FLIP ROBO

Flight Price Prediction Project Report

Submitted by:

Rajesh Kumar Singh

ACKNOWLEDGMENT

I would like to express my deepest gratitude to Flip Robo Technologies who gave me the opportunity to do this project on Flight Price Prediction, which also helped me in doing lots of research wherein I came to know about so many new things especially the data collection part.

Also, I have utilized a few external resources that helped me to complete the project. I ensured that I learn from the samples and modify things according to my project requirement. All the external resources that were used in creating this project are listed below:

1) https://www.google.com/

2) https://www.youtube.com/

3) https://scikit-learn.org/stable/user_guide.html

4) https://github.com/

5) https://www.kaggle.com/

6) https://medium.com/

7) https://towardsdatascience.com/

8) https://www.analyticsvidhya.com/

INTRODUCTION

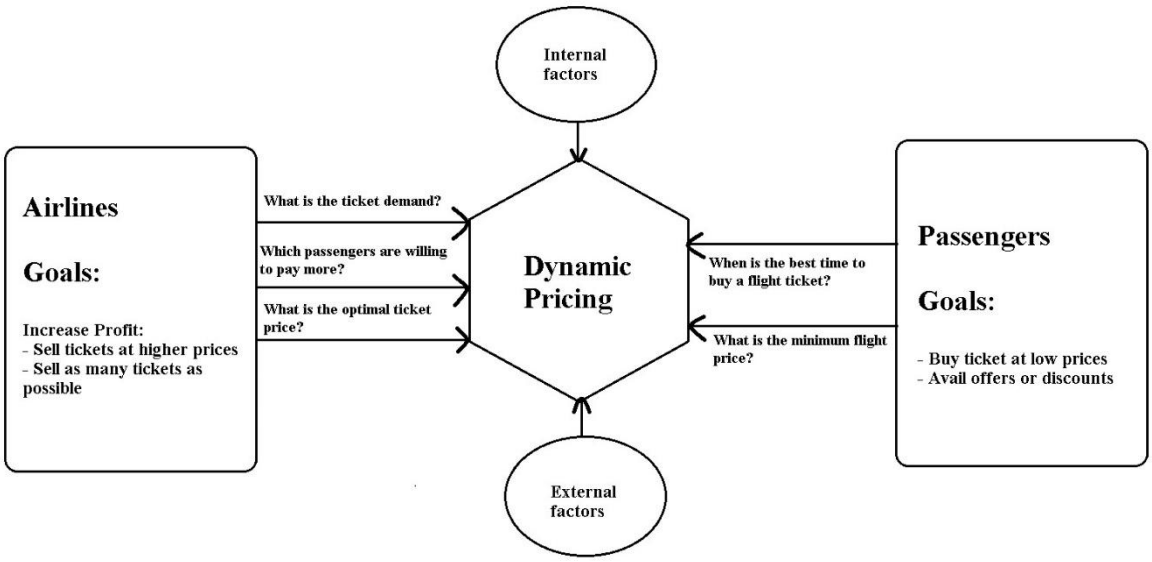
- Business Problem Framing

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 7 times a day. Customers are seeking to get the lowest price for their ticket, while airline companies are trying to keep their overall revenue as high as possible and maximize their profit. However, mismatches between available seats and passenger demand usually leads to either the customer paying more or the airlines company losing 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.

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. 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

  Airline companies use complex algorithms to calculate flight prices given various conditions present at that particular time. These methods take financial, marketing, and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we will try to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

- Review of Literature

  As per the requirement of client, I have scrapped the data from online sites and based on that data I have did analysis like for based on which feature of my data prices are changing. and checked the relationship of flight price with all the feature like what flight he should choose.

- Motivation for the Problem Undertaken

  I have worked on this on the bases of client requirements and followed all the steps till model deployment.

Analytical Problem Framing

- Mathematical/ Analytical Modeling of the Problem

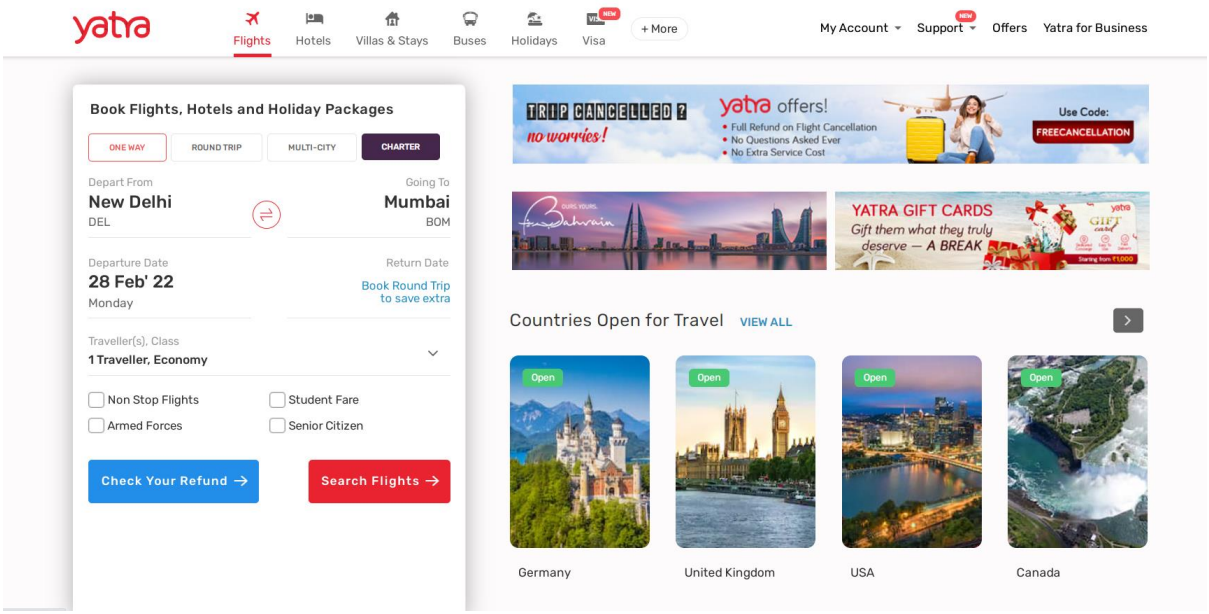
  In our scrapped dataset, our target variable "Flight_Prices " is a continuous variable. Therefore, we will be handling this modelling problem as regression.

