Flight Price Prediction Project

Submitted by:

HINAL SETH

ACKNOWLEDGMENT

I would like to thank Flip Robo Technologies, for giving me this golden opportunity to work on this valuable project. I got to learn a lot from this project about Data Scraping, Data Wrangling, and practical implementations of using machine learning modules.

I take this opportunity to express my profound gratitude and deep regards to my mentor Ms. Swati Mahaseth for her exemplary guidance, monitoring and constant encouragement throughout the course of this assignment. The blessing, help and guidance given by her time to time shall carry me a long way in the journey of life on which I am about to embark.

Lastly, I thank almighty, my parents, brother, sister and friends for their constant encouragement without which this assignment would not be possible.

References:

1. https://www.easemytrip.com/
2. https://www.yatra.com/

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.

  Anyone who has booked a flight ticket knows how unexpectedly the prices vary. Airlines use using sophisticated quasi-academic tactics which they call **"revenue management"** or **"yield management"**. 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

  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.

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

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

- Review of Literature

The airline implements dynamic pricing for the flight ticket. According to the survey, flight ticket prices change during the morning and evening time of the day. Also, it changes with the holidays or festival season. There are several different factors on which the price of the flight ticket depends. The seller has information about all the factors, but buyers are able to access limited information only which is not enough to predict the airfare prices. Considering the features such as departure time, the number of days left for departure and time of the day it will give the best time to buy the ticket. The purpose of the paper is to study the factors which influence the fluctuations in the airfare prices and how they are related to the change in the prices. Then using this information, build a system that can help buyers whether to buy a ticket or not.

It is very difficult for the customer to purchase a flight ticket at the minimum price. For this several techniques are used to obtain the day at which the price of air ticket will be minimum. Most of these techniques are using sophisticated artificial intelligence(AI) research, also known as Machine Learning.

Utilizing AI models to acquire the greatest presentation to get the least cost of aircraft ticket buying, having 85.3% precision. I contemplated the exhibition of Ridge Regression , Elastic Net, SGD Regression, Decision Tree Regression, K-Neighbors Regression, Random Forest Regression and Gradient Boosting Regression models in anticipating air ticket costs.

The data was collected from major travel journey booking website yatra.com. Additional data were also collected and are used to check the comparisons of the performances of the final model.

- Motivation for the Problem Undertaken

Objective – To Scrape the data from website (I have scrapped the data from yatra.com) and then build a machine learning model to predict the price of the flights.

Analytical Problem Framing

- Data Sources and their formats

  Data Collection is one of the most important aspect of this project. There are various sources of airfare data on the Web, which I could use to train our models. A multitude of consumer travel sites supply fare information for multiple routes, times, and airlines.
  I tried various sources ranging from many APIs to scraping consumer travel websites like yatra.com .

	Airline	Source	Destination	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price (in ₹)
0	Air India	New Delhi	Mumbai	07:00	09:00	2h 00m	Non Stop	Free Meal	2476
1	Air India	New Delhi	Mumbai	09:00	11:15	2h 15m	Non Stop	Free Meal	2476
2	Air India	New Delhi	Mumbai	15:05	18:10	3h 05m	1 stop	Free Meal	2791
3	Air India	New Delhi	Mumbai	17:50	21:35	3h 45m	1 stop	Free Meal	2791
4	Jet Airways	New Delhi	Mumbai	07:55	10:15	2h 20m	Non Stop	Free Meal	3173
...	...	...	...	...	...	...	...	...	...
1545	Air India	Bangalore	New Delhi	08:15	18:30	10h 15m	1 stop	No info	4943
1546	Air India	Bangalore	New Delhi	00:30	23:55	23h 25m	1 stop	No info	10394
1547	Jet Airways	Bangalore	New Delhi	11:40	21:20	9h 40m	1 stop	1 Long layover	27992
1548	IndiGo	Bangalore	New Delhi	18:25	21:20	2h 55m	non-stop	No info	7648
1549	IndiGo	Bangalore	New Delhi	10:05	13:00	2h 55m	non-stop	No info	7648

1550 rows × 9 columns

- Data Preprocessing Done

  A few basic cleaning and feature engineering looking at the data. A lot of data preparation needs to be done according to the model and strategy we use, but here are the basic cleaning I did initially to understand the data better.
  There were not many, but a few repetitions in the data collected. There were no missing values in the datset and since all the feature variables are of object data type, we need not check for it’s skewness, outliers or distribution.

- Data Inputs- Logic- Output Relationships

We can see this graphically that Jet Airways are more costly and Spice Jet are very affordable. Also the flights that provide free meal and additional facilities are more costly as compared to the direct flights.

