)
print('MAE:', metrics.mean_absolute_error(y_test, predknn))
print('MSE:', metrics.mean_squared_error(y_test, predknn))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predknn)))
plt.scatter(x=y_test,y=predknn)
```

```
R2_score of KNN : 0.5073223607468256
MAE: 2615.109230769231
MSE: 15219991.406051284
RMSE: 3901.280739199793
```

```
<matplotlib.collections.PathCollection at 0x22fb77ada60>
```

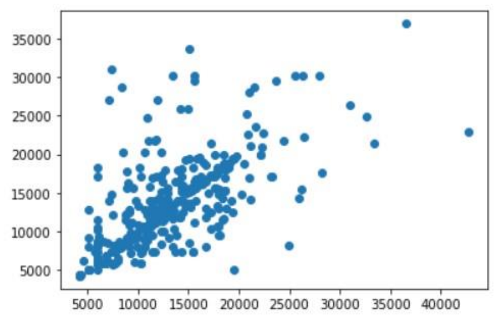


Decision Tree Regressor

```
#Applying DecisionTreeRegressor
dt = DecisionTreeRegressor()
dt.fit(x_train,y_train)
preddt = dt.predict(x_test)
DTR = r2_score(y_test,preddt)
print('R2_score of DT:',DTR)
print('MAE:', metrics.mean_absolute_error(y_test, preddt))
print('MSE:', metrics.mean_squared_error(y_test, preddt))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, preddt)))
plt.scatter(x=y_test,y=preddt)
```

```
R2_score of DT: 0.3051811522399246
MAE: 2634.905128205128
MSE: 21464617.123076923
RMSE: 4632.992242932954
```

```
<matplotlib.collections.PathCollection at 0x22fb8a00c40>
```



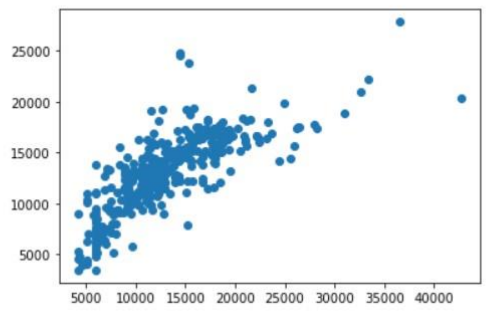
Gradient Boosting Regressor

```python
from sklearn.ensemble import GradientBoostingRegressor
GB=GradientBoostingRegressor()
GB.fit(x_train,y_train)
predgb = GB.predict(x_test)
GBR = r2_score(y_test,predgb)
print('R2_score of GB :',GBR)
print('MAE:', metrics.mean_absolute_error(y_test, predgb))
print('MSE:', metrics.mean_squared_error(y_test, predgb))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predgb)))
plt.scatter(x=y_test,y=predgb)

```

```
R2_score of GB : 0.6158109541408174
MAE: 2336.663872271142
MSE: 11868519.109451598
RMSE: 3445.071713252367
```

```
<matplotlib.collections.PathCollection at 0x22fb78d39d0>
```



- Since RandomForestRegressor is the best model in terms of model score, cross validation difference, test & train, r2 score difference, also as per the evaluation metrics, RandomForestRegressor to be the final model. Lets see if we can increase the score by using hyper parameter tuning.

```python
params = {'n_estimators': range(50,200,10),
               'max_features': ['auto', 'sqrt'],
               'max_depth': range(5,30,6),
               'min_samples_split': [2, 5, 10, 15, 100],
               'min_samples_leaf': [1, 2, 5, 10]}
```

```python
from sklearn.model_selection import RandomizedSearchCV
rf = RandomizedSearchCV(RandomForestRegressor(),params,scoring='neg_mean_squared_error',n_iter = 10,cv=5, n_jobs= -1,random_
rf.fit(x_train,y_train)
```

```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=-1,
                   param_distributions={'max_depth': range(5, 30, 6),
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': range(50, 200, 10)},
                   random_state=42, scoring='neg_mean_squared_error')
```