  This project is done in three parts:

  - Data Collection
  - Data Analysis
  - Model Building

  1. Data Collection

  You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data better the model. In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.



2. Data Analysis

After cleaning the data, you have to do some analysis on the data. Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the consumer can save the most by taking the least risk? Does price increase as we get near to departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

3. Model Building

After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model.

Follow the complete life cycle of data science. Include all the steps like

1. Data Cleaning
2. Exploratory Data Analysis
3. Data Pre-processing
4. Model Building
5. Model Evaluation
6. Selecting the best model

- Data Sources and their formats

The dataset is in the form of CSV (Comma Separated Value) format and consists of 9 columns (8 features and 1 label) with 5805 number of records as explained below:

- Airline_Names : This shows the list of all the Airline Names for which the data got scraped
- Departure_Time : In this column we have the timings of every flight departure
- Arrival_Time : Here in this column we have the timings of every flight arrival
- Flight_Duration : We can see the total duration of a flight that it took to fly from the source to the destination
- Source_Place : Gives us the name of the source place where the flight journey began
- Destination_Place : Shows us the name of the destination place where the flight journey ended
- Meal_Availability : Provides us with the information on type of meal that the passenger is eligible for
- Number_Of_Stops : Lists the number of stops the flight is going to take to complete the entire journey
- Flight_Prices : Finally we have our label column that has the ticket prices for the aircraft journey

We can see our dataset includes a target label "Used Car Price" column and the remaining feature columns can be used to determine or help in predicting the price of the used cars. Since price is a continuous value it makes this to be a Regression problem!

Out[2]:

	Unnamed: 0	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price
0	0	Air Asia	08:20	14:10	5h 50m	New Delhi	Mumbai	eCash 250	1 Stop	5,949
1	1	Air Asia	20:45	07:15	10h 30m	New Delhi	Mumbai	Emissions: 303 Kg CO2	1 Stop	5,949
2	2	Air Asia	09:35	20:35	11h 00m	New Delhi	Mumbai	Emissions: 303 Kg CO2	1 Stop	5,949
3	3	Air Asia	08:00	20:35	12h 35m	New Delhi	Mumbai	Emissions: 303 Kg CO2	1 Stop	5,949
4	4	SpiceJet	18:55	21:05	2h 10m	New Delhi	Mumbai	eCash 250	Non Stop	5,953
...	...	...	...	...	...	...	...	...	...	...
5750	5750	Air India	14:45	13:15	22h 30m	Lucknow	Jaipur	Free Meal	2 Stop(s)	23,112
5751	5751	Air India	14:45	13:15	22h 30m	Lucknow	Jaipur	Free Meal	2 Stop(s)	23,112
5752	5752	Air India	14:45	13:15	22h 30m	Lucknow	Jaipur	Free Meal	2 Stop(s)	23,112
5753	5753	Air India	14:45	13:15	22h 30m	Lucknow	Jaipur	Free Meal	2 Stop(s)	23,112
5754	5754	Air India	14:45	13:15	22h 30m	Lucknow	Jaipur	Free Meal	3 Stop(s)	23,112

5755 rows × 10 columns

- Data Preprocessing Done

Checked the datatype details for each column to understand the numeric ones and its further conversion process.

In [4]:
```
#checking info about the dataset
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5755 entries, 0 to 5754
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype 
---  ------             --------------  ----- 
 0   Unnamed: 0         5755 non-null   int64 
 1   Airline            5755 non-null   object
 2   Departure_time     5755 non-null   object
 3   Time_of_arrival    5755 non-null   object
 4   Duration           5755 non-null   object
 5   Source             5755 non-null   object
 6   Destination        5755 non-null   object
 7   Meal_availability  5755 non-null   object
 8   Number_of_stops    5755 non-null   object
 9   Price              5755 non-null   object
dtypes: int64(1), object(9)
memory usage: 449.7+ KB
```

I also took a look at all the unique value present in each of the columns and then decided to view the details for those column values that had less than 10 categories.

```
In [63]: df.nunique().sort_values().to_frame("Unique Values")
```

Out[63]:

	Unique Values
Meal_availability	3
Number_of_stops	5
Airline	6
Source	9
Destination	9
Departure_time	245
Time_of_arrival	246
Duration	400
Price	2126

The various data processing performed on our data set are shown below with the code.

```
#Lets check total stops
df["Meal_availability"].value_counts()
```

```
eCash 250                2475
Free Meal                1990
No Meal Fare             1219
Emissions: 142 Kg CO2      17
Emissions: 109 Kg CO2      14
Emissions: 303 Kg CO2      10
Emissions: 191 Kg CO2       6
Emissions: 185 Kg CO2       6
Emissions: 331 Kg CO2       6
Emissions: 140 Kg CO2       6
Emissions: 112 Kg CO2       4
Emissions: 252 Kg CO2       2
Name: Meal_availability, dtype: int64
```

```
df['Meal_availability'] = df['Meal_availability'].replace('eCash 250','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 142 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 109 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 303 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 331 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 185 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 191 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 140 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 112 Kg CO2','None')
```

```
df['Meal_availability'] = df['Meal_availability'].replace('Emissions: 252 Kg CO2','None')
```

```
df
```

```
#Extracting numerical data using Duration
df["hour"] = df.Duration.str.split('h').str.get(0)
df["min"] = df.Duration.str.split('h').str.get(1)
df["min"]=df["min"].str.split('m').str.get(0)
df["hour"]=df['hour'].astype('float')
df["min"]=df['min'].astype('float')

df["Duration"] = df["hour"] + df["min"]/60
```

```
df.head()
```

	Airline	Departure_time	Time_of_arrival	Duration	Source	Destination	Meal_availability	Number_of_stops	Price	hour	min
0	Air Asia	08:20	14:10	5.833333	New Delhi	Mumbai	None	1	5,949	5.0	50.0
1	Air Asia	20:45	07:15	10.500000	New Delhi	Mumbai	None	1	5,949	10.0	30.0
2	Air Asia	09:35	20:35	11.000000	New Delhi	Mumbai	None	1	5,949	11.0	0.0
3	Air Asia	08:00	20:35	12.583333	New Delhi	Mumbai	None	1	5,949	12.0	35.0
4	SpiceJet	18:55	21:05	2.166667	New Delhi	Mumbai	None	0	5,953	2.0	10.0

Using hour and min column I have created Duration column with float values, now I am dropping hour and min columns