- ## Hardware and Software Requirements and Tools Used

  Python was the most popular technology for implementing machine learning ideas, owing to the fact that it has a large number of built-in algorithms in the form of bundled libraries. The following are some of the most important libraries and tools we used in our project :

  1. Numpy:
     NumPy is a Python module for array processing. It includes a high-performance multidimensional array object as well as utilities for manipulating them. It is the most important Python module for scientific computing. NumPy may be used as a multi-dimensional container of general data in addition to its apparent scientific applications.
     Numpy allows any data types to be created, allowing NumPy to connect with a broad range of databases cleanly and quickly.

  2. SciPy:
     SciPy is a Python library for scientific and technical computing that is free and open-source. Optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers, and other activities used in research and engineering are all covered by SciPy modules.
     SciPy is based on the NumPy array object, and it's part of the NumPy stack, which also contains Matplotlib, pandas, and SymPy, as well as a growing number of scientific computing libraries. Other apps with comparable users to NumPy include MATLAB, GNU Octave, and Scilab.
     The SciPy stack is occasionally used interchangeably with the NumPy stack. The SciPy library is now available under the BSD licence, with an open community of developers sponsoring and supporting its development.

3. Scikit-Learn
   Scikit-learn offers a standard Python interface for a variety of supervised and unsupervised learning techniques. It is provided under several Linux distributions and is licenced under a liberal simplified BSD licence, promoting academic and commercial use. The library is being constructed.

4. Jupyter Notebook
   Jupyter Notebook is an open-source online software that lets you create and share documents with live code, equations, visualisations, and narrative prose. Data cleansing and transformation, numerical simulation, statistical modelling, data visualisation, machine learning, and more are all included.
   Jupyter Notebook is an open-source online software that lets you create and share documents with live code, equations, visualisations, and narrative text. Data cleansing and transformation, numerical simulation, statistical modelling, data visualisation, machine learning, and more are all included.

5. Chromedriver
   Chromedriver is used to web scrape the data and automate the process. I have scraped the data using Selenium python.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  We go over many techniques and datasets that were used to create this module.
  The model will be trained using a dataset comprising over 1500 tuples. The price of a flight is determined by factors such as the number of stops, duration, the kind of facilities provided, etc. I created regressor methods and compared all on different flight models because this is a regression problem.
  Anaconda seeks to address Python's dependency well, where distinct projects have various dependency versions, so that project dependencies do not require separate versions, which might conflict.

- Testing of Identified Approaches (Algorithms)

  The models used training and testing datasets are as followed:

- Ridge Regression
- Elastic Net Regression
- SGD Regressor
- KNeighbors Regressor
- Decision Tree Regressor
- Random Forest Regressor
- Gradient Boosting Regression

- Run and Evaluate selected models

  1. Ridge Regression: It is most commonly used algorithm to approximate an answer for an equation with no unique solution.

```
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size = 0.3, random_state = 45 )
ridge = Ridge(alpha = 0.5)
ridge.fit(xtrain, ytrain)
pred_test_r= ridge.predict(xtest)
print('Ridge Regression Score:',ridge.score(xtrain,ytrain))
print('Ridge Regression r2_score:',r2_score(ytest,pred_test_r))
print("Mean squared error of Ridge Regression:",mean_squared_error(ytest,pred_test_r))
print("Root Mean Square error of Ridge Regression:",np.sqrt(mean_squared_error(ytest,pred_test_r)))
```

```
Ridge Regression Score: 0.42102269986725194
Ridge Regression r2_score: 0.4427111921124248
Mean squared error of Ridge Regression: 19719147.34985683
Root Mean Square error of Ridge Regression: 4440.624657619334
```

2. Elastic Net regression: It performs more efficient regularization process as it combines both L1 and L2 approaches.

```
en = ElasticNet(alpha = 0.001)
en.fit(xtrain, ytrain)
pred_test_en= en.predict(xtest)
print('ElasticNet Regression Score:',en.score(xtrain,ytrain))
print('ElasticNet Regression r2_score:',r2_score(ytest,pred_test_en))
print("Mean squared error of ElasticNet Regression:",mean_squared_error(ytest,pred_test_en))
print("Root Mean Square error of ElasticNet Regression:",np.sqrt(mean_squared_error(ytest,pred_test_en)))
```

```
ElasticNet Regression Score: 0.42102268383685204
ElasticNet Regression r2_score: 0.4427119644724037
Mean squared error of ElasticNet Regression: 19719120.020615667
Root Mean Square error of ElasticNet Regression: 4440.621580433945
```

3. SGD Regression: The loss gradient is calculated each sample at a time, and the model is updated along the way using a decreasing strength schedule. SGD stands for Stochastic Gradient Descent (aka learning rate).

