Data Science

Internship Assignment Report

Flight Price Optimization

Author:

RANIT SEN

MANSI LONKAR

KRISHNENDU SARKAR

NAIM KHAN

Mentor:

YASIN SHAH

Assignment Time:

April-2021-May-2021

INDICE

Flight Price Optimization Introduction

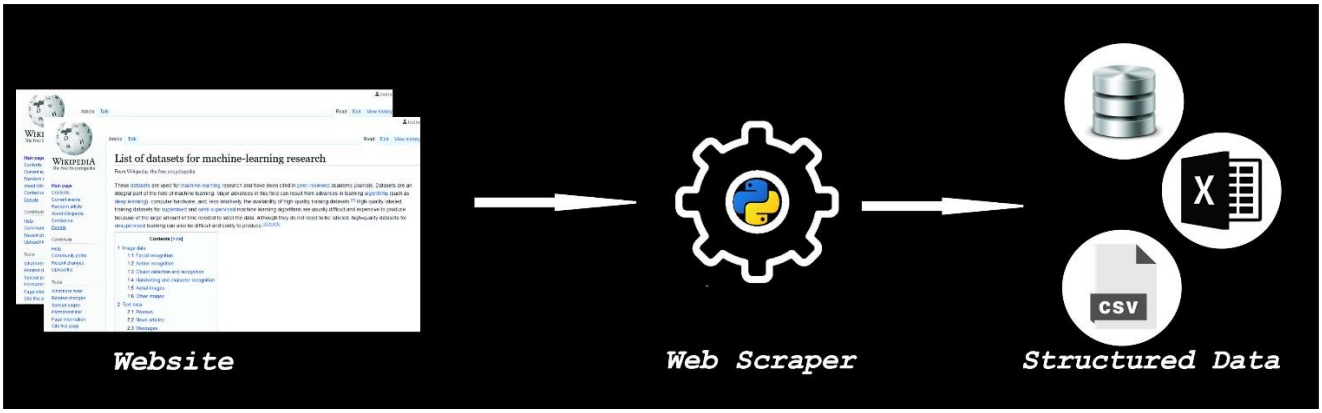
Introduction:

Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. 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. From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price.

Aim of the Assignment:

Airlines employ complex, secretly kept algorithms to vary flight ticket prices over time based on several factors, including seat availability, airline capacity, the price of oil, seasonality, etc. Our aim is to predict the optimal time to buy a ticket with the minimum ticket price. So can we can predict the time when a customer can book a ticket before the travel to get the best price according to the airline.

Web Scrapping and Dataset Building



If we want to obtain large amounts of information from a website as quickly as possible then copy and paste is not going to work. Such as large amounts of data from a website to train a Machine Learning algorithm in such a situation! That is when we'll need to use **Web Scraping**.

Now what is **Web Scrapping**?

Web Scripting is an automatic method to obtain large amounts of data from websites. Most of this data is unstructured data in an HTML format, which is then converted into structured data in a spreadsheet or a database so that it can be used in various applications. There are many different ways to perform web scraping to obtain data from websites. These include using online services, particular API's or even creating your code for web scraping from scratch. Many large websites like Google, Twitter, Facebook, StackOverflow, etc. have API's that allow you to access their data in a structured format. This is the best option but there are other sites that don't allow users to access large amounts of data in a structured form or they are simply not that technologically advanced. In that situation, it's best to use Web Scraping to scrape the website for data.

Web scraping requires two parts namely the **crawler** and the **scraper**. The crawler is an artificial intelligence algorithm that browses the web to search the particular data required by following the links across the internet. The scraper, on the other hand, is a specific tool created to extract the data from the website. The design of the scraper can vary greatly according to the complexity and scope of the project so that it can quickly and accurately extract the data.

In this assignment, we have used '**Python'**, '**Selenium** Package' and '**Chromedriver'**. For the data, we have used popular website 'KAYAK' (https://www.kayak.co.in/) an online travel agency and metasearch engine owned and operated by Booking Holdings.

We work as followed:-

- Connect Python to our web browser and access the Kayak website.
- Choose the Two way ticket type based on our preferences.
- Select Source and Destination city in India and the departure and arrival dates and the return date also.
- Compile the data into one File.

Output of web scraping the kayak website for selected parameters is as shown below:

```
Origin? Pune
Destination? Kolkata
Departure date? Please use YYYY-MM-DD format only 2021-05-24
Return when? Please use YYYY-MM-DD format only 2021-06-30
loading more.....
starting first scrape.....
switching to cheapest results.....
loading more.....
starting second scrape.....
switching to quickest results.....
loading more.....
starting third scrape.....
saved df.....
iteration 0 was complete @ 20210412-1605
sleep finished.....
loading more.....
starting first scrape.....
switching to cheapest results.....
loading more.....
```

After the data collected, for various date to flight and places we merged the dataset in a single dataset in Excel in (.xls) format.

	Out_Date	Out_Day	Out_Weekday	Out_Month	out_Year	Out_Time	Out_Cities	Out_Airline	Return_Date	Return_Day	Return_Weekday	Return_Month
0	6/5/2021	6	Thursday	May	2021	21:50–23:10	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
1	6/5/2021	6	Thursday	May	2021	19:15–14:30	+1 GOI	Air India	26/5/2021	26	Wednesday	May
2	6/5/2021	6	Thursday	May	2021	08:25–09:45	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
3	6/5/2021	6	Thursday	May	2021	21:50–23:10	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
4	6/5/2021	6	Thursday	May	2021	08:25–09:45	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May

This is the structure of our dataset.

Features of Dataset

	Out_Date	Out_Day	Out_Weekday	Out_Month	out_Year	Out_Time	Out_Cities	Out_Airline	Return_Date	Return_Day	Return_Weekday	Return_Month
0	6/5/2021	6	Thursday	May	2021	21:50–23:10	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
1	6/5/2021	6	Thursday	May	2021	19:15–14:30	+1 GOI	Air India	26/5/2021	26	Wednesday	May
2	6/5/2021	6	Thursday	May	2021	08:25–09:45	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
3	6/5/2021	6	Thursday	May	2021	21:50–23:10	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May
4	6/5/2021	6	Thursday	May	2021	08:25–09:45	IXM Madurai	IndiGo	26/5/2021	26	Wednesday	May