```python
rf.best_params_
```

```
{'n_estimators': 90,
 'min_samples_split': 5,
 'min_samples_leaf': 2,
 'max_features': 'auto',
 'max_depth': 17}
```

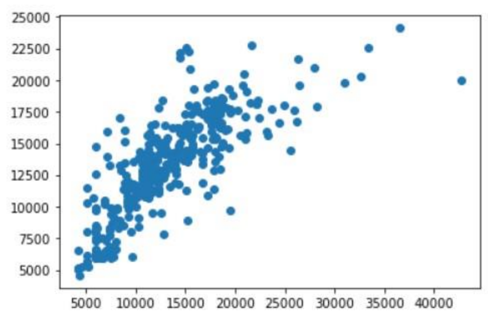
```python
pred = rf.predict(x_test)
print(r2_score(y_test,pred))
```

```
0.6480629736403024
```

```python
#Applying RandomForestRegressor
rf = RandomForestRegressor(n_estimators = 90 , min_samples_split = 5, min_samples_leaf = 2, max_features = 'auto', max_depth
rf.fit(x_train,y_train)
predrf = rf.predict(x_test)
RFR = r2_score(y_test,predrf)
print('R2_score of Random Forest:',RFR)
print('MAE:', metrics.mean_absolute_error(y_test, predrf))
print('MSE:', metrics.mean_squared_error(y_test, predrf))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predrf)))
plt.scatter(x=y_test,y=predrf)
```

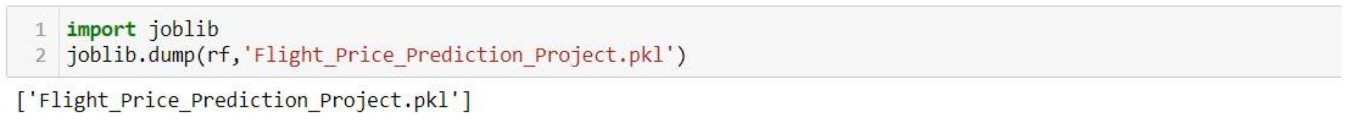
```
R2_score of Random Forest: 0.6480835949125017
MAE: 2110.4627357120066
MSE: 10871539.997632796
RMSE: 3297.2018436293515

<matplotlib.collections.PathCollection at 0x22fb3e690a0>
```



- RandomForestRegressor is giving best result out of other model.

Saving Model:



```
import joblib
joblib.dump(rf,'Flight_Price_Prediction_Project.pkl')
```

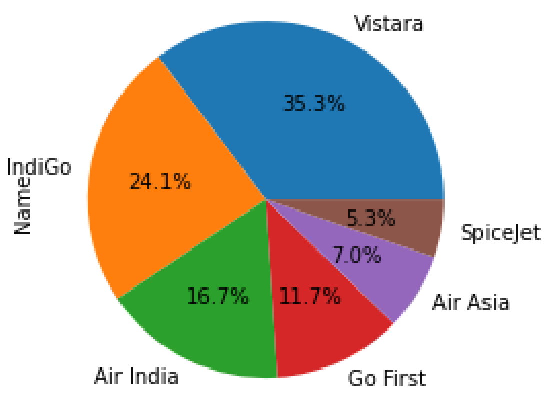
```
['Flight_Price_Prediction_Project.pkl']
```

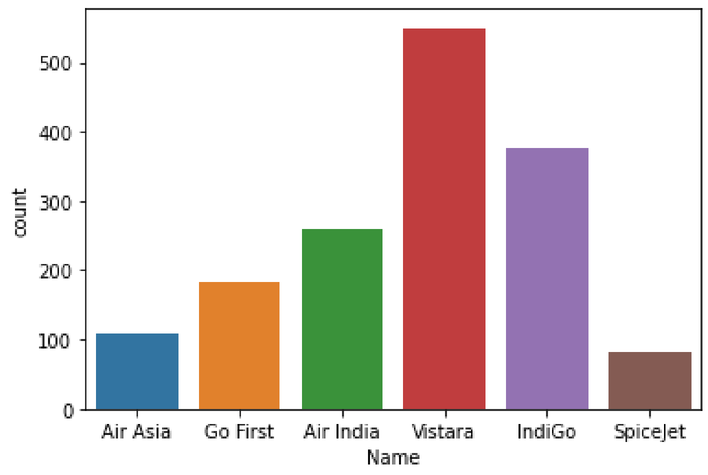
- **Key Metrics for success in solving problem under consideration**

  I have used above shown screenshot of key metrics for model building & it helped me to understand why out of 8 models, I choose the best model based on the metrics such as

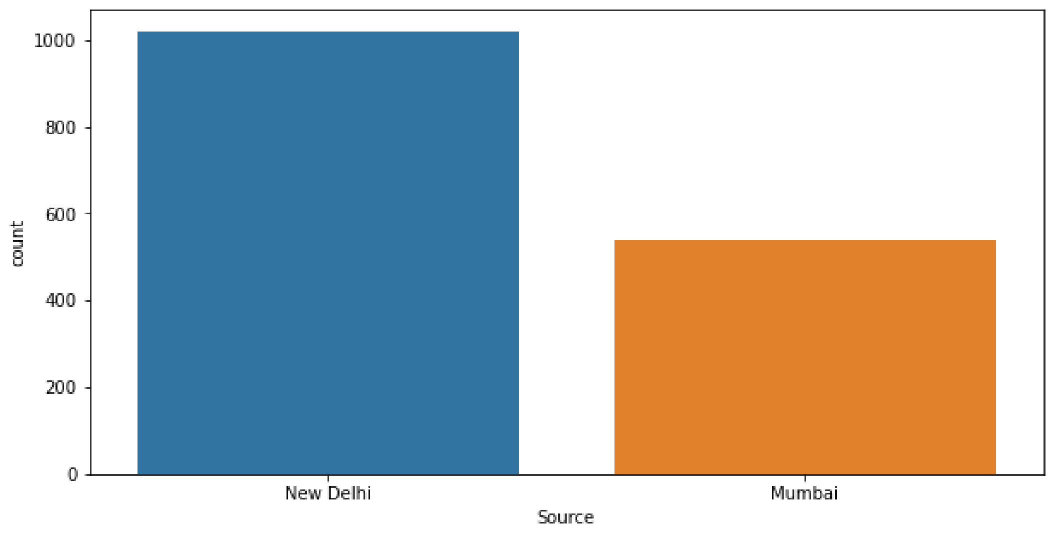
  - R2 score,
  - Mean_squared_error
  - Mean_absolute_error,
  - Root_mean_squared_error,