   The regularizer is a penalty applied to the loss function that decreases model parameters towards zero using either the squared euclidean norm L2 or the absolute norm L1 or a mix of the two (Elastic Net). The update is trimmed to 0.0 whenever the parameter update passes the 0.0 value due to the regularizer, allowing for the learning of sparse models and online feature selection.

```
sgd=SGDRegressor()
sgd.fit(xtrain,ytrain)
pred_train_sgd=sgd.predict(xtrain)
pred_test_sgd=sgd.predict(xtest)
print('SGD Regressor Score:',sgd.score(xtrain,ytrain))
print('SGD Regressor r2_score:',r2_score(ytest,pred_test_sgd))
print("Mean squared error of SGD Regressor:",mean_squared_error(ytest,pred_test_sgd))
print("Root Mean Square error of SGD Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_sgd)))
```

```
SGD Regressor Score: 0.42073845795608456
SGD Regressor r2_score: 0.44608794440570754
Mean squared error of SGD Regressor: 19599664.103301767
Root Mean Square error of SGD Regressor: 4427.150788408022
```

4. K Neighbors Regression: Algorithm Calculating the average of the numerical goal of the K nearest neighbours is a straightforward implementation of KNN regression. An inverse distance weighted average of the K closest neighbours is another method. The distance functions used in KNN regression are the same as those used in KNN classification.

```
knr = KNeighborsRegressor()
knr.fit(xtrain,ytrain)
pred_train_knr=knr.predict(xtrain)
pred_test_knr=knr.predict(xtest)
print('K Neighbors Regressor Score:',knr.score(xtrain,ytrain))
print('K Neighbors Regressor r2_score:',r2_score(ytest,pred_test_knr))
print("Mean squared error of K Neighbors Regressor:",mean_squared_error(ytest,pred_test_knr))
print("Root Mean Square error of K Neighbors Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_knr)))
```

```
K Neighbors Regressor Score: 0.7602407737247007
K Neighbors Regressor r2_score: 0.6029044870395156
Mean squared error of K Neighbors Regressor: 14050856.2547957
Root Mean Square error of K Neighbors Regressor: 3748.4471791390765
```

5. Decision Tree Regression: To get from observations about an item (represented in the branches) to inferences about the item's goal value, decision tree learning employs a decision tree (as a predictive model) (represented in the leaves). In statistics, data mining, and machine learning, it is one of the predictive modelling methodologies. Classification trees are tree models in which the goal variable can take a discrete set of values; in these tree structures, leaves indicate class labels and branches represent feature combinations that lead to those class labels. Regression trees are decision trees in which the target variable can take continuous values (usually real numbers). The objective is to build a model that predicts the value of a target variable from a set of input variables.

```
dtr=DecisionTreeRegressor(criterion='mse')
dtr.fit(xtrain,ytrain)
pred_train_dtr=dtr.predict(xtrain)
pred_test_dtr=dtr.predict(xtest)
print('Decision Tree Regressor Score:',dtr.score(xtrain,ytrain))
print('Decision Tree Regressor r2_score:',r2_score(ytest,pred_test_dtr))
print("Mean squared error of Decision Tree Regressor:",mean_squared_error(ytest,pred_test_dtr))
print("Root Mean Square error of Decision Tree Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_dtr)))
```

```
Decision Tree Regressor Score: 0.9697230199412826
Decision Tree Regressor r2_score: 0.7646436014054596
Mean squared error of Decision Tree Regressor: 8327867.773281459
Root Mean Square error of Decision Tree Regressor: 2885.8045279057724
```

6. Random Forest Regression: A regressor with a random forest. A random forest is a meta estimator that employs averaging to increase predicted accuracy and control over-fitting by fitting a number of classification decision trees on various sub-samples of the dataset.

```
rf=RandomForestRegressor()
rf.fit(xtrain,ytrain)
pred_train_rf=rf.predict(xtrain)
pred_test_rf=rf.predict(xtest)
print('Random Forest Regressor Score:',rf.score(xtrain,ytrain))
print('Random Forest Regressor r2_score:',r2_score(ytest,pred_test_rf))
print("Mean squared error of Random Forest Regressor:",mean_squared_error(ytest,pred_test_rf))
print("Root Mean Square error of Random Forest Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_rf)))
```

```
Random Forest Regressor Score: 0.942895397322718
Random Forest Regressor r2_score: 0.8195508129194671
Mean squared error of Random Forest Regressor: 6385027.0431426745
Root Mean Square error of Random Forest Regressor: 2526.861104837912
```

r2_score and Score of Random Forest Regressor is very nice .

7. Gradient Boosting Regression: Gradient boosting regressors are a type of **inductively generated tree ensemble model**. At each step, a new tree is trained against the negative gradient of the loss function, which is analogous to (or identical to, in the case of least-squares error) the residual error.