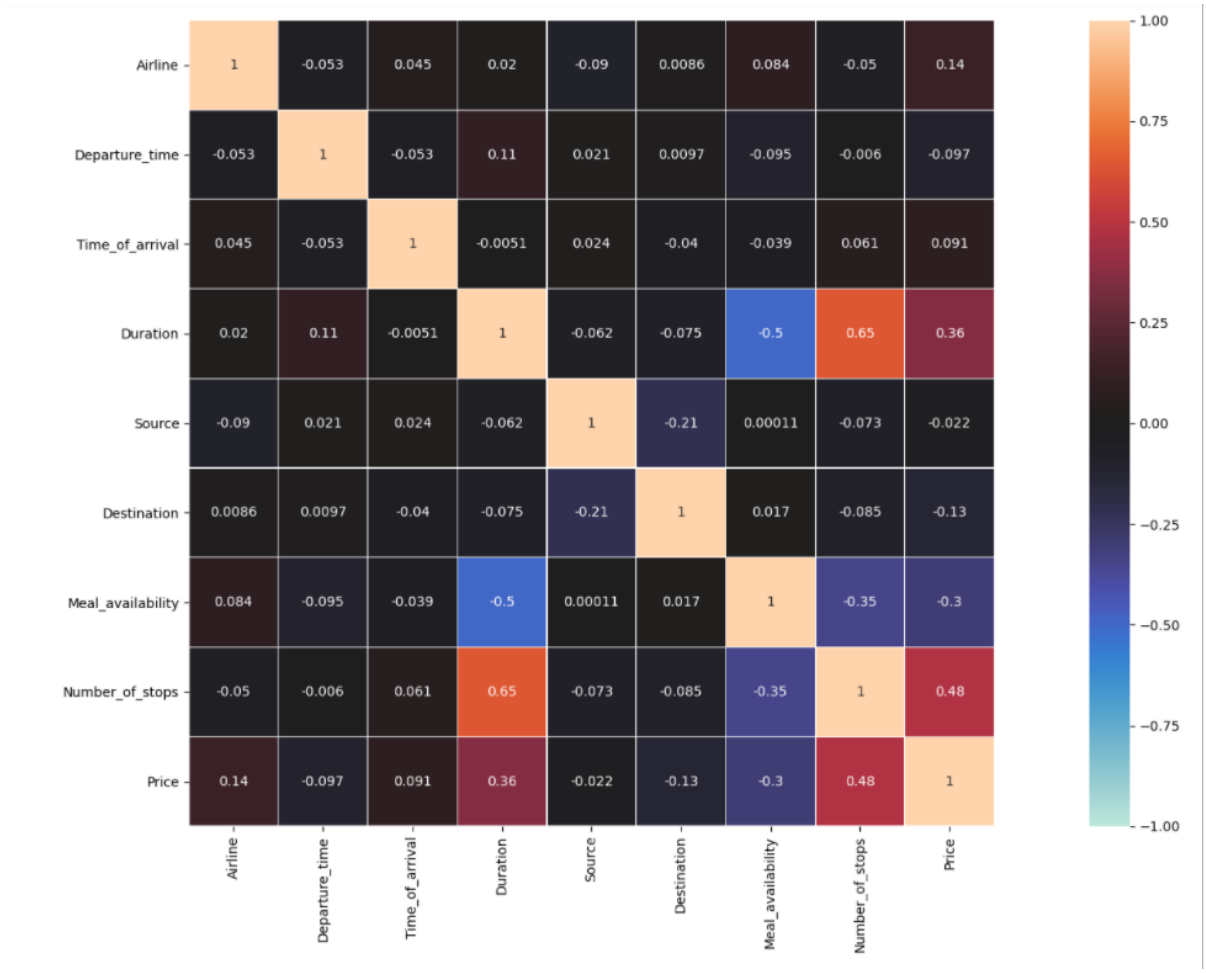
```
df.drop(columns = ["hour","min"], inplace = True)
```

```
#Lets convert data type of Price column to float
df['Price'] = df['Price'].str.replace(',','')
df['Price'] = df['Price'].astype('float')
```

```
#Similar to Duration I will extract numric data from Departure_time and Time_of_arrival columns using below codes.

df["Dep_hour"] = pd.to_datetime(df.Departure_time, format="%H:%M").dt.hour
df["Dep_min"] = pd.to_datetime(df.Departure_time, format="%H:%M").dt.minute
df["Departure_time"]= df['Dep_hour']+df['Dep_min']/60
df.drop(columns = ['Dep_hour','Dep_min'],inplace=True)

df["Arvl_hour"] = pd.to_datetime(df.Time_of_arrival, format="%H:%M").dt.hour
df["arvl_min"] = pd.to_datetime(df.Time_of_arrival, format="%H:%M").dt.minute
df["Time_of_arrival"]= df['Arvl_hour']+df['arvl_min']/60
df.drop(columns = ['Arvl_hour','arvl_min'],inplace=True)
```



- Data Inputs- Logic- Output Relationships

  The input data were initially all object datatype so had to clean the data by removing unwanted information like “h” and “m” from Flight_Duration column and ensuring the numeric data are converted accordingly. I then used Ordinal Encoding method to convert all the categorical feature columns to numeric format.

  Code:

Encoding

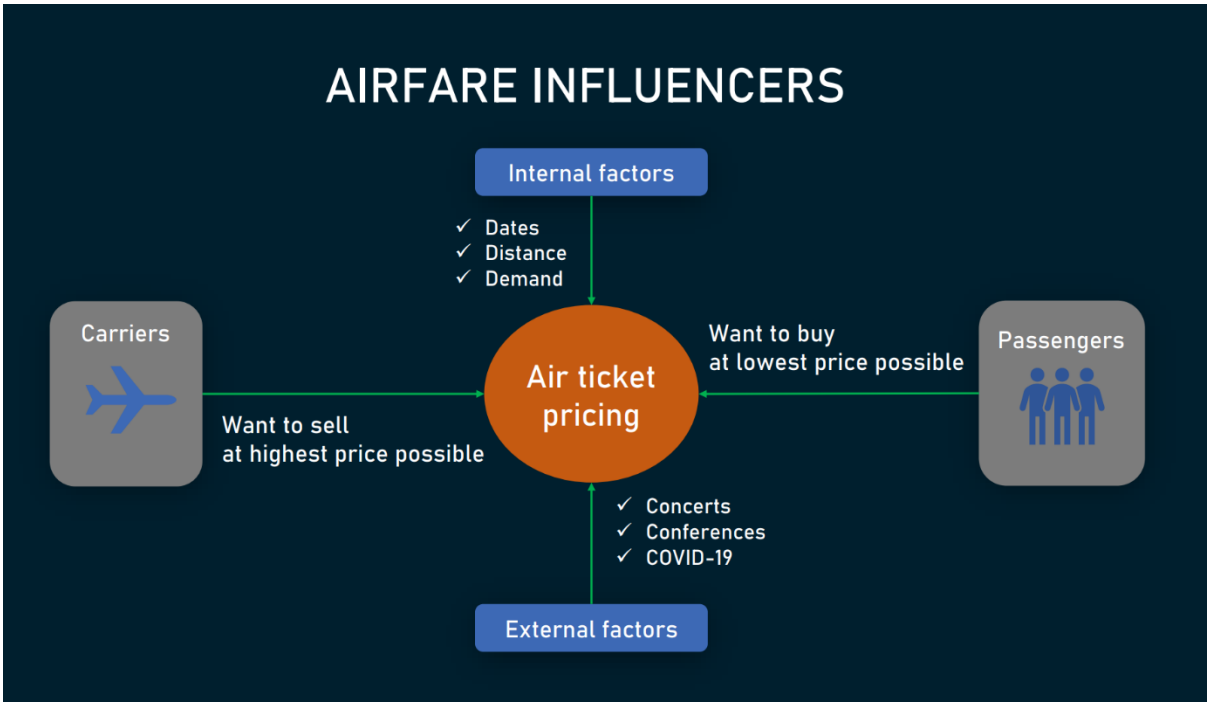
```
]: #lets convert categorical data into numeric values, using OrdinalEncoder
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
for i in df.columns:
    if df[i].dtypes == "object" :
        df[i] = enc.fit_transform(df[i].values.reshape(-1,1))
```

Since we had mostly categorical feature data we did not have to worry about outliers and skewness concerns.

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

Pricing in the airline industry is often compared to a brain game between carriers and passengers where each party pursues the best rates. Carrier's love selling tickets at the highest price possible — while still not losing consumers to competitors. Passengers are crazy about buying flights at the lowest cost available — while not missing the chance to get on board. All this makes flight prices fluctuant and hard to predict. But nothing is impossible for people armed with intellect and algorithms.

There are two main use cases of flight price prediction in the travel industry. OTAs and other travel platforms integrate this feature to attract more visitors looking for the best rates. Airlines employ the technology to forecast rates of competitors and adjust their pricing strategies accordingly.



A passenger-side predictor proposed by an OTA suggests the best time to buy a ticket so that travellers can make informed decisions. Carriers, on their end, try to find out the optimal price they should set to maximize revenue while remaining competitive.

In both cases, the task is quite challenging because numerous internal and external factors influence airfares.

Internal factors include

- purchase and departure dates,
- seasonality,
- holidays,
- the number of available airlines and flights,
- fare class,
- the current market demand, and
- flight distance.

External factors embrace events going on in the arrival or departure cities — like

- concerts,
- sports competitions,
- festivals,
- terrorist attacks,
- natural disasters,
- political gatherings,
- epidemic outbursts, and
- economic activities.

Though it's impossible to cover every external eventuality — say, nothing foreshadowed the 2020 coronavirus pandemic in the middle of 2019 — we still can predict quite a lot, using the right data and advanced machine learning (ML) models.

- Hardware and Software Requirements and Tools Used

Hardware technology being used.

- ✓ RAM: 4 GB
- ✓ CPU: INTEL Core i3, 1.99GHz.
- ✓ GPU: NVIDIA GETFORCE RAM: 4 GB

Software technology being used.

Programming language : Python

Distribution : Anaconda Navigator

Browser based language shell : Jupyter Notebook

Libraries/Packages specifically being used.

Pandas, NumPy, matplotlib, seaborn, scikit-learn, pandas-profiling.

Model/s Development and Evaluation

- Identification of possible problem-solving approaches (methods)

  1. Clean the dataset from unwanted scraped details.

  2. Rename values with meaningful information.

  3. Encoding the categorical data to get numerical input data.

  4. Compare different models and identify the suitable model.

  5. R2 score is used as the primary evaluation metric.

  6. MSE and RMSE are used as secondary metrics.

  7. Cross Validation Score was used to ensure there are no overfitting our underfitting models.

- Testing of Identified Approaches (Algorithms)

  Libraries and Machine Learning Regression models that were used in this project are shown below.

```
: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import missingno
import pandas_profiling
from sklearn.preprocessing import StandardScaler

#model selection
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.model_selection import cross_val_score

#model evaluation
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import ExtraTreesRegressor
from lightgbm import LGBMRegressor
from sklearn.linear_model import Ridge,Lasso,RidgeCV,LassoCV


import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

All the regression machine learning algorithms used are:

- Linear Regression Model
- Ridge Regularization Model
- Lasso Regularization Model
- Support Vector Regression Model
- Decision Tree Regression Model
- Random Forest Regression Model
- K Neighbours Regression Model
- Extra Trees Regression Model

- Run and Evaluate selected models

  I created a Regression Machine Leaning Model function incorporating the evaluation metrics so that we can get the required data for all the above models.

  Code:

```
lr = LinearRegression()
dt = DecisionTreeRegressor()
rf = RandomForestRegressor()
xgb = XGBRegressor()
ext = ExtraTreesRegressor()
lgb = LGBMRegressor()
lasso = LassoCV(max_iter=1000, normalize = True)
ridge = RidgeCV(cv=10,alphas=[0.1,1], normalize=True)
```

```
#creating a function to train and test the model with evaluation
def BuiltModel(model):
    print('*'*30+model.__class__.__name__+'*'*30)
    model.fit(x_train,y_train)
    y_pred = model.predict(x_train)
    pred = model.predict(x_test)

    r2score = r2_score(y_test,pred)*100

    #evaluation
    mse = mean_squared_error(y_test,pred)
    rmse = np.sqrt(mse)
    mae = mean_absolute_error(y_test,pred)
    print("MAE :", mae)
    print("RMSE :", rmse)
    print('------------------------------')

    # r2 score
    print(f"Training r2 score:", r2_score(y_train,y_pred)*100,"%")
    print(f"Testing r2 Score:", r2score,"%")
    print('------------------------------')

