FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by-:
NITISH KUMAR SHARMA

ACKNOWLEDGEMENT

It is great pleasure for me to undertake this project. I feel overwhelmed doing this project entitled – “Flight Price Prediction“.

Some of the resources that helped me to complete this project are as follows:

- Internet/web
- Stack overflow
- Analytics Vidhya
- Articles published in Medium.com
- Wikipedia

INTRODUCTION

BUSINESS PROBLEM FRAMING

This project includes the real time problem for customers and airlines where customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination.

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)

Economy or GDP growth gets affected if private or government airlines does not run to its full seating capacity due to high fixed and variable costs.

Our goal is to scrape data of flight fares with other features from different websites and build a model to predict fares of flights.

CONCEPTUAL BACKGROUND OF THE DOMAIN PROBLEM

The airline industry is considered as one of the most sophisticated industry in using complex pricing strategies. Nowadays, ticket prices can vary dynamically and significantly for the same flight, even for nearby seats.

The ticket price of a specific flight can change up to seven times a day . However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company loosing revenue. Airlines companies are generally equipped with advanced tools and capabilities that enable them to control the pricing process. However, customers are also becoming more strategic with the development of various online tools to compare prices across various airline companies. In addition, competition between airlines makes the task of determining optimal pricing is hard for everyone.

REVIEW OF LITERATURE

Airline businesses around the world was wiped out by Covid-19 as most international air travel was stopped in emergency.
The Airline Companies is considered as one of the most enlightened industries using complex methods and complex strategies to allocate airline prices in a dynamic fashion. These industries are trying to keep their all inclusive revenue as high as possible and boost their profit.

Since we have continous target (price) variable, so regression model algorithms has been used.

We will start our project with the dataset which contains eight object and one integer datatype of features. We will observe all the features with following goals in mind:

- Relevance of the features
- Distribution of the features
- Data Cleaning of the features
- Visualization of the features

After having gone through all the features and cleaning the dataset, we will move on to machine learning classification modelling:

- Pre-processing of the dataset for models
- Testing multiple algorithms with multiple evaluation metrics
- Select evaluation metric as per our specific business application
- Doing hyper-parameter tuning using GridSearchCV for the best model
- Finally saving the best model
- Predicting with the test dataset

MOTIVATION FOR THE PROBLEM UNDERTAKEN

I was interested in “Flight Price Prediction” project to help tourists find the right flight airfare based on their needs.

Finally it is important to work on application (real world application) to actually make a difference.

Due to COVID impact, airline industries suffered huge financial losses as travelling was prohibited. So its like a huge responsibility to build a proper model for predicting flight price after COVID impact, so that airline industries can maximize profits and passengers get fair value of tickets for their satisfaction.

ANALYTICAL PROBLEM FRAMING

MATHEMATICAL / ANALYTICAL MODELING OF THE PROBLEM

The dataset contains csv file : It has total nine features including one target variable (price). The attributes are airlines, date of journey, source, destination, duration, departure time, arrival time, total stops and price.

To know the overview/stats of the dataset , we will be using df.describe() function which gives informations like count, min, max, mean, standard deviation values of features . By plotting of heat map we can correlate features if they are highly inter correlated or not. So we can drop columns if problem of multicollinearity arises (if vif >5) when we observe through variance inflation factor.

```
df.describe().T # to get high understanding of dataset or to get overview/stats of the dataset
```

	count	mean	std	min	25%	50%	75%	max
Price	1588.0	9703.163098	4234.597613	2346.0	6777.5	8989.5	11657.0	31818.0

From an initial statistical overview of the dataset, we infer that our target variable is of continuous datatype.

DATA SOURCES AND THEIR FORMATS

The data that I scraped from website was in CSV format (Comma Separated Values). After reading the data by using function df=pd.read_csv (‘ ---file path --- ’) in jupyter notebook there were 1588 rows and 9 columns.

Dataset/Attributes Description :

1. Airline - Names of different flights/Airlines
2. Date of Journey - Day on which journey begins (i.e to travel by flight)
3. Source - City from where flight takes off
4. Destination - City where flight lands
5. Duration - Total time taken by flight to reach its destination (in hrs mins)
6. Departure time - Time when flight takes off
7. Arrival Time - Time when flight lands
8. Total Stops - Number of stoppages
9. Price - Flight ticket price (in inr)

Dataset datatypes are as follow :

```
df.info() # to know datatype of each columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1588 entries, 0 to 1587
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline          1588 non-null   object
 1   Date_of_Journey  1588 non-null   object
 2   Source           1588 non-null   object
 3   Destination      1588 non-null   object
 4   Duration         1588 non-null   object
 5   Departure Time   1588 non-null   object
 6   Arrival Time     1588 non-null   object
 7   Total Stops      1588 non-null   object
 8   Price            1588 non-null   int64 
dtypes: int64(1), object(8)
memory usage: 111.8+ KB
```

```
# to count number of unique values in each columns
df.nunique()
```

```
Airline              6
Date_of_Journey     13
Source              13
Destination          8
Duration           315
Departure Time     238
Arrival Time       331
Total Stops          4
Price              687
dtype: int64
```

```
# to list categorical features in our dataset
cat_features=[i for i in df.columns if df.dtypes[i]=='object']
cat_features
```

```
['Airline',
 'Date_of_Journey',
 'Source',
 'Destination',
 'Duration',
 'Departure Time',
 'Arrival Time',
 'Total Stops']
```

```
# to list continous features in our dataset
con_features=[i for i in df.columns if df.dtypes[i]=='int64' or df.dtypes[i]=='float64']
con_features
```

```
['Price']
```