This is the structure of our dataset.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1		Out_Date	Out_Day	ut_Weekd	Out_Month	out_Year	Out_Time	Out_Cities	ut_Airline	eturn_Dat	eturn_Da	urn_Week	eturn_Mon	eturn_Yea	eturn_Time	eturn_Citi	eturn_Airli	Travel_Tim	n_Travel	Journey_t	_Journey	t_Stop_Cit	rn_Stop_C	Price	timestamp	sort
2	0	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	17:10–21:1	BLR Beng	IndiGo	1h 20m	4h 00m	direct	1 stop	0	0	6242	20210416-	Best
3	1	6/5/2021	6	Thursday	May	2021	19:15–14:3	+1 GOI	Air India	26/5/2021	26	Wednesda	May	2021	06:30–10:5	BLR Beng	Air India	19h 15m	4h 20m	1 stop	1 stop	MYQ	MYQ	5100	20210416-	Best
4	2	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
5	3	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
6	4	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
7	5	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
8	6	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
9	7	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Best
10	8	6/5/2021	6	Thursday	May	2021	14:40–16:0	GOI Dabol	Vistara	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	6159	20210416-	Best
11	9	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:25–16:3	BLR Beng	AirAsia Inc	1h 00m	1h 10m	direct	direct	0	0	6306	20210416-	Best
12	10	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	05:45–07:0	BLR Beng	AirAsia Inc	1h 00m	1h 15m	direct	direct	0	0	6306	20210416-	Best
13	11	6/5/2021	6	Thursday	May	2021	19:20–20:2	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:25–16:3	BLR Beng	AirAsia Inc	1h 05m	1h 10m	direct	direct	0	0	6306	20210416-	Best
14	12	6/5/2021	6	Thursday	May	2021	19:20–20:2	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	05:45–07:0	BLR Beng	AirAsia Inc	1h 05m	1h 15m	direct	direct	0	0	6306	20210416-	Best
15	13	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Best
16	14	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Best
17	15	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Best
18	16	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:45–16:5	BLR Beng	GoAir	1h 00m	1h 05m	direct	direct	0	0	5619	20210416-	Best
19	17	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	17:10–21:1	BLR Beng	IndiGo	1h 20m	4h 00m	direct	1 stop	0	0	6242	20210416-	Cheap
20	18	6/5/2021	6	Thursday	May	2021	19:15–14:3	+1 GOI	Air India	26/5/2021	26	Wednesda	May	2021	06:30–10:5	BLR Beng	Air India	19h 15m	4h 20m	1 stop	1 stop	MYQ	MYQ	5100	20210416-	Cheap
21	19	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
22	20	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
23	21	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
24	22	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
25	23	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
26	24	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	Cheap
27	25	6/5/2021	6	Thursday	May	2021	14:40–16:0	GOI Dabol	Vistara	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	6159	20210416-	Cheap
28	26	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:25–16:3	BLR Beng	AirAsia Inc	1h 00m	1h 10m	direct	direct	0	0	6306	20210416-	Cheap
29	27	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	05:45–07:0	BLR Beng	AirAsia Inc	1h 00m	1h 15m	direct	direct	0	0	6306	20210416-	Cheap
30	28	6/5/2021	6	Thursday	May	2021	19:20–20:2	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:25–16:3	BLR Beng	AirAsia Inc	1h 05m	1h 10m	direct	direct	0	0	6306	20210416-	Cheap
31	29	6/5/2021	6	Thursday	May	2021	19:20–20:2	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	05:45–07:0	BLR Beng	AirAsia Inc	1h 05m	1h 15m	direct	direct	0	0	6306	20210416-	Cheap
32	30	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Cheap
33	31	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Cheap
34	32	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	5543	20210416-	Cheap
35	33	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:45–16:5	BLR Beng	GoAir	1h 00m	1h 05m	direct	direct	0	0	5619	20210416-	Cheap
36	34	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	17:10–21:1	BLR Beng	IndiGo	1h 20m	4h 00m	direct	1 stop	0	0	6242	20210416-	fast
37	35	6/5/2021	6	Thursday	May	2021	19:15–14:3	+1 GOI	Air India	26/5/2021	26	Wednesda	May	2021	06:30–10:5	BLR Beng	Air India	19h 15m	4h 20m	1 stop	1 stop	MYQ	MYQ	5100	20210416-	fast
38	36	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
39	37	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
40	38	6/5/2021	6	Thursday	May	2021	08:25–09:4	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
41	39	6/5/2021	6	Thursday	May	2021	21:50–23:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
42	40	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	20:05–21:3	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
43	41	6/5/2021	6	Thursday	May	2021	12:50–14:1	IXM Madur	IndiGo	26/5/2021	26	Wednesda	May	2021	06:20–07:4	BLR Beng	IndiGo	1h 20m	1h 25m	direct	direct	0	0	5995	20210416-	fast
44	42	6/5/2021	6	Thursday	May	2021	14:40–16:0	GOI Dabol	Vistara	26/5/2021	26	Wednesda	May	2021	12:20–13:5	BLR Beng	Vistara	1h 20m	1h 30m	direct	direct	0	0	6159	20210416-	fast
45	43	6/5/2021	6	Thursday	May	2021	07:35–08:3	GOI Dabol	AirAsia Inc	26/5/2021	26	Wednesda	May	2021	15:25–16:3	BLR Beng	AirAsia Inc	1h 00m	1h 10m	direct	direct	0	0	6306	20210416-	fast

This is the excel sheet we got after scrapping.

In this part, we will explain the features of it. The dataset consists of 6650 rows and 26 columns. Here we will give details about the 26 attributes.

1. **'Unnamed':** This column just consists of serial number and not need for our analysis.
2. **'Out_Date':** This column consists of the day we are planning to travel means departure day.
3. **'Out_Day':** This is the day number we are travelling. If we planning to travel on 6/05/2021. This column will have the number 6 only.
4. **'Out_Weekday':** This column consists of which weekday we are travelling like 'Sunday' or 'Monday'.
5. **'Out_Month':** This is the month, which we are travelling**.**
6. **'out_Year':** This is the year, which we are travelling**.**
7. **'Out_Time':** The time we are travelling in flight.
8. **'Out_Cities':** From the city we start travelling through flight.
9. **'Out_Airline':** The airline name we are travelling to our destination.
10. **'Return_Date:** The day we are planning the return like 26/05/2021.
11. **'Return_Day:** Only the he day we are return like in this case it will be 26.
12. **'Return_Weekday:** The Weekday we are planning to return.
13. **'Return_Month:** The month we are planning to return.
14. **'Return_Year':** The year we planning to return.
15. **'Return_Time':** The time of we are travelling in the flight in the time of returning the origin city.
16. **'Return_City':** The destination city we are returning in the time of return.
17. **'Return_Airline':** The airline we prefer in the time of returning.
18. **'Out_Travel_Time':** The duration of the travelling time on the time of departure from Origin city.
19. **'Return_Travel_Time':** The duration of the travelling time on the time of arrival to Origin city from destination city.
20. **'Out_Jorney_Type':** The type of journey we are planning to destination city like 'direct', '1 stop', '2 stop' or '3 stop'.
21. **'Return_Jorney_Type':** The type of journey we are planning during return to origin city like 'direct', '1 stop', '2 stop' or '3 stop'.
22. **'Out_Stop_Cities:** The cities we are halting during '1 stop', '2 stop' or '3 stop' flights in the time of journey to destination city.
23. **'Return_Stop_Cities:** The cities we are halting during '1 stop', '2 stop' or '3 stop' flights in the time of journey to origin city.
24. **'Price':** The total price including the return journey (2 way).
25. **'timestamp':** the timestamp of journey duration.
26. **'sort':** The type of sort we prefer for our flight like 'Best', 'Cheap' or 'fast'.