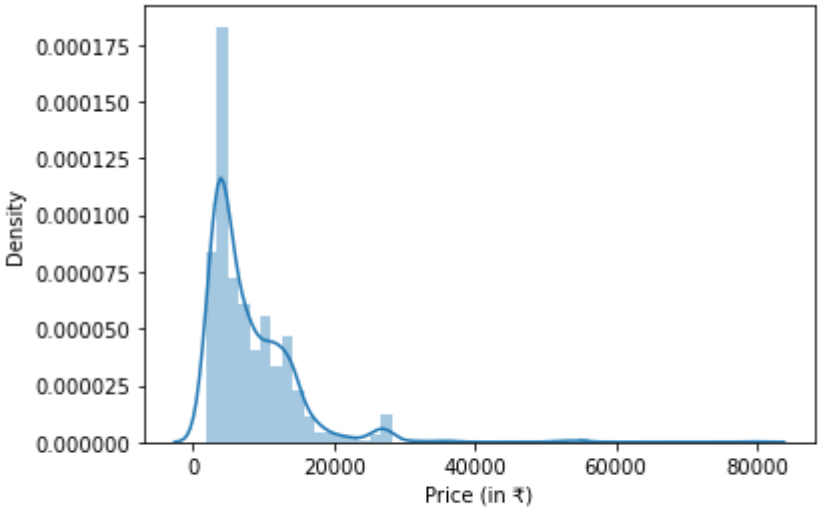
```
gb=GradientBoostingRegressor()
gb.fit(xtrain,ytrain)
pred_train_gb=gb.predict(xtrain)
pred_test_gb=gb.predict(xtest)
print('Gradient Boosting Regressor Score:',gb.score(xtrain,ytrain))
print('Gradient Boosting Regressor r2_score:',r2_score(ytest,pred_test_gb))
print("Mean squared error of Gradient Boosting Regressor:",mean_squared_error(ytest,pred_test_gb))
print("Root Mean Square error of Gradient Boosting Regressor:",np.sqrt(mean_squared_error(ytest,pred_test_gb)))
```

```
Gradient Boosting Regressor Score: 0.8682020013864885
Gradient Boosting Regressor r2_score: 0.7595134586149259
Mean squared error of Gradient Boosting Regressor: 8509393.115582515
Root Mean Square error of Gradient Boosting Regressor: 2917.0864086589063
```

r2_score and Score of Gradient Boosting Regressor is very nice and similar to Random forest Regressor.