    #cross validation score
    scores = cross_val_score(model, X_std, np.log(y), cv = 10).mean()*100
    print("\nCross validation score :", scores)

    #result of accuracy minus cv score
    result = r2score - scores
    print("\nAccuracy Score - Cross Validation Score :", result)

    sns.regplot(y_test,pred)
    plt.show()
```

Output:

```
******************************LinearRegression******************************
MAE : 0.2528846914948316
RMSE : 0.325207433262602
------------------------------
Training r2 score: 37.347377134652405 %
Testing r2 Score: 33.14540818022213 %
------------------------------

Cross validation score : 36.07644139820354

Accuracy Score - Cross Validation Score : -2.9310332179814154
```

- Key Metrics for success in solving problem under consideration

RMSE Score:
Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are; RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit.

R2 Score:
The R2 score is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

Cross Validation Score:
Cross-validation is a statistical method used to estimate the skill of machine learning models. It is commonly used in applied machine learning to compare and select a model for a given predictive modelling problem because it is easy to understand, easy to implement, and results in skill estimates that generally have a lower bias than other methods. The k-fold cross validation is a procedure used to estimate the skill of the model on new data. There are common tactics that you can use to select the value of k for your dataset (I have used 5-fold validation in this project). There are commonly used variations on cross-validation such as stratified and repeated that are available in scikit-learn.

Hyper Parameter Tuning:
In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is

used to control the learning process. By contrast, the values of

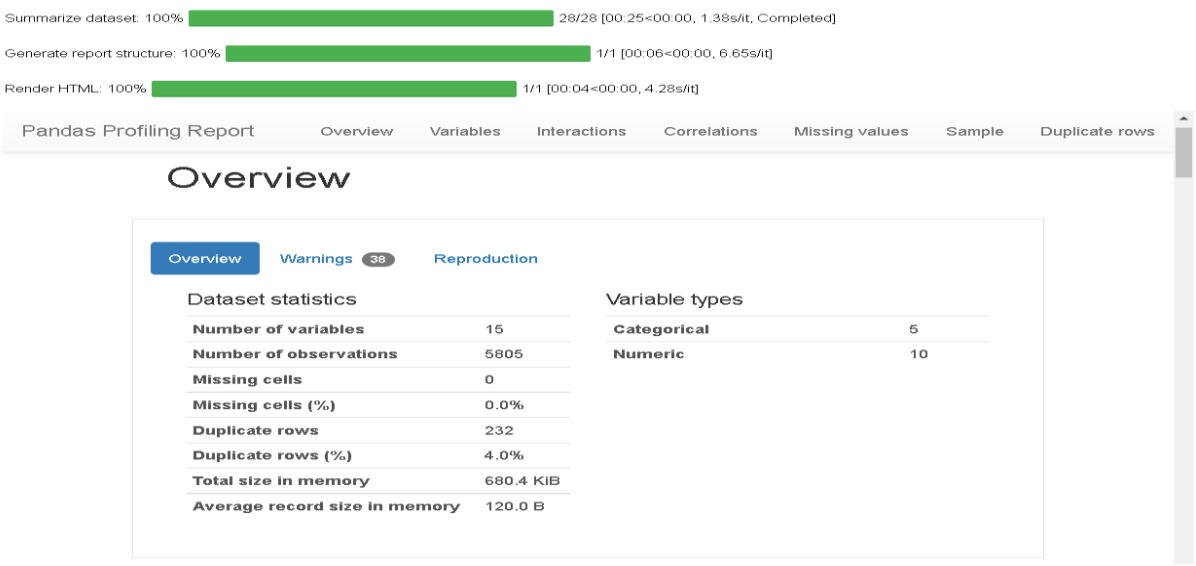
Visualizations

I used the pandas profiling feature to generate an initial detailed report on my dataframe values. It gives us various information on the rendered dataset like the correlations, missing values, duplicate rows, variable types, memory size etc. This assists us in further detailed visualization separating each part one by one comparing and research for the impacts on the prediction of our target label from all the available feature columns.

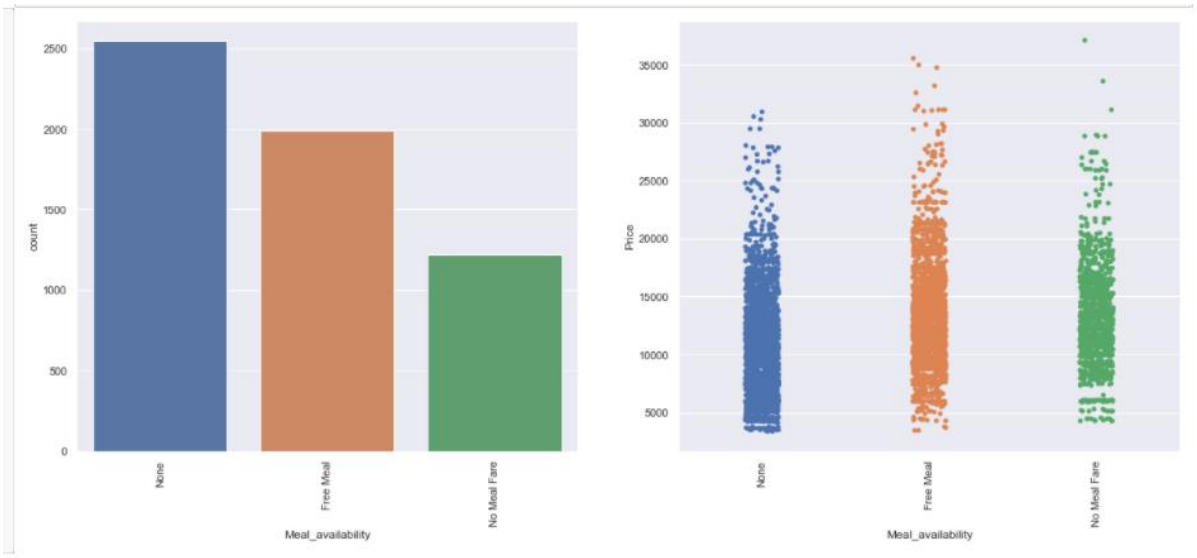
Code:

```
pandas_profiling.ProfileReport(df)
```

Output:



I generated count plots, bar plots, pair plots, heatmap and others to visualise the datapoint present in our column records.

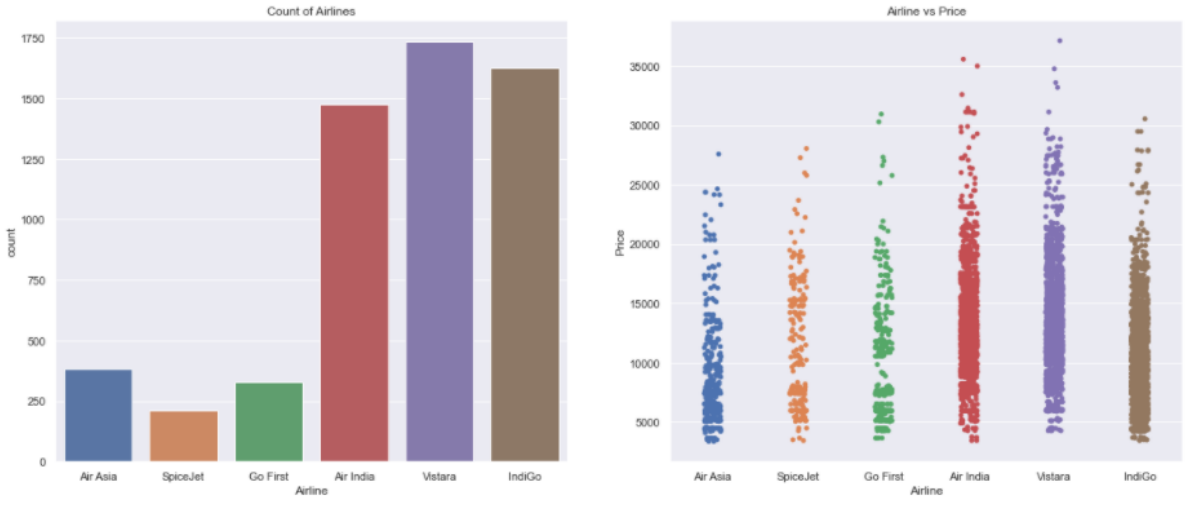


Observation on Fig 1:

There are 73.25% flights that mostly serve no meals in their domestic journey since they are of short distances and duration

We can see that 17.47% flights serve free meals which are probably for tickets that include those prices and meal services

Finally, there are 9.29% flights which offer the eCash meals option that can be redeemed to purchase food during long journey flights mostly with multiple stops.



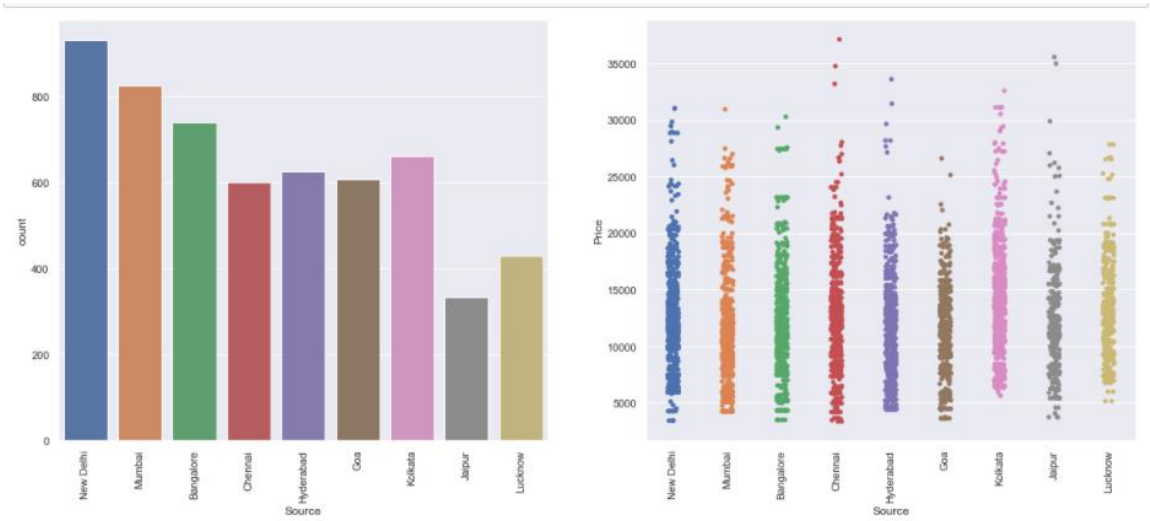
Observation on Fig 2:

Highest number of airlines preferred by people are vistara covering 30.7% of the total record.

We can see that Air India is quite close to the first one and a close competitor standing at the second position holding 28.22% of the total record

At third place we have Vistara airlines that covers 26.08% of total record in our airline data

Airlines Go First, Air Asia and SpiceJet are the least used by people covering 8.48%, 3.67% and 2.86% respectively.



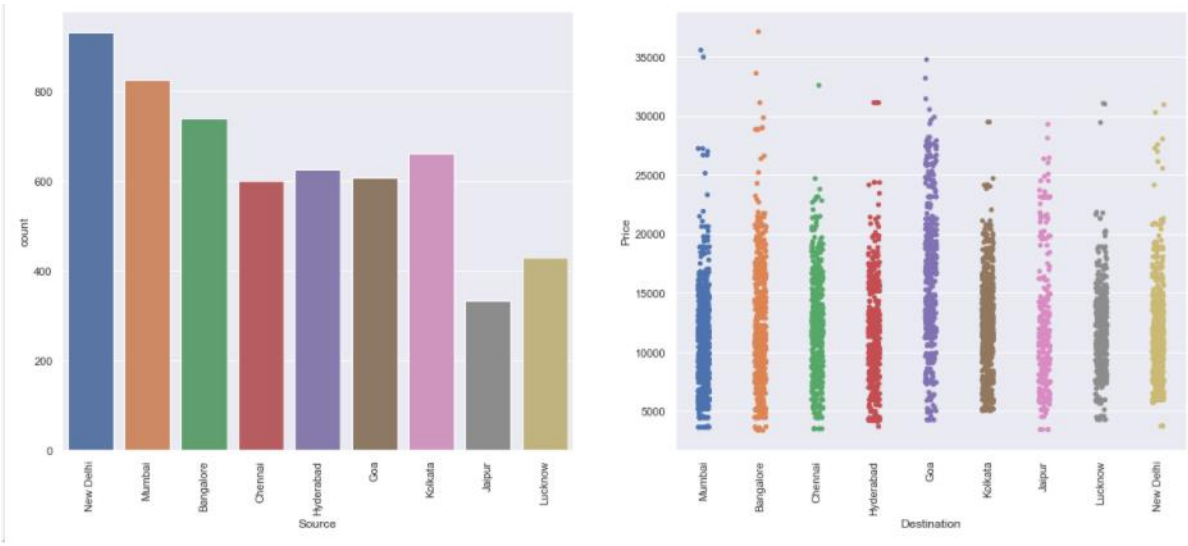
Observation on Fig 3:

The departure area or source place highly used or people majorly flying from the city is "Mumbai" covering 13.88% record in the column

We see that "Bangalore" is a close second wherein it covers 13.01% records in the column

Other two famous locations where people chose to fly from are "New Delhi" and "Kolkata" covering 12.89% and 12.82% respectively

The least travel from location is "Jaipur" where I believe not many people chose to travel from unless they are returning from a vacation there and hence covers 7.36% of record in the column.



Observation on Fig 4:

Just as in case of departure area even in terms of arrival area or destination place people prefer to fly towards the city "Mumbai" covering 13.87% of record

Again, in a similar fashion "Bangalore" city is a close second destination that people like to fly towards covering 13.54% record in the column

Surprisingly we have "Hyderabad" city taking the third place for destination that people prefer landing covering 13.07% of record

Finally, similar to the departure location the least travel to area is "Jaipur" and it covers 5.24% record in the column

Observation on Fig 5:

People generally buy flight tickets that have 1 stop layover covering 63.67% rows in the column

Next in line are 2 stop layovers which cover 19.88% rows

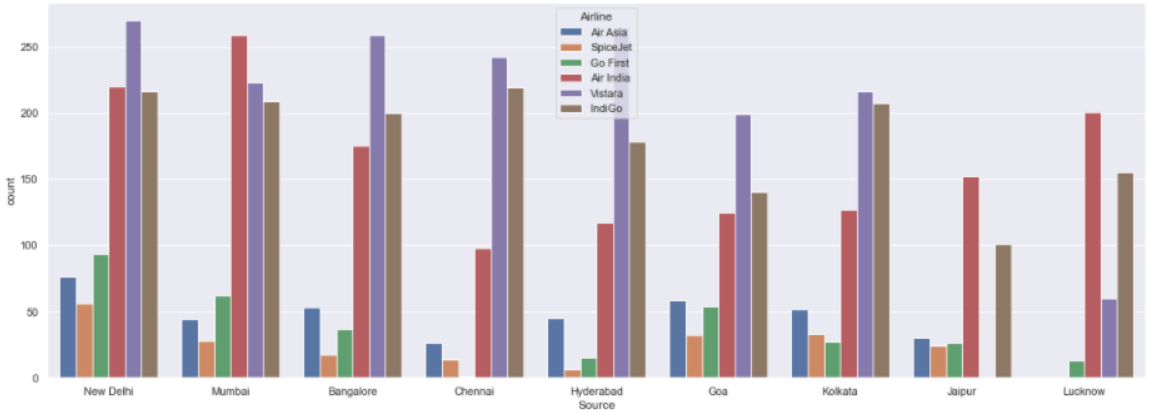
Here we can see that the 0 stop flights availability is around 11.94% of the total record

In domestic flight we rarely see 3 or 4 stop layovers and therefore they cover 4.2% and 0.31% of total rows in the column respectively

Code:

```
#Lets chcek the most popular flight region wise
plt.figure(figsize=(20,7))
sns.countplot(x = "Source", hue = "Airline", data = df)
plt.show()
```

Output:



Observation:

Checking out the Source place details for each and every airline we can see that Mumbai city has the highest number of departure flights for Air India airlines

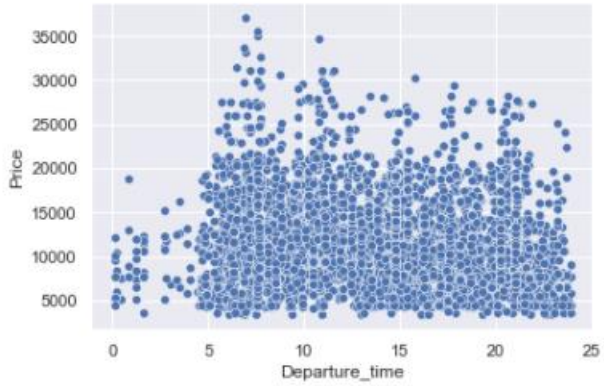
Go First, Indigo and Air India are the airlines that are used in almost all the cities to depart while the other airlines do not cover some or the other city

Looking at the Destination place details for each and every airline we can see that Hyderabad city has the highest number of arrival flights for Air India airlines

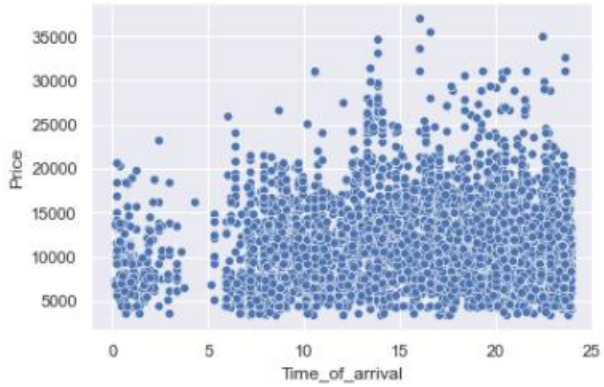
Once again, we can observe that Go First, Indigo and Air India are the leading airlines that are used in almost all cities to arrive while the other airlines miss out on some or the other regions

Overall, I can notice that Air India and Indigo flights do quite well and can be used for arrival and departure to and from any location in India

```
#check the relation between Departure_time and Price
sns.scatterplot(x='Departure_time',y='Price',data=df)
plt.show()
```