Dataset Analysis

We start with importing many required libraries.

```
In [2]: #Importing Basic Libraries
        import pandas as pd
        import numpy as np
        import pickle
        from time import sleep, strftime
        from random import randint
        import pandas as pd
        from selenium import webdriver
        from selenium.webdriver.common.keys import Keys
        import smtplib
        from email.mime.multipart import MIMEMultipart


        #import warning library
        import warnings
        warnings.filterwarnings("ignore")


        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        sns.set()


        #Importing Miscellaneous Libraries
        pd.set_option("display.max_columns",None)
        pd.set_option("display.max_rows",None)
        pd.set_option('display.width', None)

        # Importing libraries for evaluation of model
        from sklearn.metrics import *
        from math import sqrt

        # Importing libraries for model building
        from sklearn.model_selection import GridSearchCV
        from sklearn.model_selection import RandomizedSearchCV

        from sklearn.neighbors import KNeighborsRegressor
        from sklearn.linear_model import Ridge
        from sklearn.linear_model import Lasso
        from sklearn.tree import DecisionTreeRegressor
        from sklearn.ensemble import RandomForestRegressor
        from xgboost import XGBRegressor
        from sklearn.model_selection import KFold
```

After importing the dataset, we perform basic operation to have an idea about the dataset.

Our dataset have initially 6650 rows and 26 columns.

We used ‘pandas_profiling’ to have a detailed statistical overview over the dataset. We also removed the unnecessary columns and changed the datatypes as needed and we have created timestamp for out time.

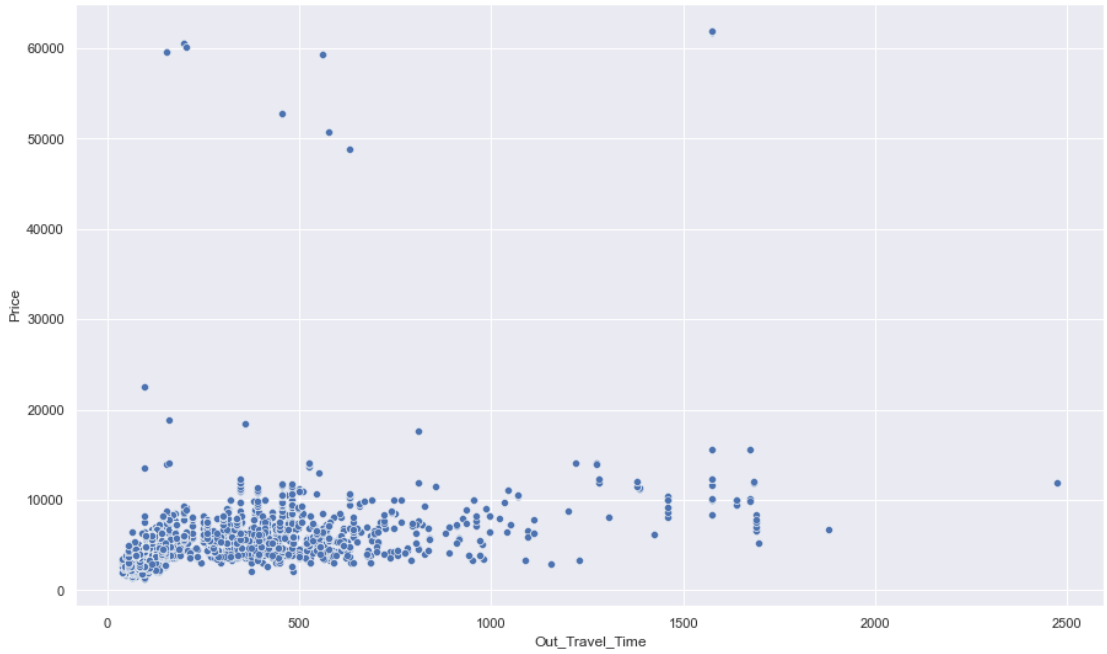
Next we calculate the out price=total price*0.56 as approax calculation. We drop every return columns, as those are not needed.

Exploratory Data Analysis

The main purpose of EDA is to help look at data before making any assumptions. It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

We will explore the data in this stage and try to get as many insights we can find inside them. We will use various graphs and plots in this stage to understand the patters hidden inside.