DATA PREPROCESSING DONE

```
# to check format of date of journey
df['Date_of_Journey'].unique()
```

```
array(['29-10-2021', '30-10-2021', '31-10-2021', '02-11-2021',
       '05-11-2021', '08-11-2021', '11-11-2021', '10-11-2021',
       '27-11-2021', '29-11-2021', '13-12-2021', '10-12-2021',
       '30-12-2021'], dtype=object)
```

- The date of journey scraped as shown above was in different format which could not be further used for analysis.

```
# Converting Date_of_Journey format into timestamp so as to use this column properly for prediction.
df['Date_of_Journey']=df['Date_of_Journey'].replace('29-10-2021','29/10/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('30-10-2021','30/10/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('31-10-2021','31/10/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('02-11-2021','02/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('05-11-2021','05/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('08-11-2021','08/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('11-11-2021','11/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('10-11-2021','10/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('27-11-2021','27/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('29-11-2021','29/11/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('13-12-2021','13/12/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('10-12-2021','10/12/2021',regex=True)
df['Date_of_Journey']=df['Date_of_Journey'].replace('30-12-2021','30/12/2021',regex=True)
```

- So converted the format of date of journey into timestamp format by using replace function.

```
# checking the format again
df['Date_of_Journey'].unique()
```

```
array(['29/10/2021', '30/10/2021', '31/10/2021', '02/11/2021',
       '05/11/2021', '08/11/2021', '11/11/2021', '10/11/2021',
       '27/11/2021', '29/11/2021', '13/12/2021', '10/12/2021',
       '30/12/2021'], dtype=object)
```

- Now the format of date of journey is good for further preprocessing.

```
# converting object data type to datetime object.
df['Journey_Day'] = pd.to_datetime(df.Date_of_Journey, format="%d/%m/%Y").dt.day
df['Journey_Month'] = pd.to_datetime(df["Date_of_Journey"], format="%d/%m/%Y").dt.month
```

```
# dropping the original column
df=df.drop(columns='Date_of_Journey')
```

- Now splitting the date of journey feature into month and day new features for visualization purpose with the target variable by using pandas to_datetime() method. Then dropping the original column.

```
# Extracting the hours and minutes from departure time column and afterwards we will drop the original column.
df["Dep_hour"] = pd.to_datetime(df['Departure Time']).dt.hour
df['Dep_min'] = pd.to_datetime(df['Departure Time']).dt.minute
df.drop(columns=['Departure Time'],inplace=True)
```

```
# Extracting the hours and minutes from Arrival time column and afterwards we will drop the original column.
df["Arrival_hour"] = pd.to_datetime(df['Arrival Time']).dt.hour
df['Arrival_min'] = pd.to_datetime(df['Arrival Time']).dt.minute
df.drop(columns=['Arrival Time'],inplace=True)
```

- Using the same pandas to_datetime() method for departure time and arrival time features, splitting them into hour and minutes new features.

```
# Feature engineering on Duration column

Duration = list(df['Duration'])

for i in range(len(Duration)):
    if len(Duration[i].split()) != 2:
        if "h" in Duration[i]:
            Duration[i] = Duration[i].strip() + " 0m"
        else:
            Duration[i] = "0h " + Duration[i]

Duration_hours = []
Duration_mins = []

for i in range(len(Duration)):
    Duration_hours.append(int(Duration[i].split(sep='h')[0]))
    Duration_mins.append(int(Duration[i].split(sep='m')[0].split()[-1]))
```

- Doing feature engineering on duration column and stripping them into hours and minutes for data visualization.

```
# Storing the hours and minutes into dataframe and afterwards we will remove the original column.
df['Duration_hours'] = Duration_hours
df['Duration_mins'] = Duration_mins
df.drop(columns=['Duration'],inplace=True)
```

- Storing hours and minutes values into separate dataframes.

```
# categorizing hours into times of day
def tod(x):                    # defining function times of day as tod
    if (x > 4) and (x <= 8):
        return 'Early Morning'
    elif (x > 8) and (x <= 12 ):
        return 'Morning'
    elif (x > 12) and (x <= 16):
        return'Noon'
    elif (x > 16) and (x <= 20) :
        return 'Evening'
    elif (x > 20) and (x <= 24):
        return'Night'
    elif (x <= 4):
        return'Late Night'
```

```
# applying the condition
df['Arrival_Times_of_day']=df['Arrival_hour'].apply(tod)
df.drop(['Arrival_hour'],axis=1,inplace=True)
```

```
# checking after categorizing
df['Arrival_Times_of_day'].unique()
```

```
array(['Noon', 'Evening', 'Morning', 'Early Morning', 'Night',
       'Late Night'], dtype=object)
```

- Using if-else condition for arrival hour feature and then categorizing it into various times of day as stated above.
- This will help to identify the price of ticket is costlier and cheaper in which times of day.

```
# to know all type of unique values present in the column
df['Journey_Month'].unique()
```

```
array([10, 11, 12], dtype=int64)
```

```
# categorizing months into object
def month(x):                    # defining function Journey month as month
    if (x==10):
        return 'October'
    elif (x==11 ):
        return 'November'
    elif (x==12):
        return'December'
```

```
# applying the condition
df['Journey Month']=df['Journey_Month'].apply(month)
df.drop(['Journey_Month'],axis=1,inplace=True)
```

```
# checking after categorizing
df['Journey Month'].unique()
```

```
array(['October', 'November', 'December'], dtype=object)
```

- Converting integer values of journey month into object values for predicting price of ticket is high and low in which upcoming months by using visualization tools.

```
# converting objects into integers of data
lab_enc = LabelEncoder()
list1 = ['Airline', 'Source', 'Destination','Total Stops','Arrival_Times_of_day','Journey Month']
for val in list1:
    df[val] = lab_enc.fit_transform(df[val].astype(str))
```

- Using label encoder to convert all the features having object datatype values into integer values (i.e machine language) for model building.