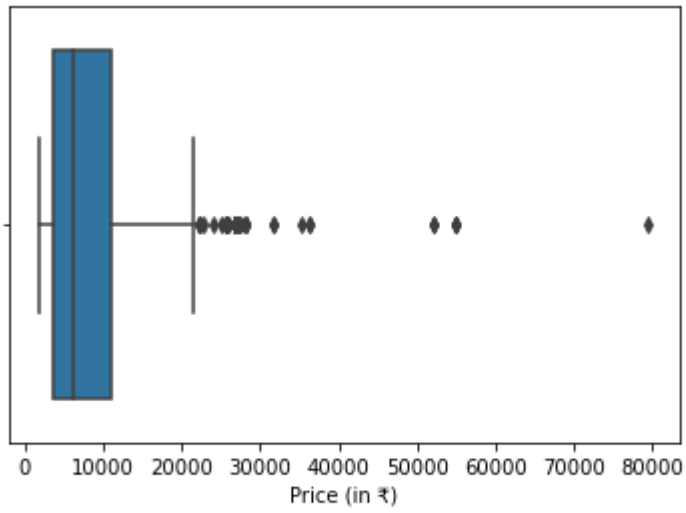
- Visualizations

1. Distribution plot



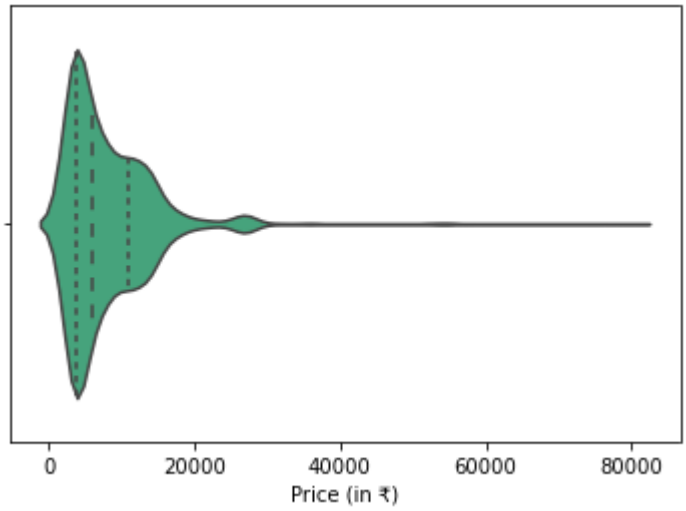
The maximum flights are within ₹ 10000

2. Box plot

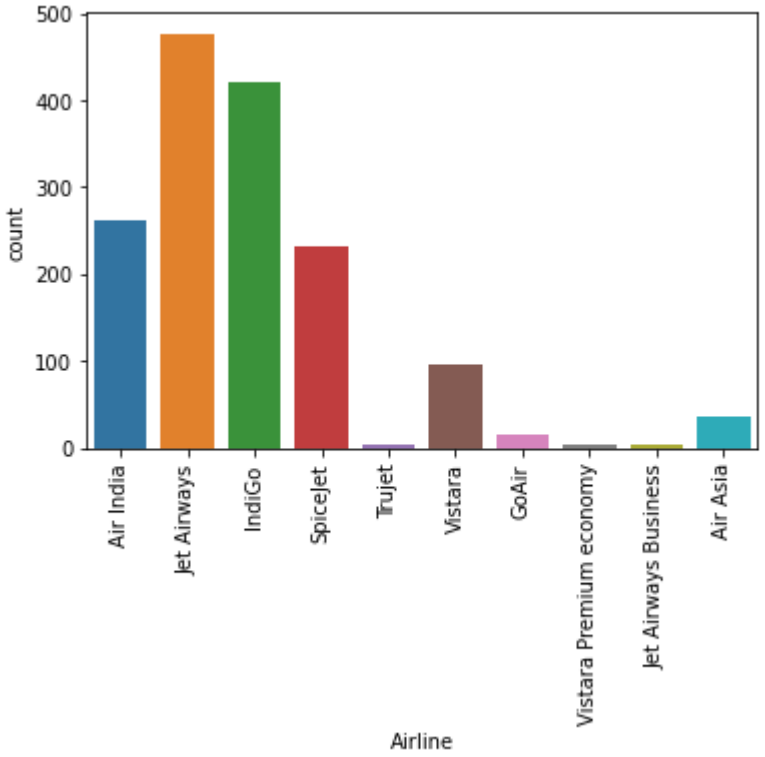


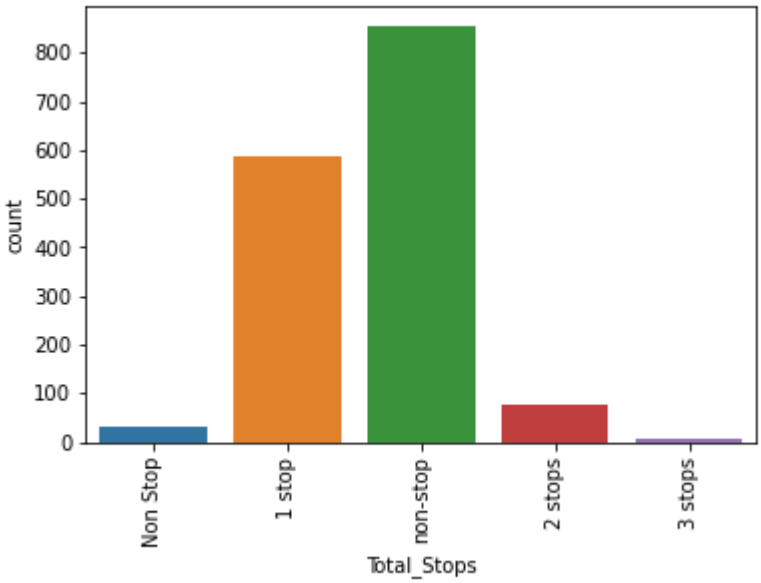
There are few outliers in our target and also it is nearly tightly distributed.