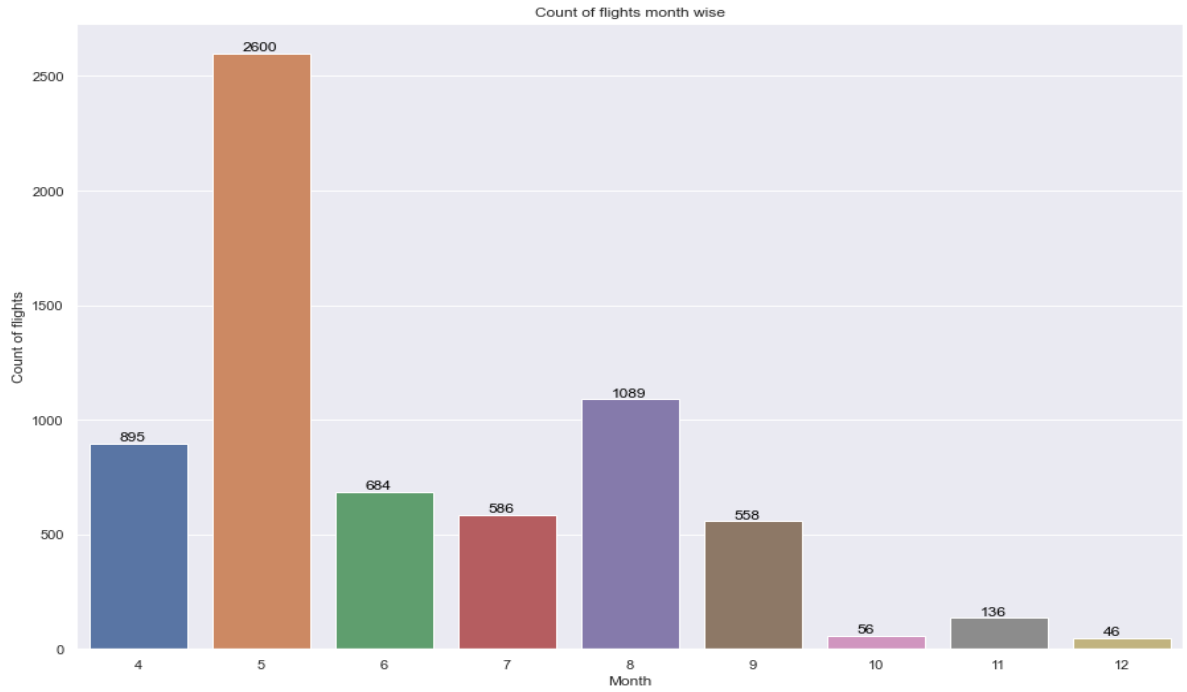
- **Plot 1:**



On the above plot, we have plotted a scatterplot for 'Out_Travel_Time' vs 'Price'. We can notice that Price is not that much related with the Travel time in this case.

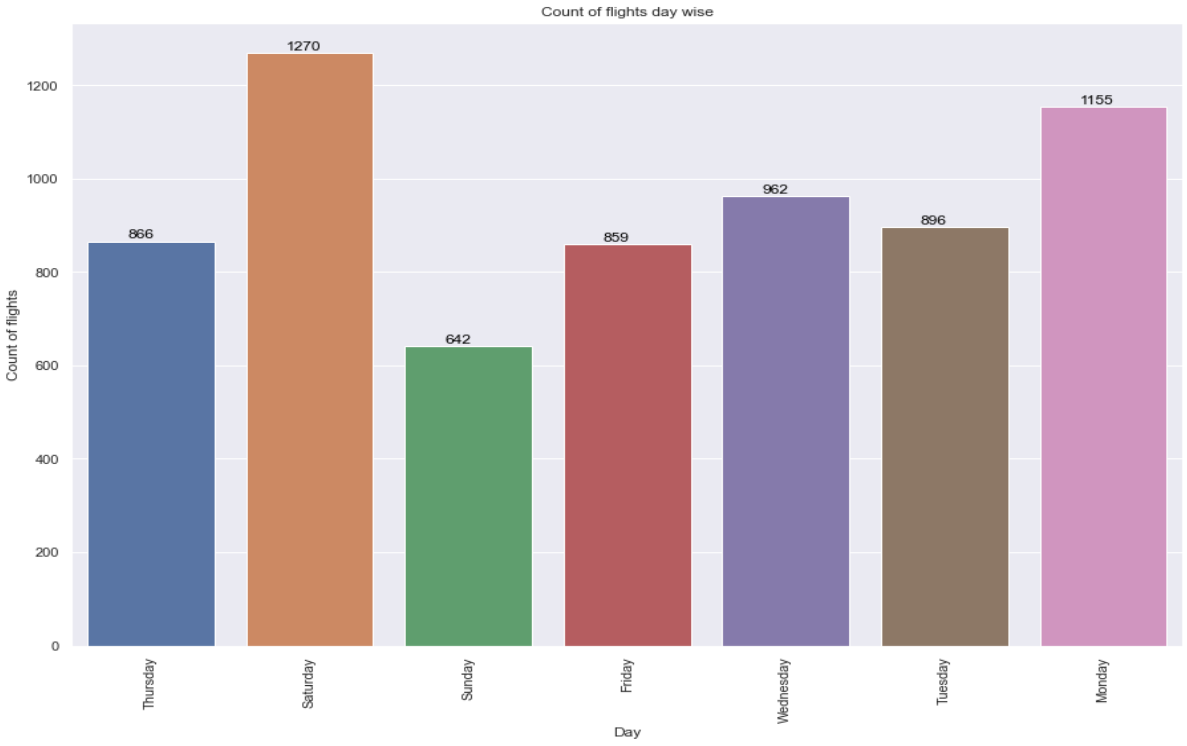
We will do further analysis on various attributes in the next steps.

➢ Plot 2:



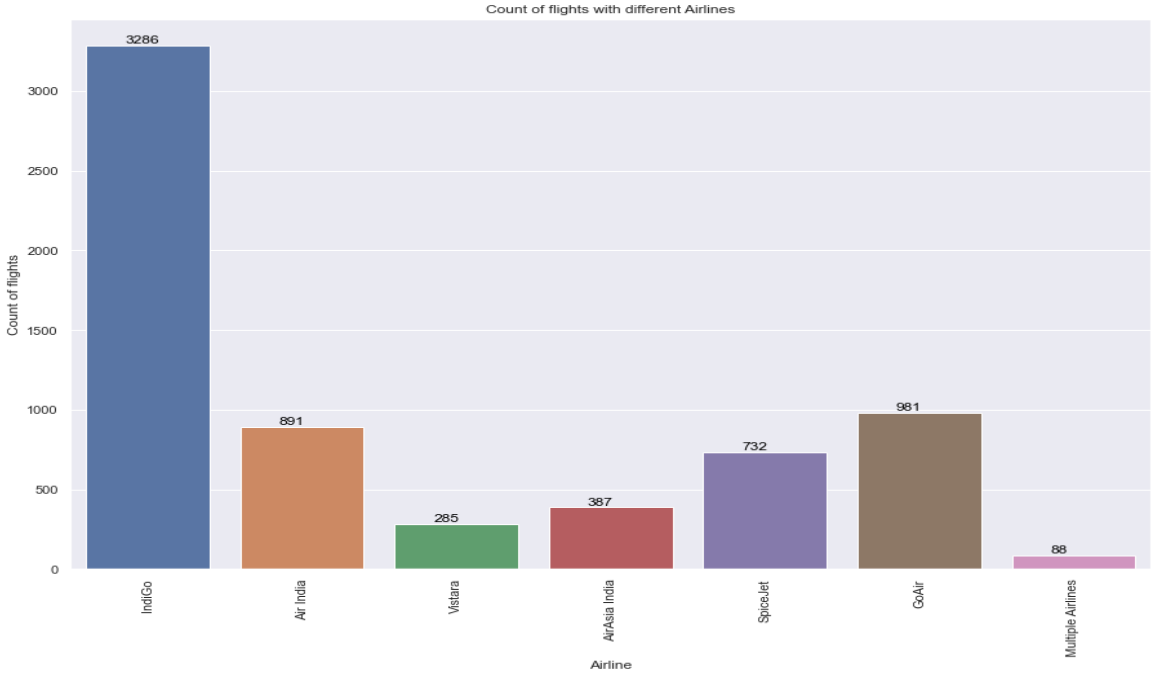
The above plot represents Count of flight per month wise.
We can observe that the demand in the month of May is quiet high than other month. The Month of December has a low count for flight booking.