HARDWARE / SOFTWARE REQUIREMENTS AND TOOLS USED

- Anaconda Navigator 1.10.0
- Jupyter Notebook 6.1.4 , Python 3

Libraries

```
import pandas as pd # for handling dataset
import numpy as np  # for mathematical computation

from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split,GridSearchCV,cross_val_score
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report,
                            plot_roc_curve, roc_auc_score
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.svm import SVC
from sklearn.linear_model import SGDClassifier

from sklearn.decomposition import PCA
from scipy.stats import skew

# for visualization
import matplotlib.pyplot as plt
%matplotlib inline
from matplotlib import style
import seaborn as sns
import pickle

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

Libraries and Packages used:

- Pickle for saving & loading machine learning model.
- GridSearchCV for Hyper-parameter tuning.
- Cross validation score to cross check if the model is overfitting or not.
- Seaborn and matplotlib for visualization.
- Label encoder for converting object datatype into integer datatype.

MODEL/s DEVELOPMENT AND EVALUATION

IDENTIFICATION OF POSSIBLE PROBLEM-SOLVING APPROACHES (METHODS)

As we have to predict price of flight ticket with variables containing maximum object datatype values but our target variable was of continuous data. Thus it's a regression problem and we have to apply regression algorithms and build a model giving good performance.

Dropped original features like date of journey, arrival time, departure time and splitted them into hours, minutes and days for data visualization and comparing it with target variable (price) to understand the dynamic ranging of price of flight ticket through seaborn, matplotlib visualization tools.

Plotted heatmap to visualize which variables were showing good positive or negative correlation with the target variable. So that it could help us in further prediction analysis.

Checked for outliers, treated them with 1.5 IQR method and for skewness applied log transformation for improving accuracy.

Checked for multicollinearity by using vif (Variance Inflation Factor) method .

TESTING OF IDENTIFIED APPROACHES (ALGORITHMS)

Algorithms used:

- Linear Regression
- Decision Tree Regressor
- Random Forest Regressor
- KNN Regressor
- Gradient Boosting Regressor
- XGB Regressor

RUN AND EVALUATE SELECTED MODELS

1. Linear Regression

```
regression = LinearRegression()
regression.fit(x_train,y_train)
```

```
LinearRegression()
```

```
# Adjusted R2 score
regression.score(x_train,y_train)
```

```
0.3977709722263013
```

```
# To cross verify
y_pred = regression.predict(x_test)
```

```
r2_score(y_test,y_pred)
```

```
0.3904844364494573
```

Training Result : 39.77% Test Result : 39.04%

2. Decision Tree Regressor

```
dt_reg = DecisionTreeRegressor()
dt_reg.fit(x_train,y_train)

print("\n===================Train Result=====================")

print(f"Accuracy Score: {dt_reg.score(x_train,y_train) * 100:.2f}%")
print("_____________________________________________")

#********************** Test Score ************************

y_pred = dt_reg.predict(x_test)
print("\n===================Test Result=====================")
print(f"Accuracy Score: {dt_reg.score(x_test,y_test) * 100:.2f}%")
print("_____________________________________________")
```

Training Result : 99.95% Test Result : 39.53%

3. Random Forest Regressor

```
rand_reg = RandomForestRegressor(n_estimators = 100,random_state=51)
rand_reg.fit(x_train,y_train)

print("\n===================Train Result=====================")

print(f"Accuracy Score: {rand_reg.score(x_train,y_train) * 100:.2f}%")
print("_____________________________________________")

#********************** Test Score ************************

y_pred = rand_reg.predict(x_test)
print("\n===================Test Result=====================")
print(f"Accuracy Score: {rand_reg.score(x_test,y_test) * 100:.2f}%")
print("_____________________________________________")
```

Training Result : 95.11% Test Result : 68.96%

4. KNN Regressor

```
knn=KNeighborsRegressor(n_neighbors=3)
knn.fit(x_train,y_train)

print("\n==================Train Result=====================")

print(f"Accuracy Score: {knn.score(x_train,y_train) * 100:.2f}%")
print("_____________________________________________")

#********************** Test Score ************************

y_pred = knn.predict(x_test)
print("\n==================Test Result=====================")
print(f"Accuracy Score: {knn.score(x_test,y_test) * 100:.2f}%")
print("_____________________________________________")
```

Training Result : 71.69% Test Result : 46.29%

5. Gradient Boosting Regressor

```
from sklearn.ensemble import GradientBoostingRegressor
np.random.seed(42)
gbr = GradientBoostingRegressor()
gbr.fit(x_train, y_train)

print("\n==================Train Result=====================")

print(f"Accuracy Score: {gbr.score(x_train,y_train) * 100:.2f}%")
print("_____________________________________________")