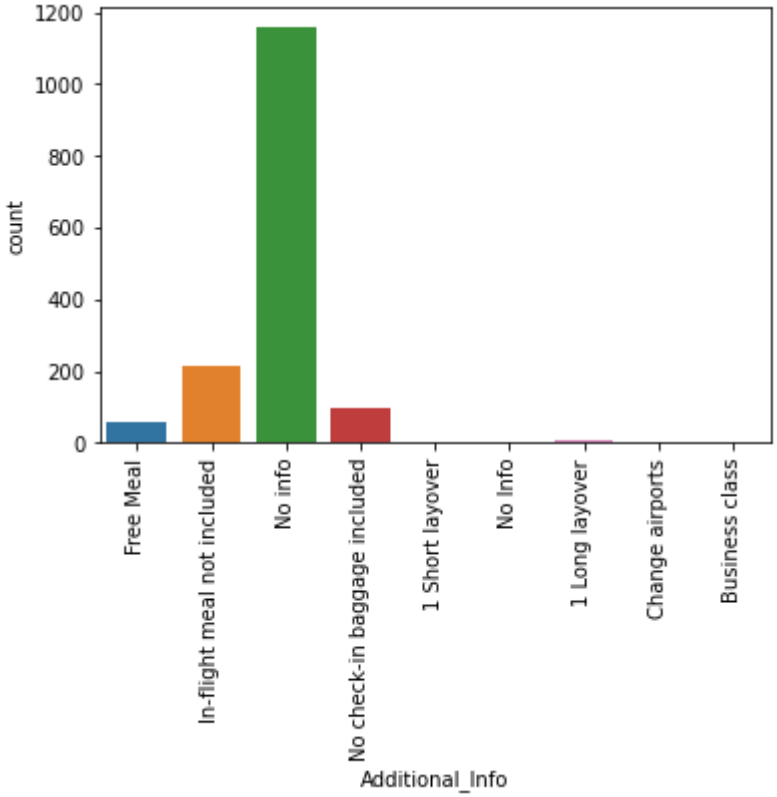
3. Violin plot


0
20000
40000
60000
80000
Price (in ₹)

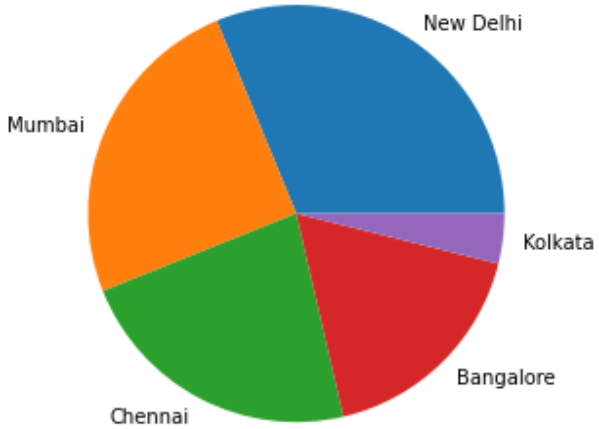
4. Count plot


500
400
300
200
100
0
count
Air India
Jet Airways
IndiGo
Spicejet
Trujet
Vistara
GoAir
Vistara Premium economy
Jet Airways Business
Air Asia
Airline

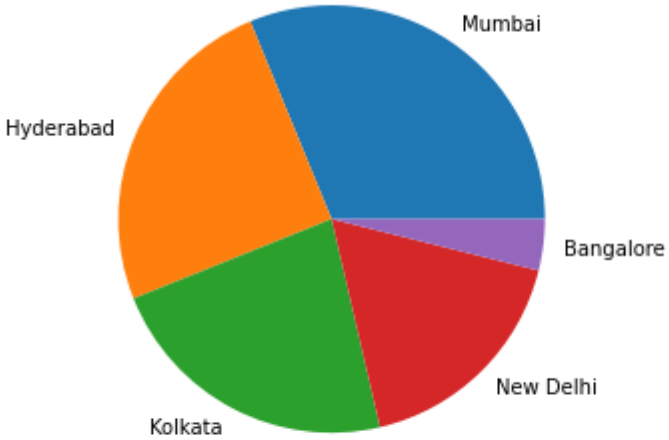

800
700
600
500
400
300
200
100
0
count
Non Stop
1 stop
non-stop
2 stops
3 stops
Total_Stops


1200
1000
800
600
400
200
0
count
Free Meal
In-flight meal not included
No info
No check-in baggage included
1 Short layover
No Info
1 Long layover
Change airports
Business class
Additional_Info

5. Pie plot



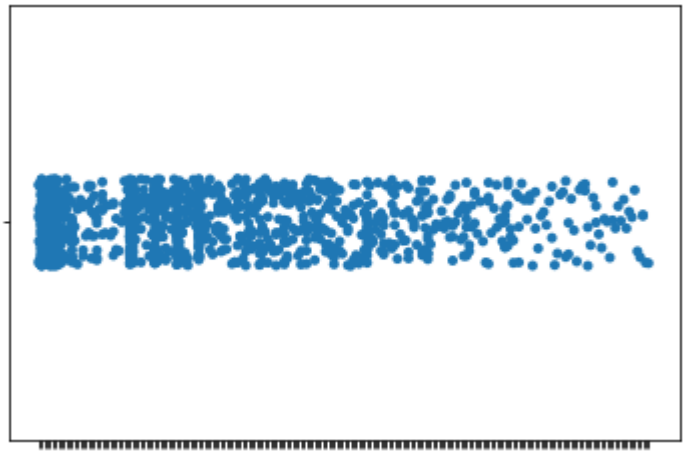
New Delhi and Mumbai have more flights compared to other sources.