➢ Plot 3:



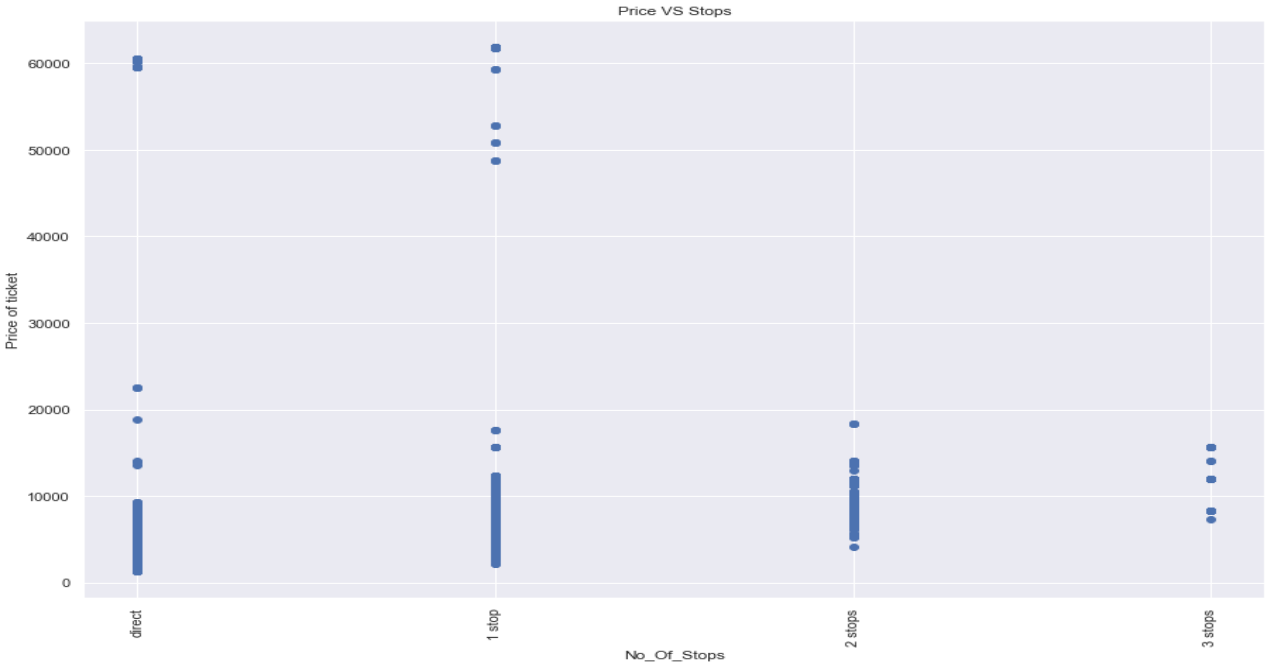
This is a plot representing Flight booking as per day wise. We can get that Sunday and Monday has a higher demand of booking than other days.

➢ Plot 3:



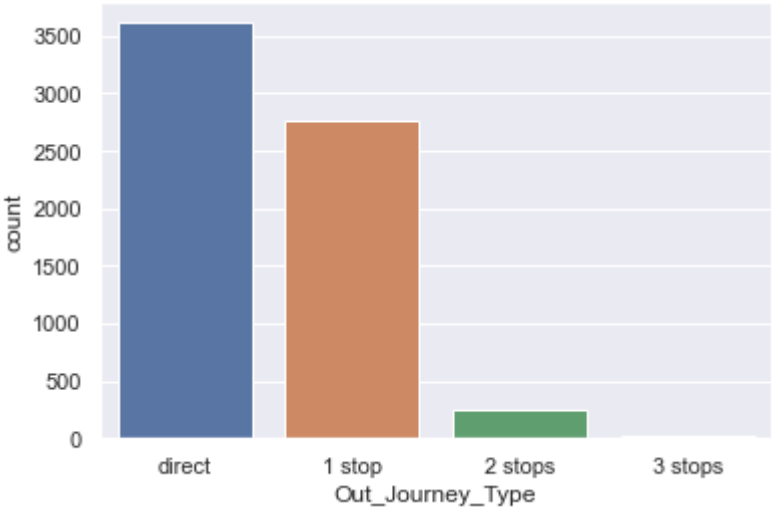
The above plot represents Count of flight with different airline. We found an important feature is that Airline Company 'IndiGo' has a much higher booking count than other airlines. We know 'IndiGo' offers cheap price with almost all of the location in India. That may be the reason behind it.

➢ Plot 4:



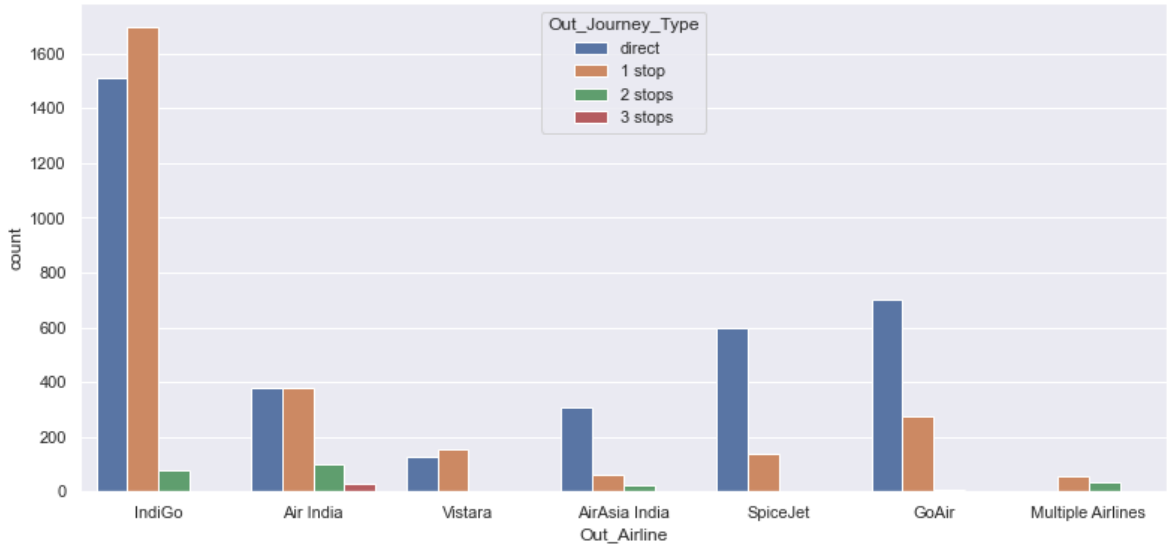
This is a plot about Price vs Stops. We can see that the direct and one stop flight price are higher than two stop and three-stop flight price.