  - Hyper parameter tuning to increase the accuracy.

- **Visualizations & Interpretation of the Results**

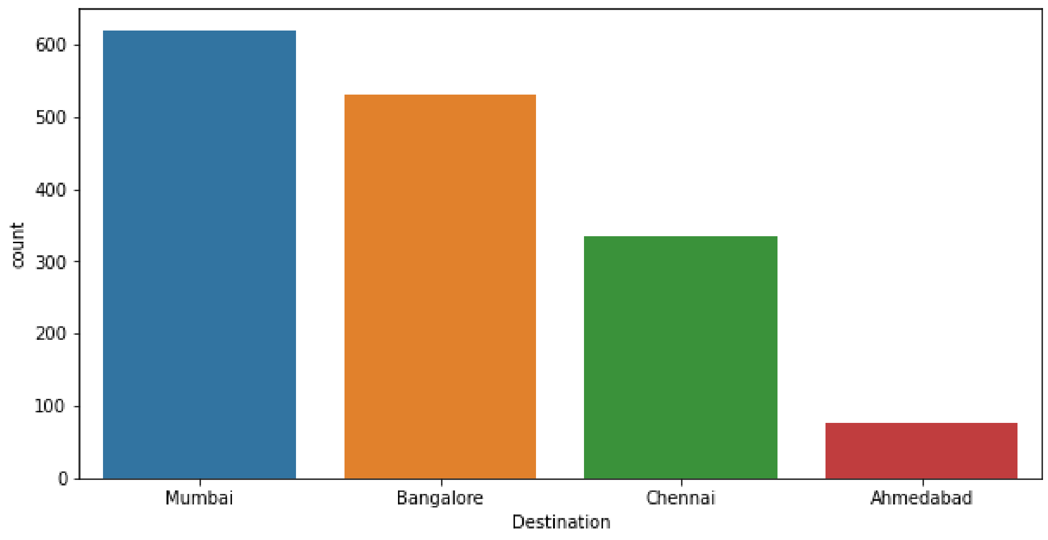




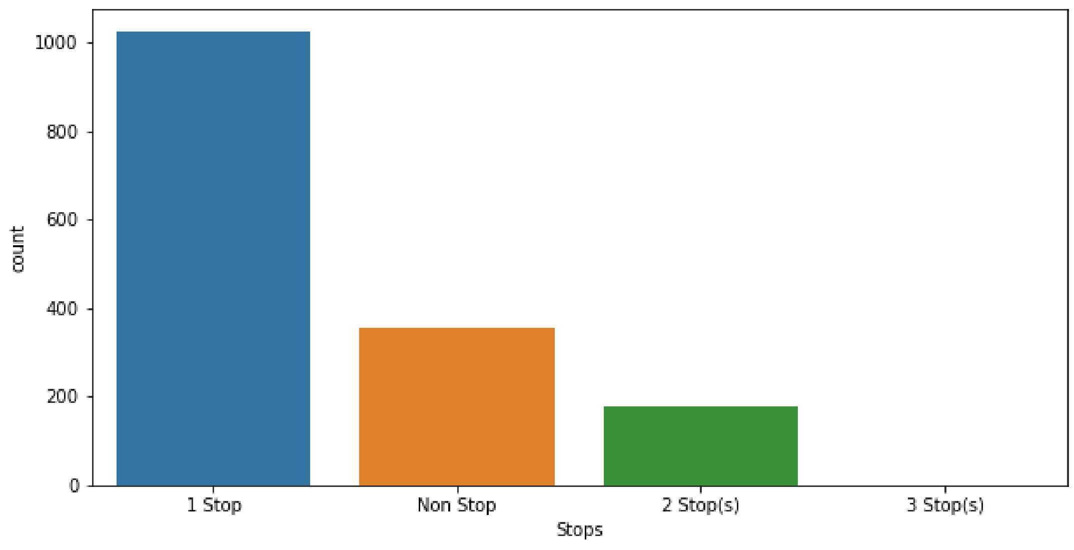
Vistara is higher and lowest is Spicejet.
AirIndia and Indigo has comparable, followed by GoFirst.



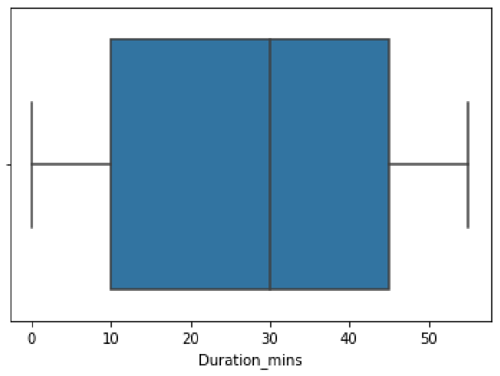
count
0
200
400
600
800
1000
New Delhi
Mumbai
Source

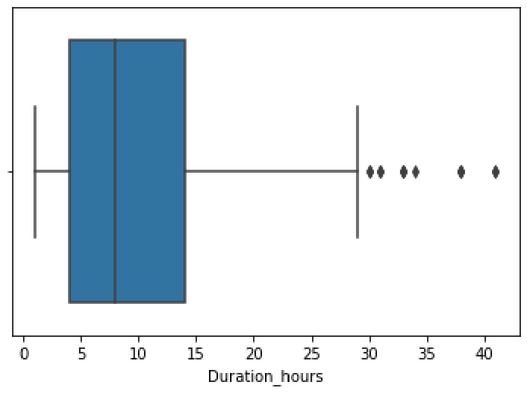


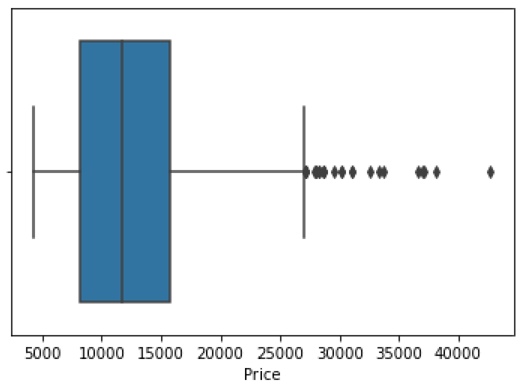
count
0
100
200
300
400
500
600
Mumbai
Bangalore
Chennai
Ahmedabad
Destination