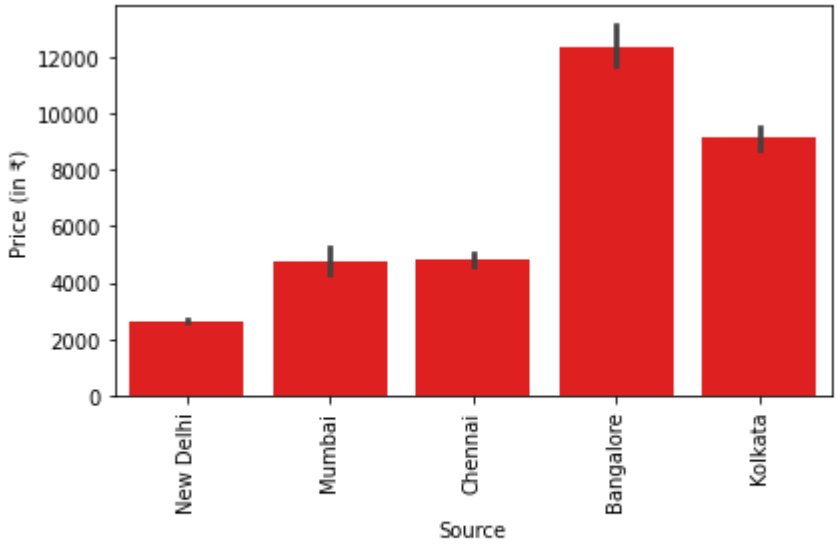
Maximum passengers take a flight to Mumbai.

6. Strip plot

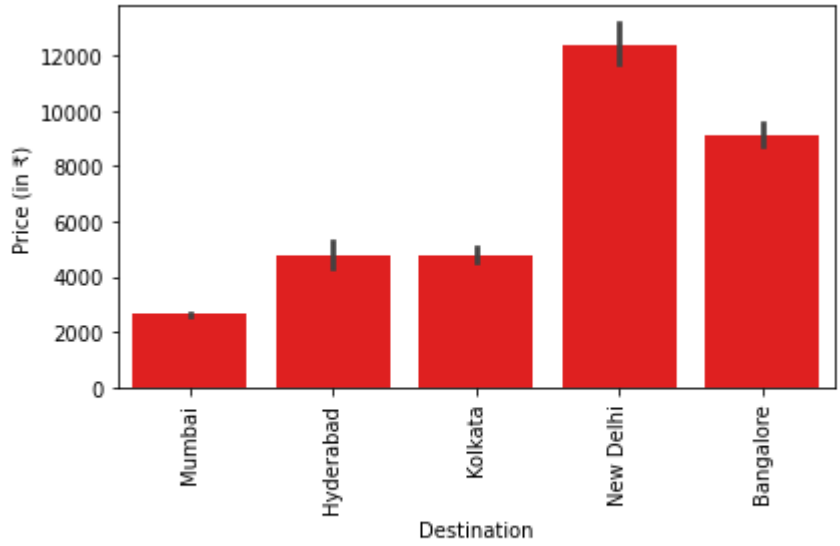
   Reflecting wide range of duration.