#********************** Test Score ************************

y_pred = gbr.predict(x_test)
print("\n==================Test Result=====================")
print(f"Accuracy Score: {gbr.score(x_test,y_test) * 100:.2f}%")
print("_____________________________________________")
```

Training Result : 71.02% Test Result : 64.02%

6. XGB Regressor

```
from xgboost.sklearn import XGBRegressor
np.random.seed(42)
xgb = XGBRegressor()
xgb.fit(x_train, y_train)

print("\n==================Train Result=====================")

print(f"Accuracy Score: {xgb.score(x_train,y_train) * 100:.2f}%")
print("_____________________________________________")

#********************** Test Score ************************

y_pred = xgb.predict(x_test)
print("\n==================Test Result=====================")
print(f"Accuracy Score: {xgb.score(x_test,y_test) * 100:.2f}%")
print("_____________________________________________")
```

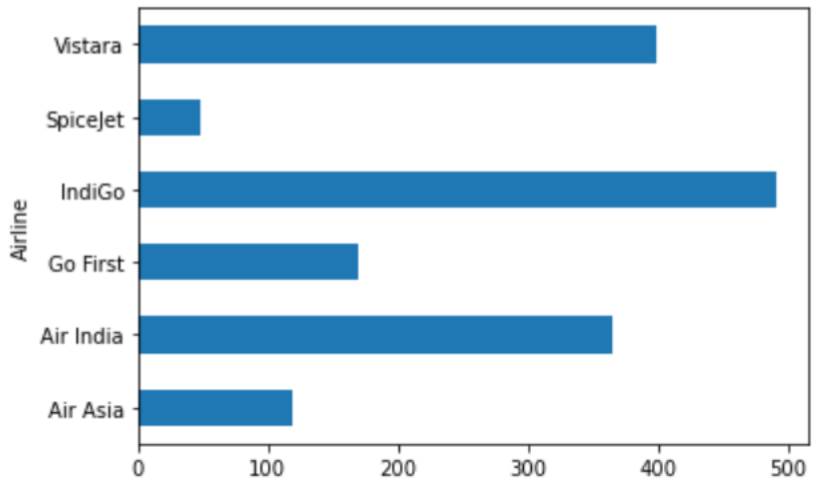
Training Result : 99.66% Test Result : 65.65%

KEY METRICS FOR SUCCESS IN SOLVING PROBLEM UNDER CONSIDERATION

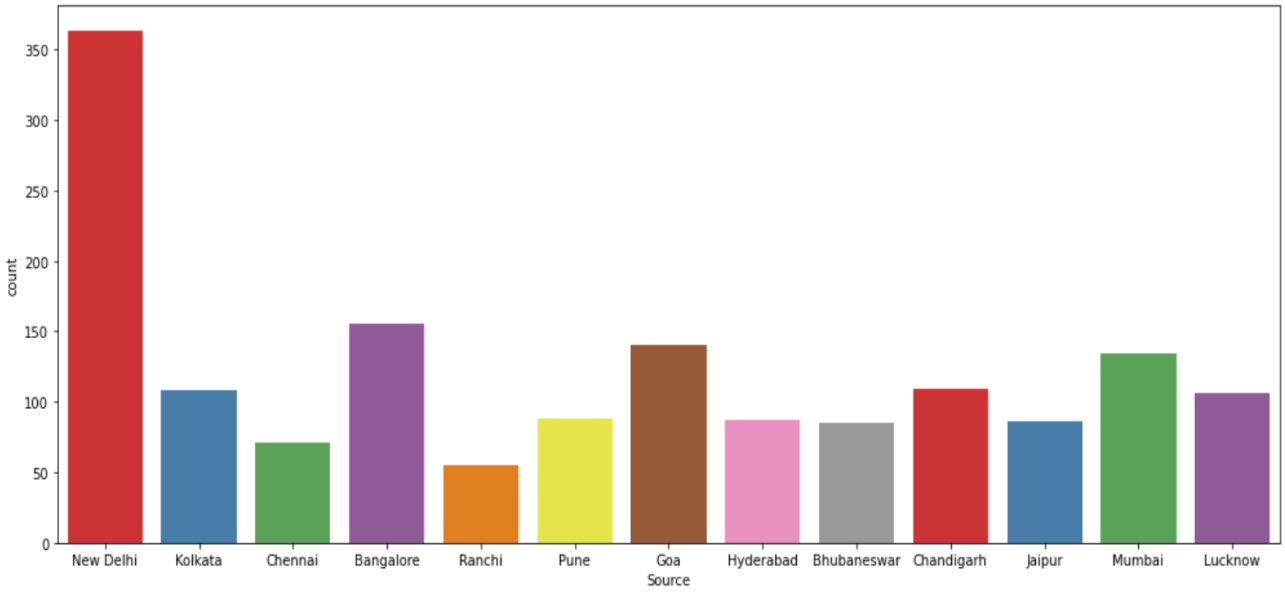
Accuracy score : It is the ratio of the total number of correct predictions and the total number of predictions. It is used when the True Positives and True negatives are more important. Accuracy can be used when the class distribution is similar