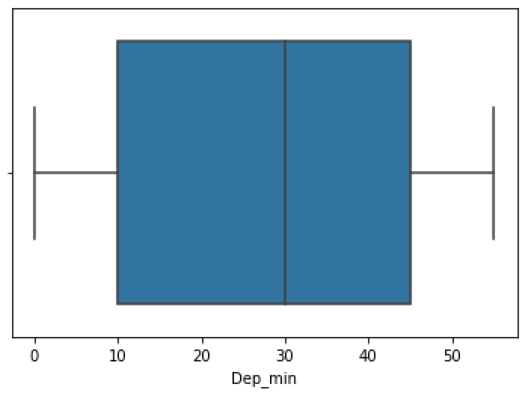


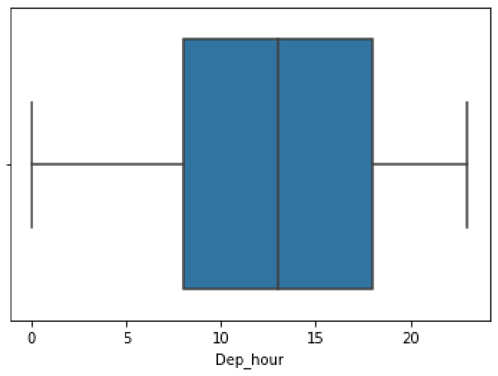
count
0
200
400
600
800
1000
1 Stop
Non Stop
2 Stop(s)
3 Stop(s)
Stops

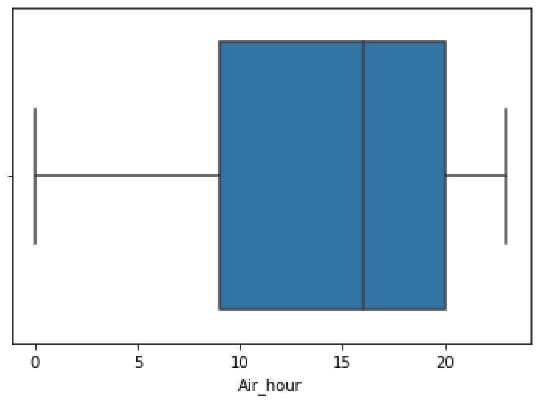

0
10
20
30
40
50
Duration_mins

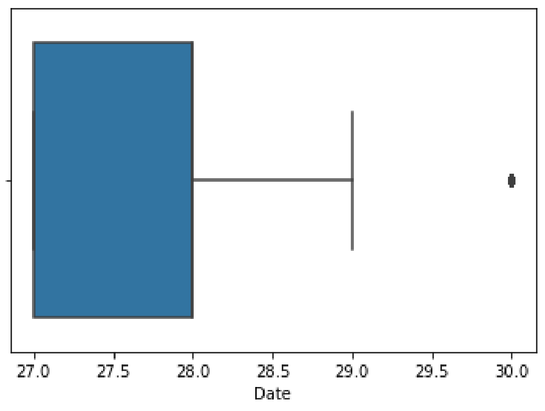

0
5
10
15
20
25
30
35
40
Duration_hours

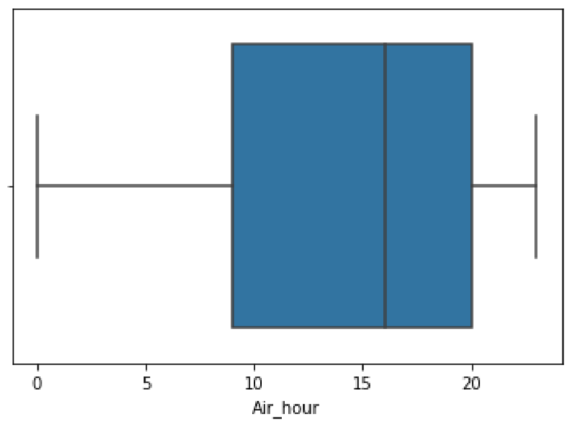

5000
10000
15000
20000
25000
30000
35000
40000
Price

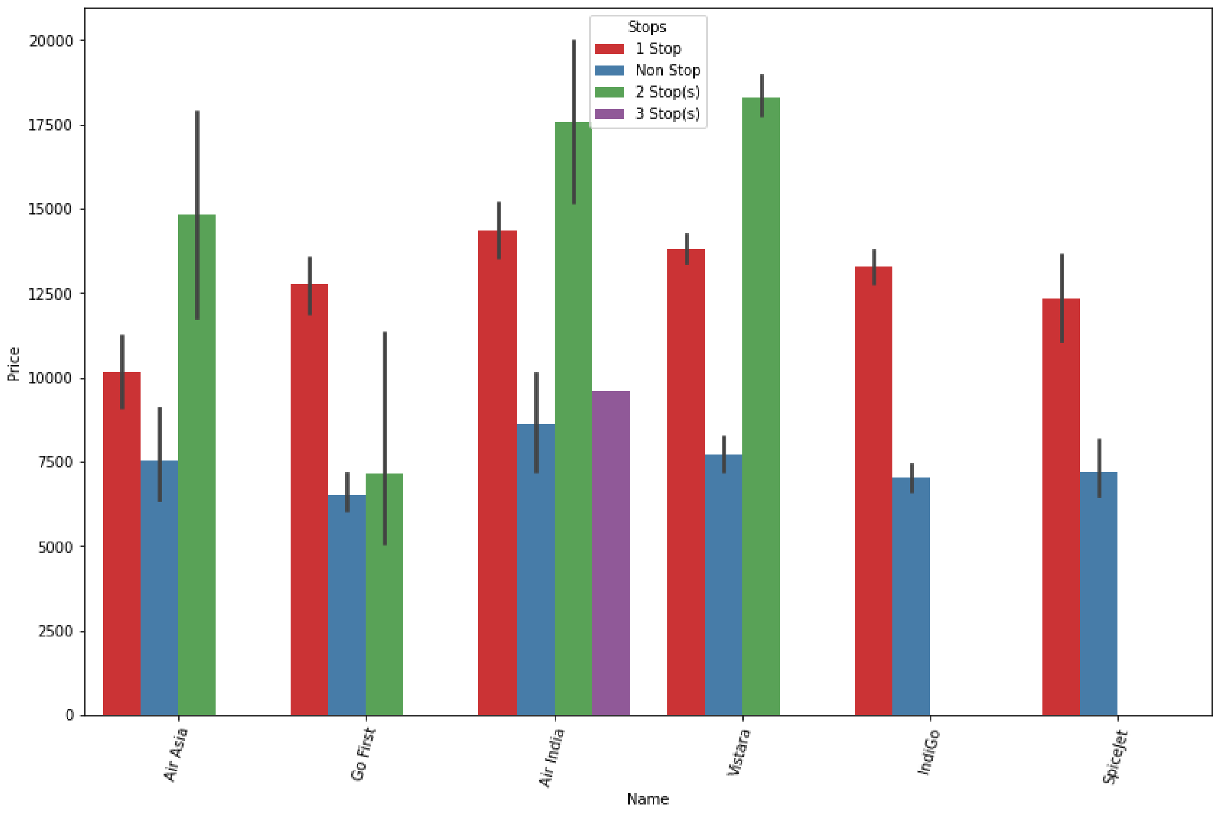

0
10
20
30
40
50
Dep_min


0
5
10
15
20
Dep_hour


0
5
10
15
20
Air_hour

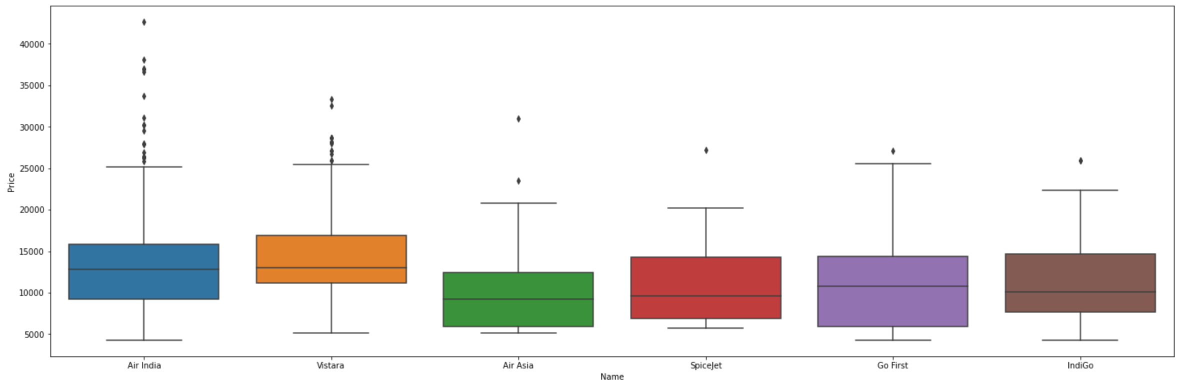

27.0
27.5
28.0
28.5
29.0
29.5