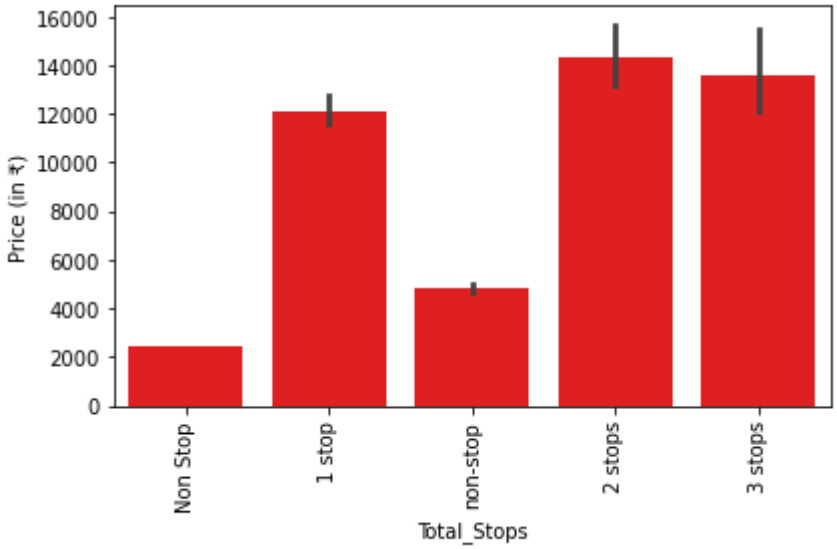
7. Bar plot

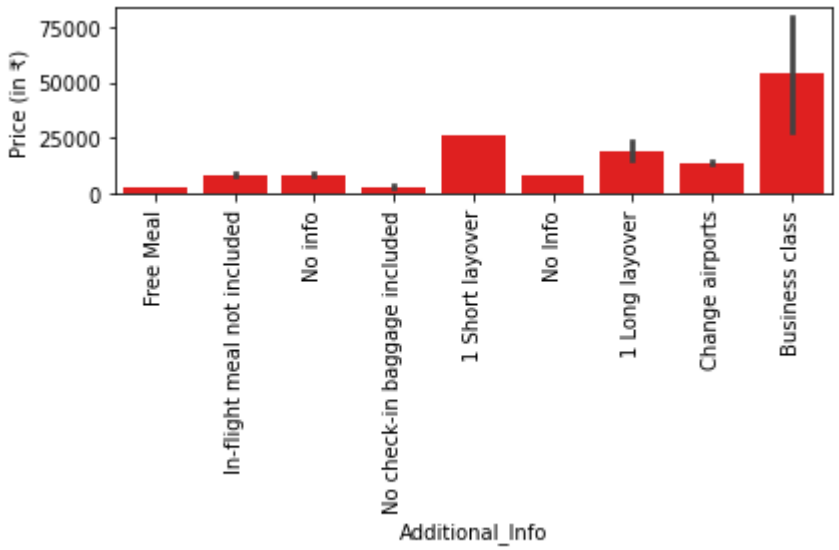


Flights from Bangalore and Kolkata are costly

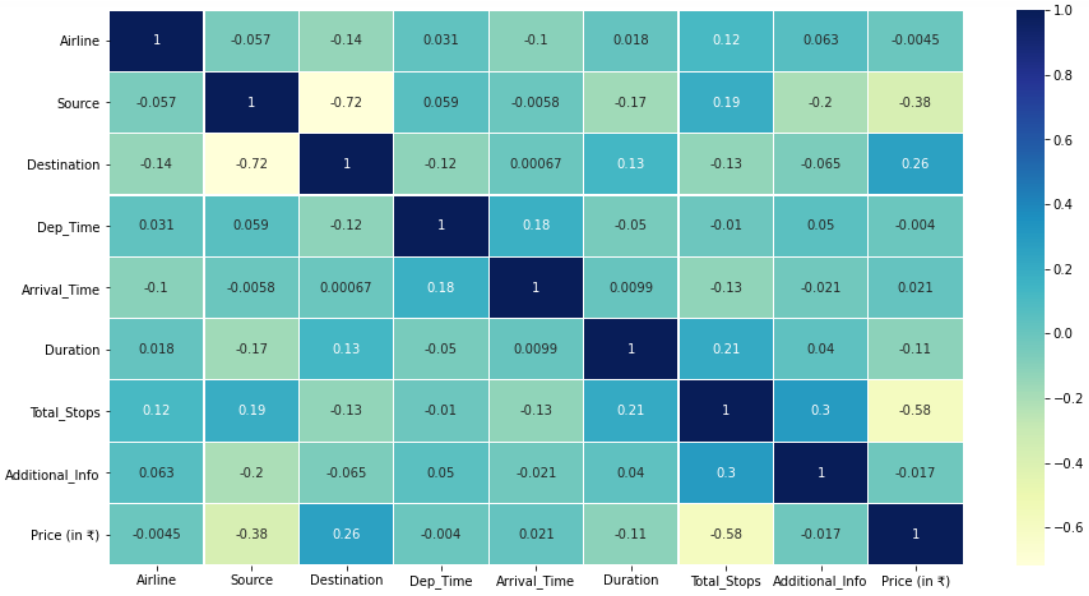


Flights to New Delhi and Bangalore are costly.





8. Heat map



There does not exist any multicollinearity in the dataset.

- ## Interpretation of the Results

  The Jet Airway Airlines are more costly than others whereas SpiceJet and IndiGo are quite affordable. Flights from metro cities are more in number and hence few are in budget and few are way too expensive. The expensive flights usually come with layover(long/short), free meal and some other additional facilities as well.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  The trend of flight prices vary over various months and across the holiday. There are two groups of airlines: the economic 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.

- ## Learning Outcomes of the Study in respect of Data Science

  Collected and analysed data for 6 routes which spanned across business & tourist routes in India

  Some of the routes had non-decreasing prices and thus the model suggested to buy the ticket always

  Implemented algorithms like Decision Tree, Random Forest , Gradient Boosting and statistical analysis

  Developed a basic UI for the model

- ## Limitations of this work and Scope for Future Work

I can also consider days of weeks and months as well to predict the price more efficiently.

To make the prediction accurate, time of booking should also be considered (as in when a person is booking tickets, how many days prior to the journey?, etc.)