Cross_val_score : To run cross-validation on multiple metrics and also to return train scores, fit times and score times. Get predictions from each split of cross- validation for diagnostic purposes. Make a scorer from a performance metric or loss function.

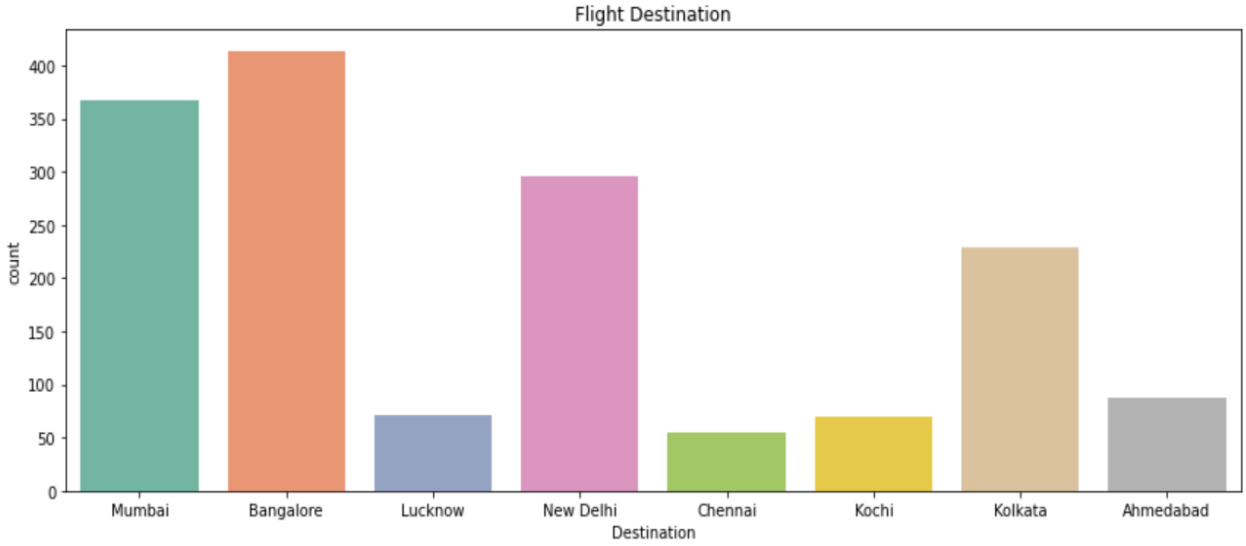
VISUALIZATION



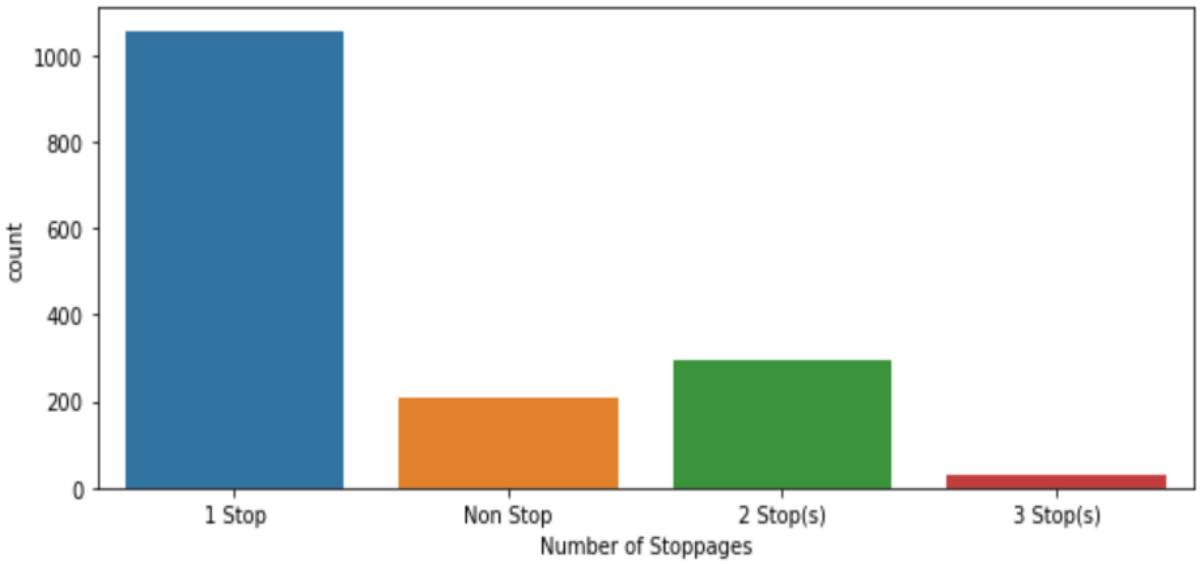
- Indigo flights will be available more from Oct-Dec month and Spicejet flights would be less available and Indigo flights would be more available for passengers.



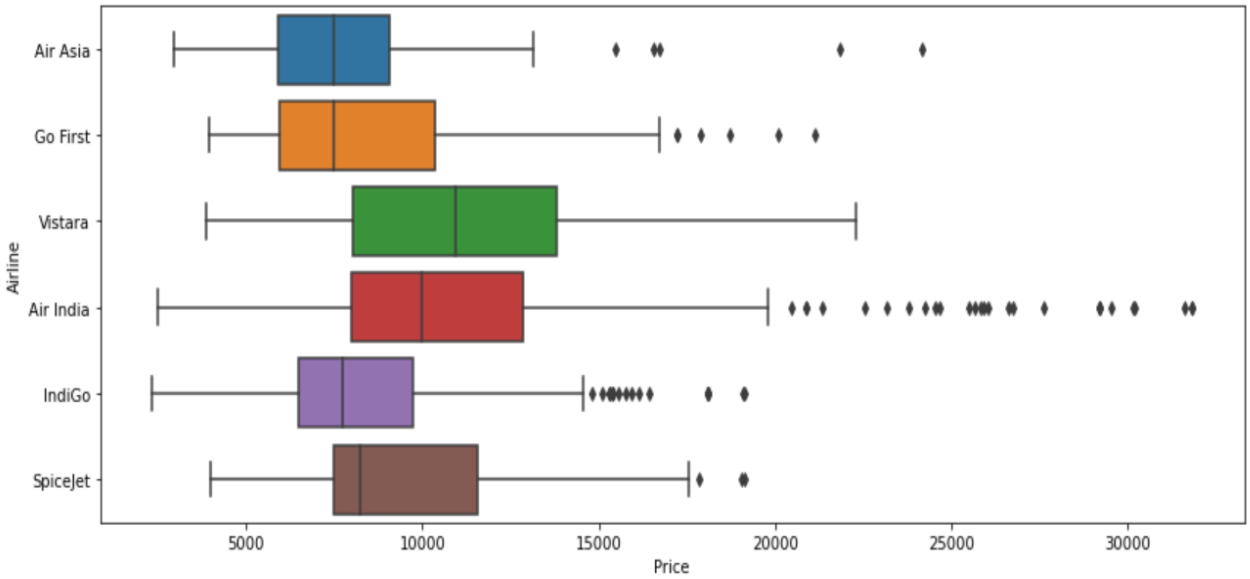
- Mostly flights will take off from New Delhi and least from Ranchi in Oct-Dec month according to the data scraped for various routes.



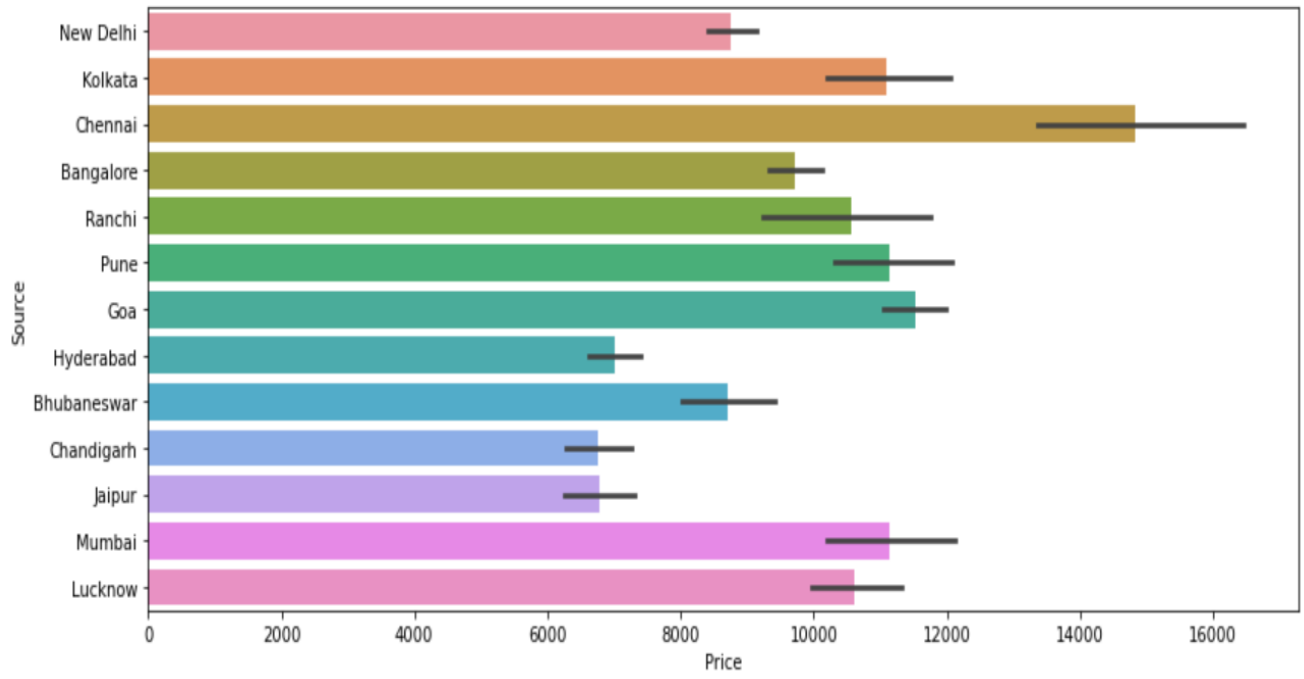
- Maximum flights will land at Bangalore and least flights will land at Chennai.



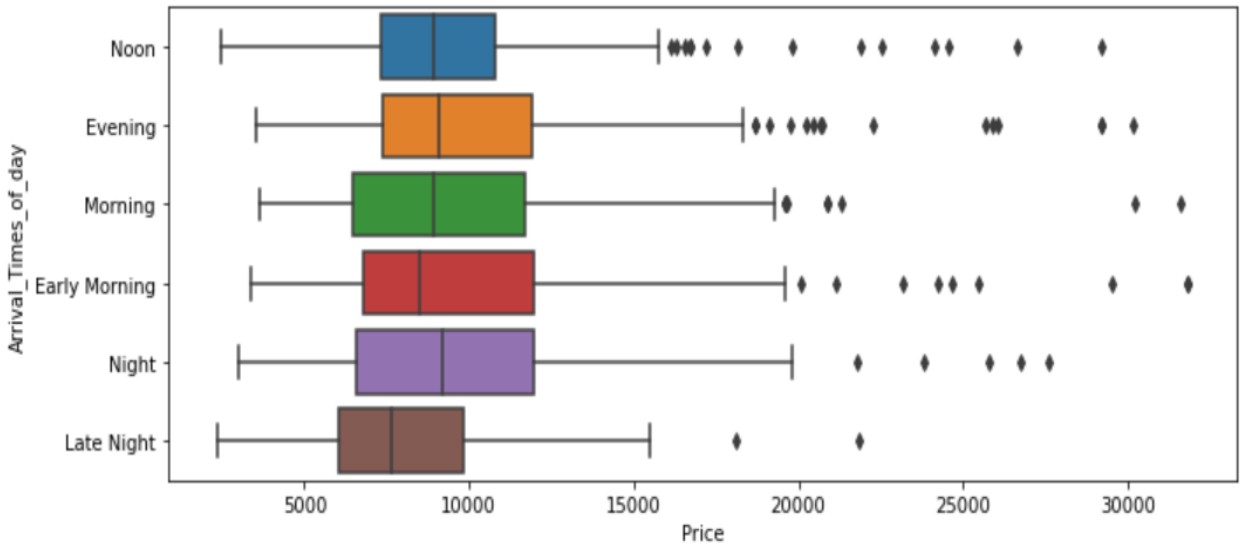
- Maximum number of stoppages is only one for various routes(i.e source-destination).



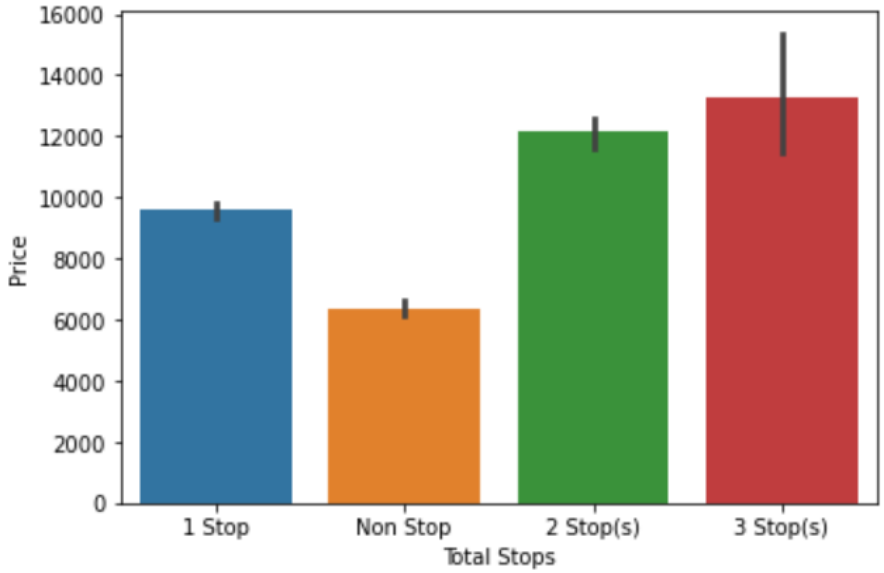
- Air India flight price is costlier than other airlines for month of Oct-Dec 2021.



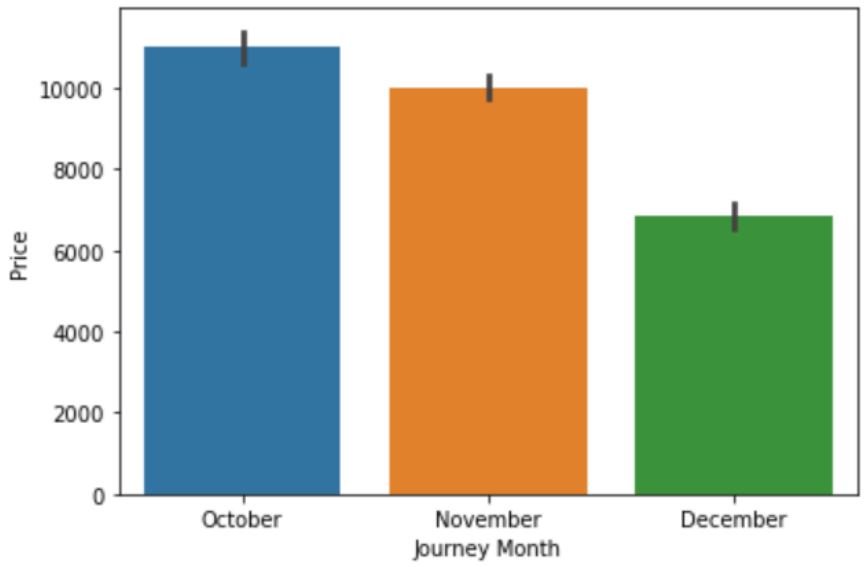
- The price of flight ticket from Chennai is much high than other cities.