```
#check the relation between Time_of_arrival and Price
sns.scatterplot(x='Time_of_arrival',y='Price',data=df)
plt.show()
```



Observation:

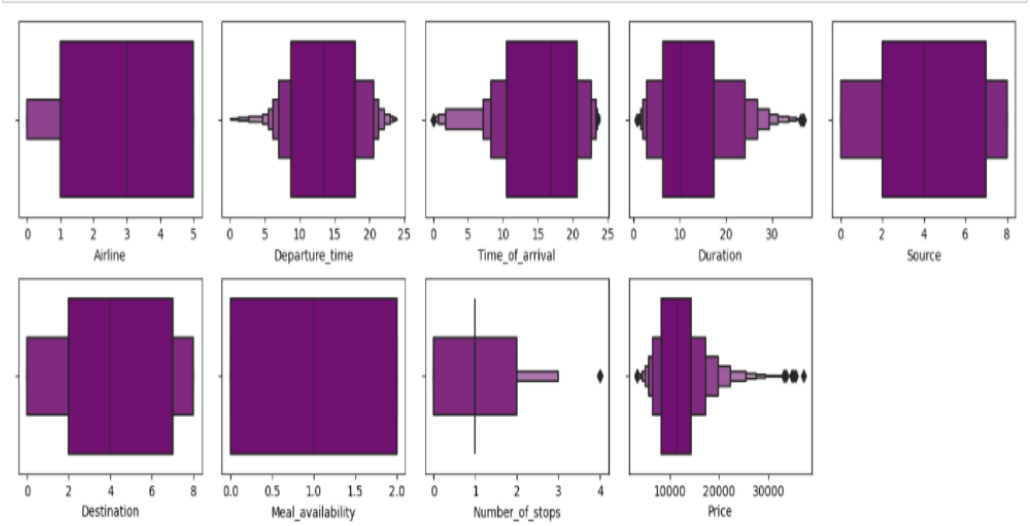
In the above scatter plot, we see the comparison where flight prices are the highest during peak hours as compared to late hours or specifically graveyard timings for departure

We can see a similar trend in flight price scatter plot details when it comes to arrival timings where we see a decrease in prices between 1:00 hrs and 6:00 hrs

Outliers Code:

```
plt.figure(figsize=(14,7))
outl_df = df.columns.values
for i in range(0, len(outl_df)):
    plt.subplot(3, 5, i+1)
    ax = sns.boxenplot(df[outl_df[i]], color='purple')
    plt.tight_layout()
```

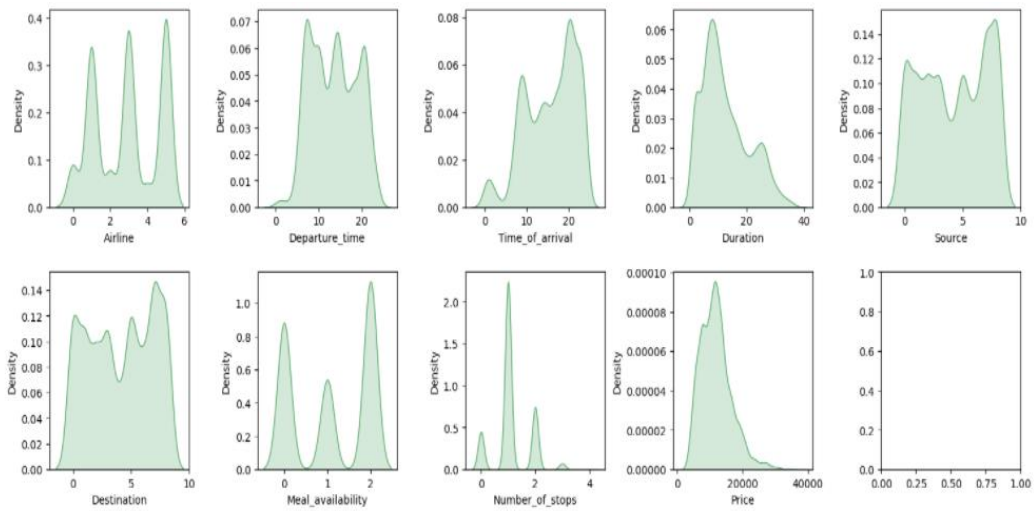
Output:



Skewness Code:

```
fig, ax = plt.subplots(ncols=5, nrows=3, figsize=(15,9))
index = 0
ax = ax.flatten()
for col, value in df.items():
    sns.distplot(value, ax=ax[index], hist=False, color="g", kde_kws={"shade": True})
    index += 1
plt.tight_layout(pad=0.8, w_pad=0.8, h_pad=2.0)
plt.show()
```

Output:



- Interpretation of the Results

  From the above EDA we can easily understand the relationship between features and we can even see which things are affecting the price of flights. These methods take financial, marketing, and various social factors into account to predict flight prices. Nowadays, the number of people using flights has increased significantly. It is difficult for airlines to maintain prices since prices change dynamically due to different conditions. That's why we have tried to use machine learning to solve this problem. This can help airlines by predicting what prices they can maintain. It can also help customers to predict future flight prices and plan their journey accordingly.

CONCLUSION

- ## Key Findings and Conclusions of the Study

  In this project we have scraped the flight data from airline webpages. Then the comma separated value file is loaded into a data frame. Luckily, we don't have any missing values in our data set. Looking at the data set we understand that there are some features needs to be processed like converting the data types and get the actual value from the string entries from the time related columns. After the data is been processed, I have done some EDA to understand the relation among features and the target variable. Features like flight duration, number of stops during the journey and the availability of meals are playing major role in predicting the prices of the flights. As we have seen, the prediction is showing a similar relationship with the actual price from the scrapped data set. This means the model predicted correctly and it 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. The above research will help our client to study the latest flight price market and with the help of the model built he can easily predict the price ranges of the flight, and also will helps him to understand Based on what factors the fight price is decided.

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

  Visualization part helped me to understand the data as it provides graphical representation of huge data. It assisted me to understand the feature importance, outliers/skewness detection and to compare the independent-dependent features. Data cleaning is the most important part of model building and therefore before model building, I made sure the data is cleaned. I have generated multiple

regression machine learning models to get the best model wherein I found Extra Trees Regressor Model being the best based on the metrics I have used.

- Limitations of this work and Scope for Future Work

  As looking at the features we came to know that the numbers of features are very less, due to which we are getting somewhat lower R2 scores. Some algorithms are facing over-fitting problem which may be because of a smaller number of features in our dataset. We can get a better R2 score than now by fetching some more features from the web scraping by that we may also reduce the over fitting problem in our models. Another limitation of the study is that in the volatile changing market we have taken the data, to be more precise we have taken the data at the time of pandemic and recent data, so when the pandemic ends the market correction might happen slowly. So, based on that again the deciding factors of it may change and we have shortlisted and taken these data from the important cities across India. If the customer is from the different country our model might fail to predict the accuracy prize of that flight.