➢ Plot 5:



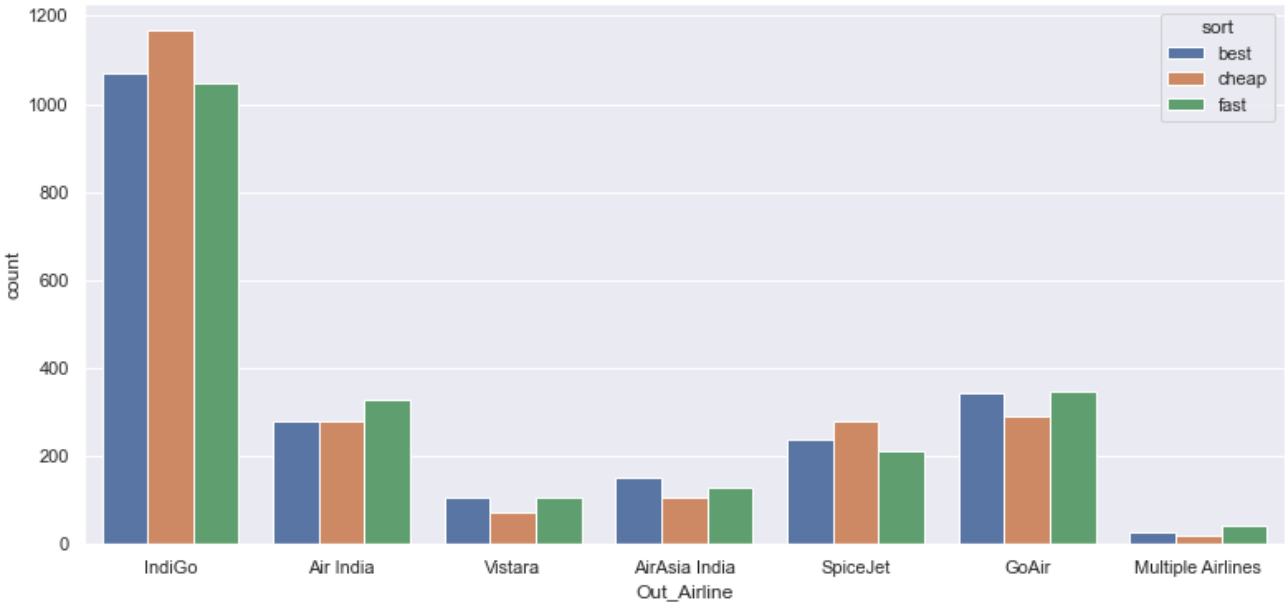
This plot is about journey type vs count. We can analyse from this plot that the passengers choose direct flight more than other flight. However, one stop is also ok with the customers but they do not like more than one stop flights like two or three stop.

➢ Plot 6:



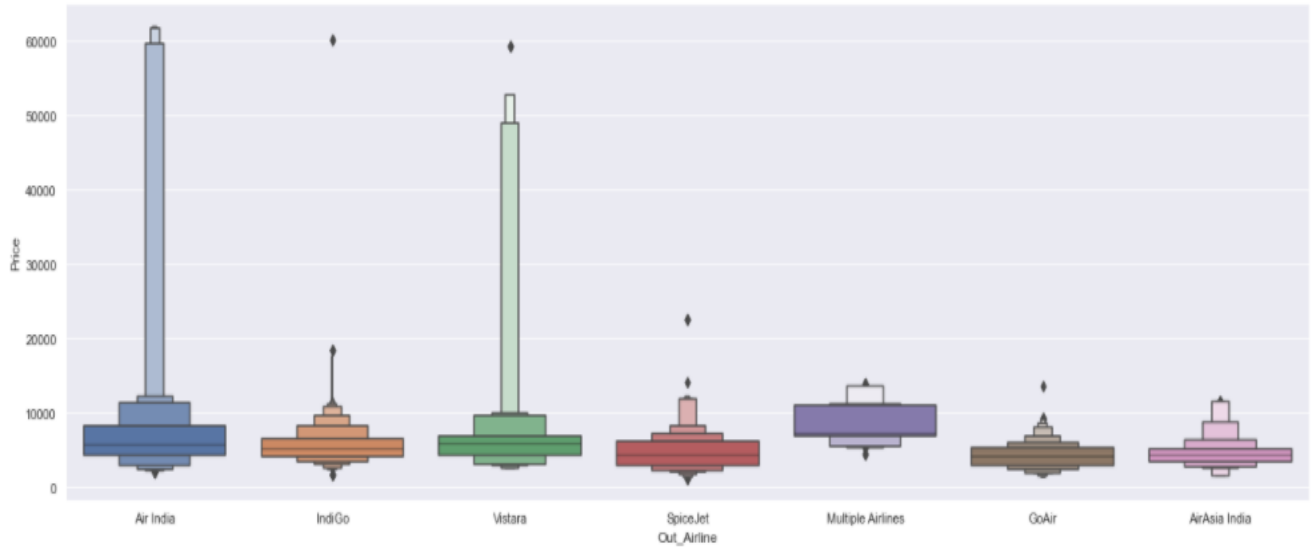
This a more detail countplot and we can observe that 'IndiGo' has a much higher flight booking than other airlines. Customers prefer one stop just as direct flight from 'IndiGo'. Overall People prefer direct flight than any stop if all airlines considered.

➢ Plot 7:



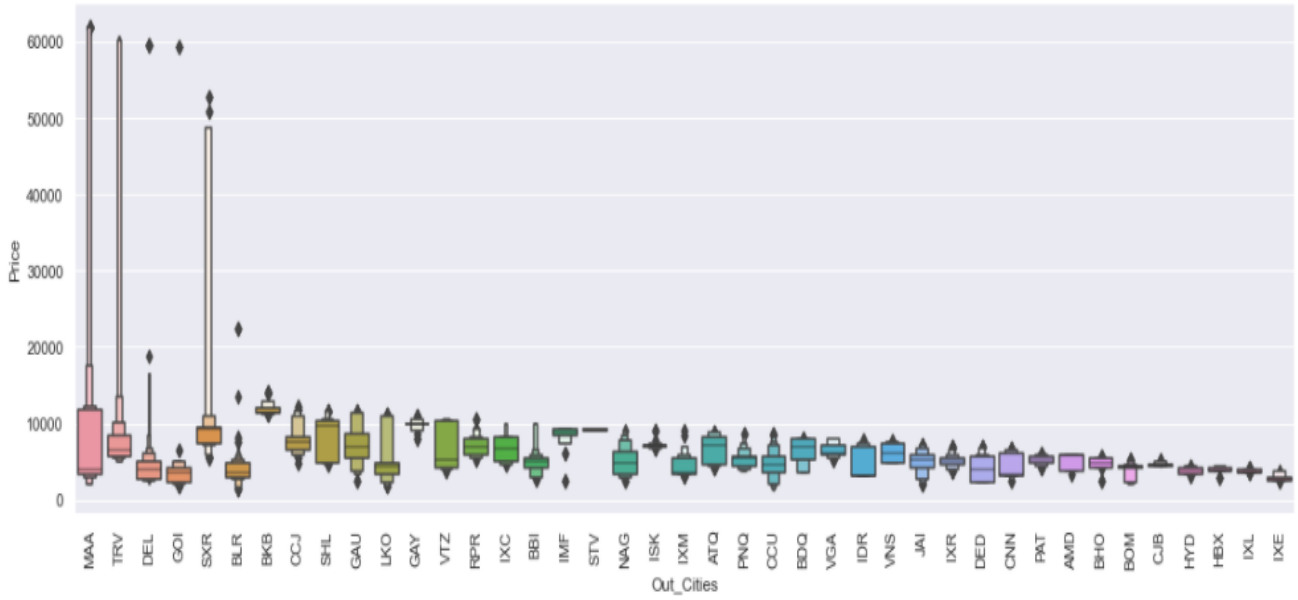
In this plot we have count the sort type (best, cheap and fast) for the different airline. In this, case 'IndiGo' is best for all three types.

➢ Plot 8:



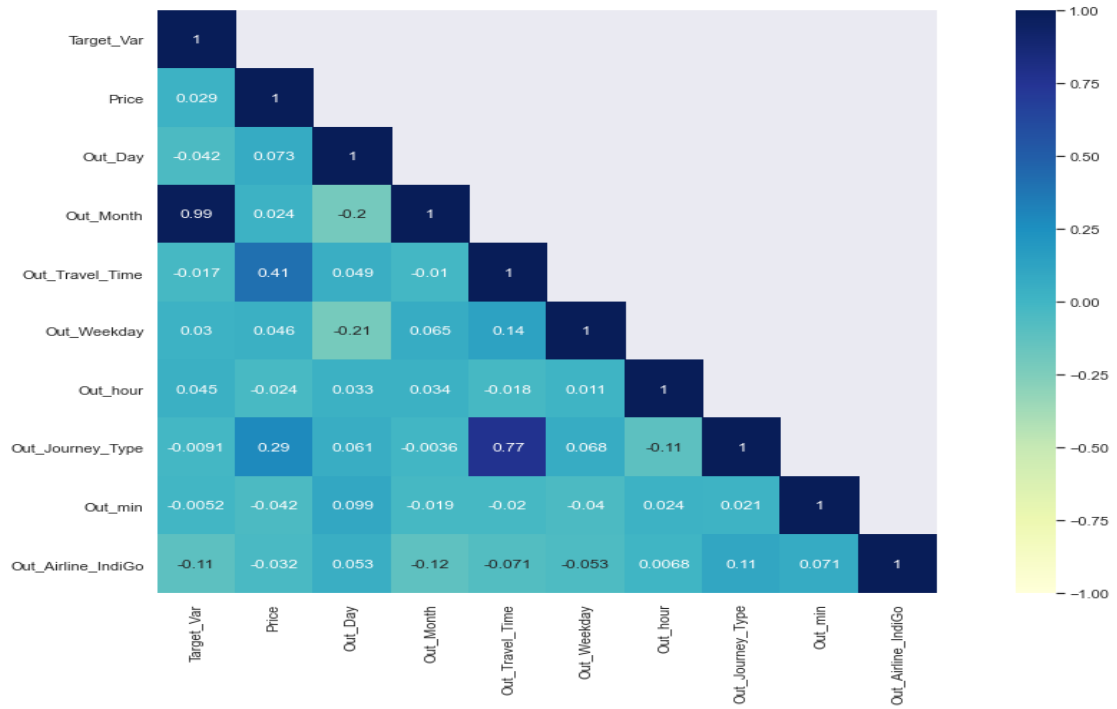
This plot represents out airline vs price and we can see that 'IndiGo' do more business than other airlines. Although 'Vistara' is close to it.

➢ Plot 9:



This plot represents flight piece according to destination cities. We can see that 'MAA' has higher price than other does.

➢ Plot 10:



This heat map shows how much the variables are correlated with each other. The travelling month is important for target variable. 'Out_Journey_Type' and 'Out_Travel_Time' has a relation with Price. So as 'Out_Journey_Type' vs 'Out_Travel_Time'.

Feature selection and Modelling

After Completing the EDA, we split the dataset in 70:30 ration with respect to Train and Test. Then we apply standard scalar, PCA and jump to the modelling section.

Model Building:

After cleaning our given data set and performing PCA on formed data set, we started by selecting best model for our dataset. We tried with approximately 9 models for the selection of the best one. The list of the model and their respective accuracy and MAE values are mentioned in the table given below:

Sr. No.	Model Name	Train R2	Train MAE	Test R2	Test MAE
1	SVC Regressor	0.998	2941.927	0.998	2974.582
2	KNN	1	0.415	0.991	1133.269
3	XGBoost Regressor	1	103.999	0.993	2017.389
4	Random Forest Regressor	0.996	2032.324	0.988	3703.585
5	Decision Tree Regressor	1	329.473	0.98	2220.542
6	MLP Regressor	0.999	2427.886	0.998	2469.007
7	SGD Regressor	0.999	2193.268	0.999	2229.082
8	Lasso Regression	0.999	2196.623	0.999	2246.86
9	Ridge Regression	0.999	2197.549	0.999	2243.425

From the above table, by comparing the accuracy and MAE values we decided to go with the SGD Regressor model.

As “Stochastic Gradient Descent” (SGD) is simple yet very efficient approach to fitting linear classifiers and Regressors under convex loss function such as (linear) support vector machine and logistic regression.

SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Given that the data is sparse, the classifiers in this module easily scale to problems with more than 10^5 training examples and more than 10^5 features.