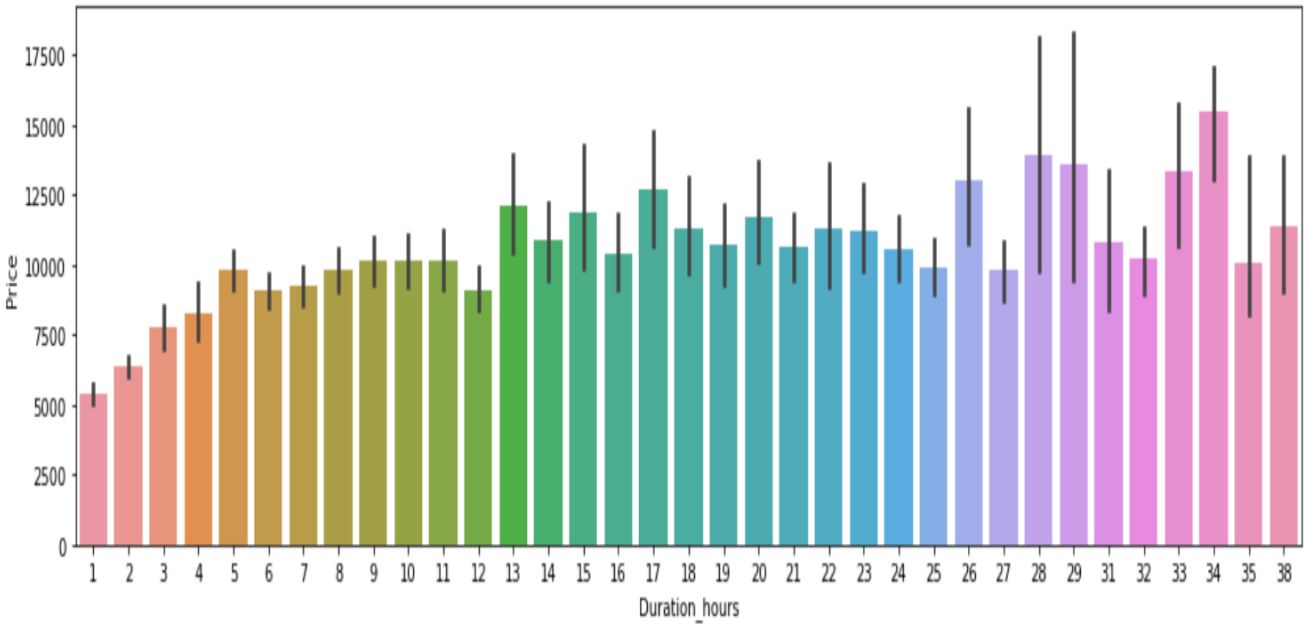
- Price of flight ticket in evening & early morning is costlier and cheap in night when boarding to destination.



- Price of flight ticket increases with increase in number of stoppages.



- Price of flight ticket decreases with increase in month (i.e You will get ticket at low cost price if you book your ticket earlier)



- Maximum the duration of time taken by the flight to reach its destination , higher will be the ticket price.

INTERPRETATION OF RESULTS

- The analysis of heatmap showed that duration hour and journey month are important determinants of target (Price) variable for flight price prediction.
- The airfare varies depending on the time of departure, making timeslot used in analysis an important parameter.
- New Delhi is a hub from where passengers board to various destination.
- Indigo flight price is cheaper and Air India flight price is costlier than other airlines.
- Flight price of various airlines is cheaper if booked earlier (i.e 1 or 2 months before).
- Price of flight ticket increases with increase in number of stoppages.
- Morning and evening flights are expensive compared to night flights.
- Airfares changes frequently and always price tends to go up overtime.