30.0
Date

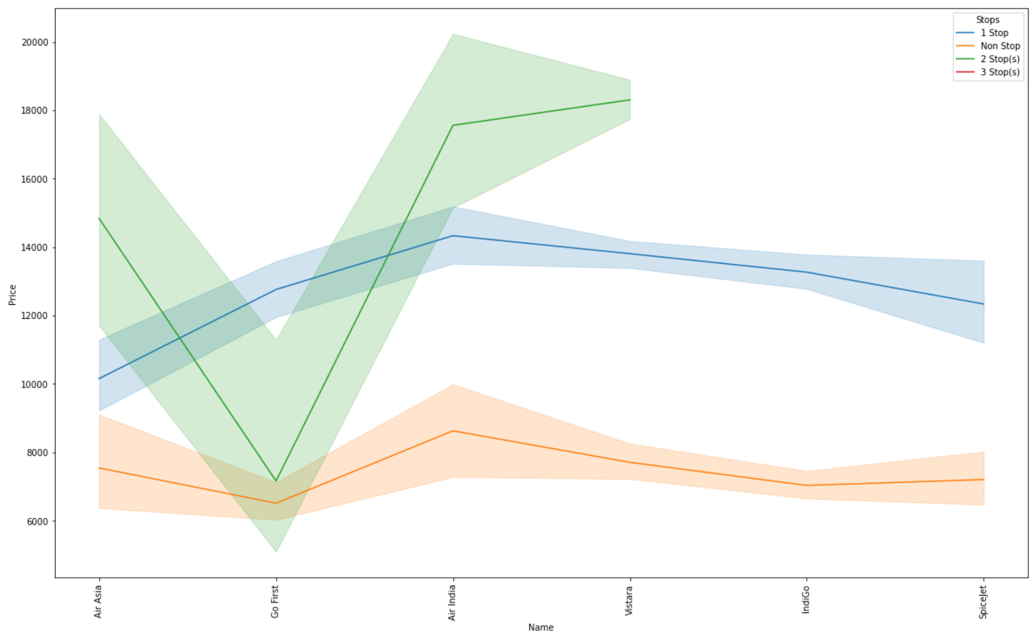

0
5
10
15
20
Air_hour



Vistara is showing the highest price having a 2 stops followed by 1 stop.

Spice jet is selling tickets with maximum fare followed by Go first & Vistara



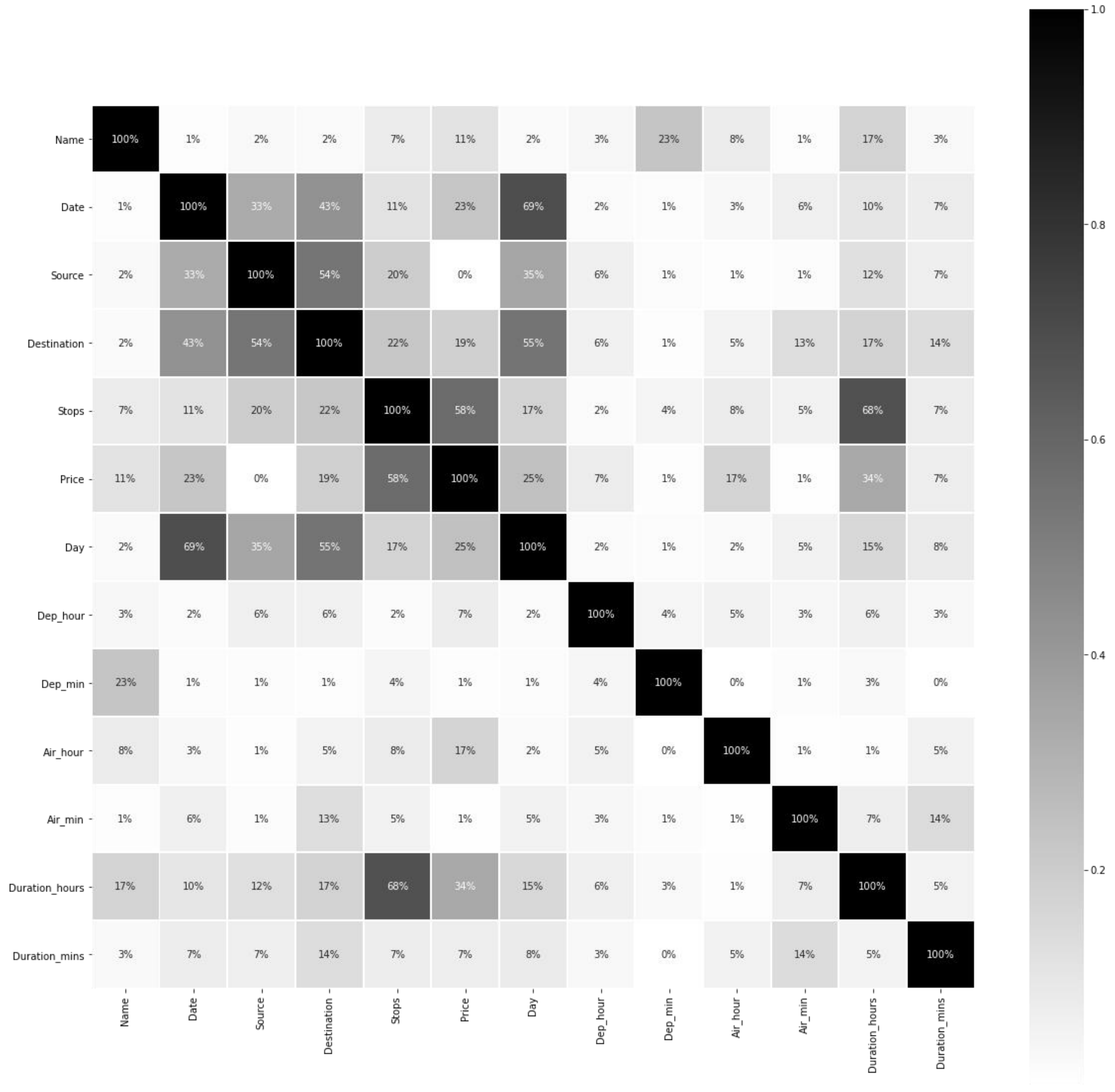


Action Taken based on observation:

We have seen how categorical data is distributed & correlated with the target variable. Also, I have mentioned the findings below each plot.

Performed One Hot Encoding for Nominal categorical columns: Airline & Destination & performed label encoding for an ordinal categorical column: Total Stops, Source & Day.

No need to check for outliers/skewness as all the independent features are categorical. Visualizing the correlation in heat map to check if there is any coefficient of multicolinearity



1)Total stops is more correlated with target variable.
2)Multi collinearity: Date & Source

CONCLUSION

Flight prices almost always remain constant or increase between the major cities There are two groups of airlines: the economical group and the luxurious group. SpiceJet, AirAsia, IndiGo, Go Air are in the economical class, whereas Jet Airways and Air India in the other. Vistara has a more spread out trend

As we have seen, the prediction is showing a similar relationship with the actual price from the scrapped data set, which means the model predicted correctly & this could help airlines by predicting what prices they can maintain. It could also help customers to predict future flight prices and plan the journey accordingly because it is difficult for airlines to maintain prices since it changes dynamically due to different conditions. Hence by using Machine learning techniques we can solve this problem.