Advantages of Stochastic Gradient Descent:

1. It is easier to fit into memory due to a single training sample being processed by the network.
2. It is computationally fast as only one sample is processed at a time.
3. For larger datasets, it can converge faster as it causes updates to the parameters more frequently.
4. Due to frequent updates the steps taken towards the minima of the loss function have oscillations which can help getting out of local minimums of the loss function (in case the computed position turns out to be the local minimum)

Perform hyperparameter tuning on the test model:

The code of our final model over training data set is as below:

Model 3- SGD Regressor

```
from sklearn.linear_model import SGDRegressor
sgd = SGDRegressor(alpha=0.0001, average=True, early_stopping=True, epsilon=0.2,
       eta0=0.12, fit_intercept=True, l1_ratio=0.25,
       learning_rate='invscaling', loss='squared_loss', max_iter=10,n_iter_no_change=5, penalty='l1', power_t=0.22,
       random_state=15, shuffle=True, tol=None, validation_fraction=0.1,
        warm_start=False)



folds = KFold(n_splits = 7, shuffle = True, random_state = 4)
sgd_mod = RandomizedSearchCV(estimator = sgd, param_distributions ={} , cv =folds, scoring = 'neg_mean_squared_error',n_
                             return_train_score = True,random_state = 30)

```

```
sgd_mod.fit(X_train_pca, y_train)
```

```
Fitting 7 folds for each of 1 candidates, totalling 7 fits
```

```
]: RandomizedSearchCV(cv=KFold(n_splits=7, random_state=4, shuffle=True),
                   estimator=SGDRegressor(average=True, early_stopping=True,
                                          epsilon=0.2, eta0=0.12, l1_ratio=0.25,
                                          max_iter=10, penalty='l1',
                                          power_t=0.22, random_state=15,
                                          tol=None),
                   n_jobs=-1, param_distributions={}, random_state=30,
                   return_train_score=True, scoring='neg_mean_squared_error',
                   verbose=1)
```

The final result over training data set is as given below:

```
print("Train Results for SGDRegressor Model:")
print(50 * '-')
print("Root mean squared error: ", round(sqrt(mean_squared_error(y_train.values, y_train_pred)),3))
print("Mean absolute error: ",train_mae)
print("Mean squared error: ",round(mean_squared_error(y_train.values, y_train_pred),3))
print("R-squared: ", train_r2)
```

```
Train Results for SGDRegressor Model:
--------------------------------------------------
Root mean squared error:  2730.917
Mean absolute error:  2193.268
Mean squared error:  7457909.863
R-squared:  0.999
```

As we can see, this model gives us the satisfactory outcome on training data having 0.99 of R-squared value and 2730.917 of Root mean squared value.

Even we tested our model on test data set. The results are as given below:

```
print("Test Results for SGDRegressor Model:")
print(50 * '-')
print("Root mean squared error: ", round(sqrt(mean_squared_error(y_test.values, y_test_pred)),3))
print("Mean absolute error: ",test_mae)
print("Mean squared error: ",round(mean_squared_error(y_test.values, y_test_pred),3))
print("R-squared: ", test_r2)
```

```
Test Results for SGDRegressor Model:
--------------------------------------------------
Root mean squared error:  2771.966
Mean absolute error:  2229.082
Mean squared error:  7683797.923
R-squared:  0.999
```

We performed further deployment process by using this selected SGD Regressor model.

```
print("Train Results for Random Forest Regressor Model:")
print(50 * '-')
print("Root mean squared error: ", round(sqrt(mean_squared_error(y_train1.values, y_train_pred)),3))
print("Mean absolute error: ",train_mae)
print("Mean squared error: ",round(mean_squared_error(y_train1.values, y_train_pred),3))
print("R-squared: ", train_r2)
```

```
Train Results for Random Forest Regressor Model:
--------------------------------------------------
Root mean squared error:  2246.695
Mean absolute error:  564.268
Mean squared error:  5047638.008
R-squared:  0.815
```

For the optimal pricing, we use Random Forest Regressor Model.

Deployment

There are many ways to deploy application on various platform like that of Heroku, Microsoft Azure and Google Cloud Platform.

In our project we have, used Python based Flask web framework in order to have speedy deployment.

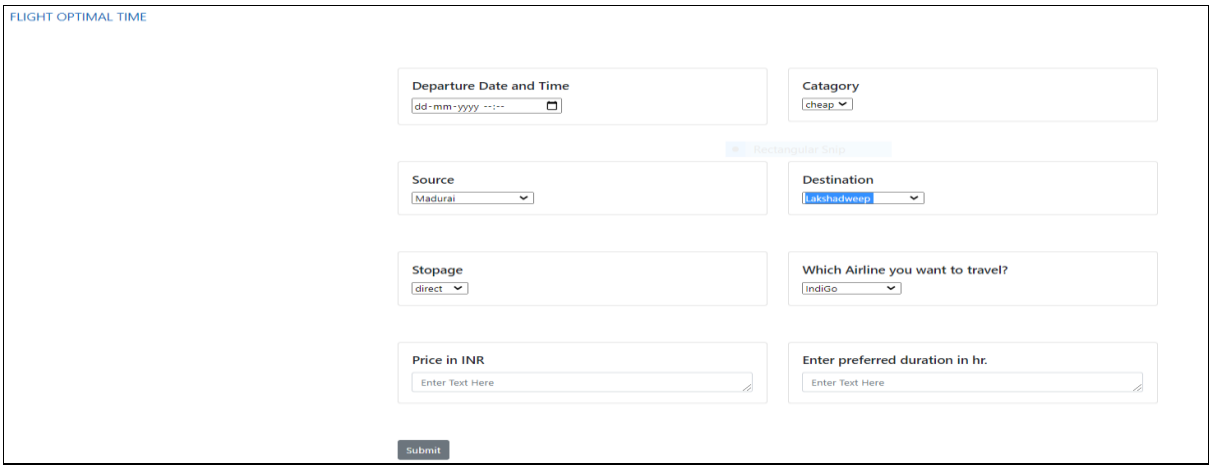
The major benefits of using Flask are:

- Extremely flexible
- Simple to use
- Small core and easily extensible
- Fast in deployment
- Built-in encryption for user sessions

We did as follows:

I. Virtual Environment is created for the working project directory
II. Application is tested locally, on http://127.0.0.1:5000/ endpoint
III. The Heroku CLI is used for managing the application after deployment and Git Bash, both are downloaded and install on local server
IV. Pip is used to install library such flask and guicorn.
V. "Pip freeze > requirements.txt" command is used to generate file that contains all the libraries and their versions
VI. Procfile is created in order to tell Heroku to run the application using guicorn. Command written inside Procfile is as follows:-
web: gunicorn app:app
VII. We need to create Heroku login and upload code to the Git
VIII. After login, we need to create new application inside Heroku and push the entire project to Heroku by using Heroku CLI commands
Hence, the application got deployed on Heroku and its link is

https://flight-time.herokuapp.com/



THANK YOU