- Flight price increases drastically when heading towards the departure date.

CONCLUSION

KEY FINDINGS AND CONCLUSIONS OF THE STUDY

From the whole project evaluation these are the inferences that I could draw from the visualization of data.

- Economy class in flights is like a boon for middle class people to enjoy their journey by flights because business class tickets price is way much costlier than economy class.
- Travellers/Passengers can plan their journey before months and book flight tickets at reasonable price to avoid high costs.
- You have to pay more if you book ticket at last moment of your date of journey.
- Indigo flight price is cheaper and Air India flight price is costlier than other airlines.
- Airfare varies according to the day of the week of travel. It is higher for weekends and Monday and slightly lower for the other days.
- Only Duration hours feature was having outliers when visualized through boxplot.
- Longer the duration of journey more will be the price of flight tickets.

LEARNING OUTCOMES OF THE STUDY IN RESPECT OF DATA SCIENCE

Through Visualization it was clear that the target variable (price) of various flights gives the information about the highs and lows in the airfares according to the days, weekend and time of the day that is morning, evening and night.

Also the features that depicted a lot of story telling when visualization was done for all of them. Visualization gives meaning to a data which helps drawing inference from it.

Gradient Boosting Regression was the best model to be deployed in production because its accuracy and CV value was least among all models.

The most difficult yet most interesting aspect of the project was to understand the relationship between the flight price and other various features and the performance of various machine learning algorithms.

The challenges that I faced while working on this project was that the dataset scraped by me was small with limited features, so did not get good accuracy as expected. But still we have to build a good model to maximize revenue of airline industries.

LIMITATIONS OF THIS WORK AND SCOPE FOR FUTURE WORK

For further improvement of results, more features such as additional info (i.e free meal, discount coupons etc), routes, the available seats and whether the departure day is a holiday or not, can be added to improve the performance of the model.

Scraped data for only specific dates which were very few as various airline websites did not have that interface to scrape details of flight prices for whole dates of month sequentially.