FLIP ROBO

Project Report On
"Flight Price Prediction"

Why do Flight Prices Fluctuate?

Submitted by:

GAURAV MORE

ACKNOWLEDGMENT

I would like to express my sincere thanks of gratitude to my SME as well as "Flip Robo Technologies" team for letting me work on "Flight Price Prediction" project also huge thanks to my academic team "Data Trained". Their suggestions and directions have helped me in the completion of this project successfully. This project also helped me in doing lots of research wherein I came to know about so many new things.

References:

I have also used few external resources that helped me to complete this project successfully. Below are the external resources that were used to create this project.

1. https://www.google.com/
2. https://scikit-learn.org/stable/index.html
3. www.researchgate.net/

TABLE OF CONTENTS:

1. <u>INTRODUCTION</u>

1.1 Business Problem Framing

Airline industry is one of the most sophisticated in its use of dynamic pricing strategies to maximize revenue, based on proprietary algorithms and hidden variables. That is why the airline companies use complex algorithms to calculate the flight ticket prices. 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, arrival time and time of the day it will give the best time to buy the ticket.

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 models 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.

Business goal: The main aim of this project is to predict the price of flight tickets based on various features. 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. So, we will deploy an Machine Learning model for flight ticket price prediction and analysis. This model will provide the approximate selling price for the flight tickets based on different features.

1.2 Conceptual Background of the Domain Problem

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. 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).

Here we are trying to help the buyers to understand the price of the flight tickets by deploying machine learning models. These models would help the sellers/buyers to understand the flight ticket prices in market and accordingly they would be able to book their tickets.

1.3 Review of Literature

Literature review covers relevant literature with the aim of gaining insight into the factors that are important to predict the flight ticket prices in the market. In this study, we discuss various applications and methods which inspired us to build our supervised ML techniques to predict the price of flight tickets in different locations. We did a background survey regarding the basic ideas of our project and used those ideas for the collection of data information by doing web scraping from www.yatra.com, www.makemytrip.com, www.tripodeal.com and www.vimaansafar.com website which is a web platform where buyers can book their flight tickets.

This project is more about data exploration, feature engineering and pre-processing that can be done on this data. Since we scrape huge amount of data that includes more flight related features, we can do better data exploration and derive some interesting features using the available columns. Different techniques like ensemble techniques, and decision trees have been used to make the predictions.

The goal of this project is to build an application which can predict the price of flight tickets with the help of other features. In the long term, this would allow people to better explain and reviewing their purchase in this increasing digital world.

1.4 Motivation for the Problem Undertaken

Air travel is the fastest method of transport around, and can cut hours or days off of a trip. But we know how unexpectedly the prices vary. So, I was interested in Flight Fares Prediction listings to help individuals and find the right

fares based on their needs. And also, to get hands on experience and to know that how the data scientist approaches and work in an industry end to end.

2. <u>ANALYTICAL PROBLEM FRAMING</u>

2.1 Mathematical/ Analytical Modelling of the Problem:

We need to develop an efficient and effective Machine Learning model which predicts the price of flight tickets. So, "Price" is our target variable which is continuous in nature. Clearly it is a Regression problem where we need to use regression algorithms to predict the results. This project is done on three phases:

- **Data Collection Phase:** I have done web scraping to collect the data of flights from the well-known website www.yatra.com, www.makemytrip.com, www.tripodeal.com and www.vimaansafar.com where I found more features of flights compared to other websites and I fetch data for different locations. As per the requirement we need to build the model to predict the prices of flight tickets.
- **Data Analysis:** After cleaning the data I have done some analysis on the data by using different types of visualizations.
- **Model Building Phase:** After collecting the data, I built a machine learning model. Before model building, have done all data pre-processing steps. The complete life cycle of data science that I have used in this project are as follows:
  - Data Cleaning
  - Exploratory Data Analysis
  - Data Pre-processing
  - Model Building
  - Model Evaluation
  - Selecting the best model

2.2 Data Sources and their formats

We have collected the dataset from the website www.yatra.com, www.makemytrip.com, www.tripodeal.com and www.vimaansafar.com which is a web platform where the people can purchase/book their flight tickets. The data is scraped using Web scraping technique and the framework used is Selenium. We scrapped nearly 1712 of the data and fetched the data for different locations and collected the information of different features of the

flights and saved the collected data in excel format. The dimension of the dataset is 1712 rows and 8 columns including target variable "Price". The particular dataset contains both categorical and numerical data type. The data description is as follows:

Variables	Definition
Airline Name	The Name of airline
Departure_time	The time when the journey starts from the source
Arrival Time	Time of arrival at the destination
Duration	Total duration taken by the flight to reach the destination from the source
Source	The source from which the service begins
Destination	The destination where the service ends
Total Stops	Total stops between the source and destination
Price	The price of the flight ticket

2.3 Data Pre-processing Done

Data pre-processing is the process of converting raw data into a well-readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model. I have used following pre-processing steps:

Importing necessary libraries and loading collected dataset as a data frame.

Checked some statistical information like shape, number of unique values present, info, unique (), data types, value count function etc.

Checked null values and found no missing values in the dataset.

Taking care of Timestamp variables by converting data types of "Departure Time" and "Arrival Time" from object data type into datetime data types.

Done feature engineering on some features as they had some irrelevant values like "," and replaced them by empty space.

The column Duration had values in terms of minutes and hours. Duration means the time taken by the plane to reach the destination and it is the difference between the arrival time and Departure time. So, I have extracted proper duration time in terms of float data type from arrival and departure time columns.

Extracted Departure Hour, Departure Min and Arrival Hour, Arrival Min columns from Departure Time and Arrival Time columns and dropped these columns after extraction.

The target variable "price" should be continuous numeric data but due to some string values like "," it was showing as object data type. So, I replaced this sign by empty space and converted into float data type.

From the value count function of Total Stops I found categorical data so replaced them with numeric data according to stops.

Checked statistical description of the data and separated categorical and numeric features.

Performed univariate, bivariate and multivariate analysis to visualize the data. Visualized each feature using seaborn and matplotlib libraries by plotting several categorical and numerical plots like pie plot, count plot, bar plot, reg plot, strip plot, line plot, box plot, boxen plot, distribution plot, and pair plot.

Identified outliers using box plots and found no outliers.

Checked for skewness and removed skewness in numerical column "Duration" using square root transformation method.

Encoded the columns having object data type using Label Encoder method. Used Pearson's correlation coefficient to check the correlation between label and features. With the help of heatmap and correlation bar graph was able to understand the Feature vs Label relativity.

Seperated feature and label data and feature scaling is performed using Standard Scaler method to avoid any kind of data biasness.

2.4 Data Inputs- Logic- Output Relationships

The dataset consists of label and features. The features are independent and label is dependent as the values of our independent variables changes as our label varies.

- Since we had both numerical and categorical columns, I checked the distribution of skewness using dist plots for numerical features and checked the counts using count plots & pie plots for categorical features as a part of univariate analysis.
- To analyse the relation between features and label I have used many plotting techniques where I found numerical continuous variables having some relation with label Price with the help of categorical and line plot.
- I have checked the correlation between the label and features using heat map and bar plot. Where I got both positive and negative correlation

between the label and features. Below is the bar graph to know the correlation between features and label.

Correlation between label and feature using bar plot
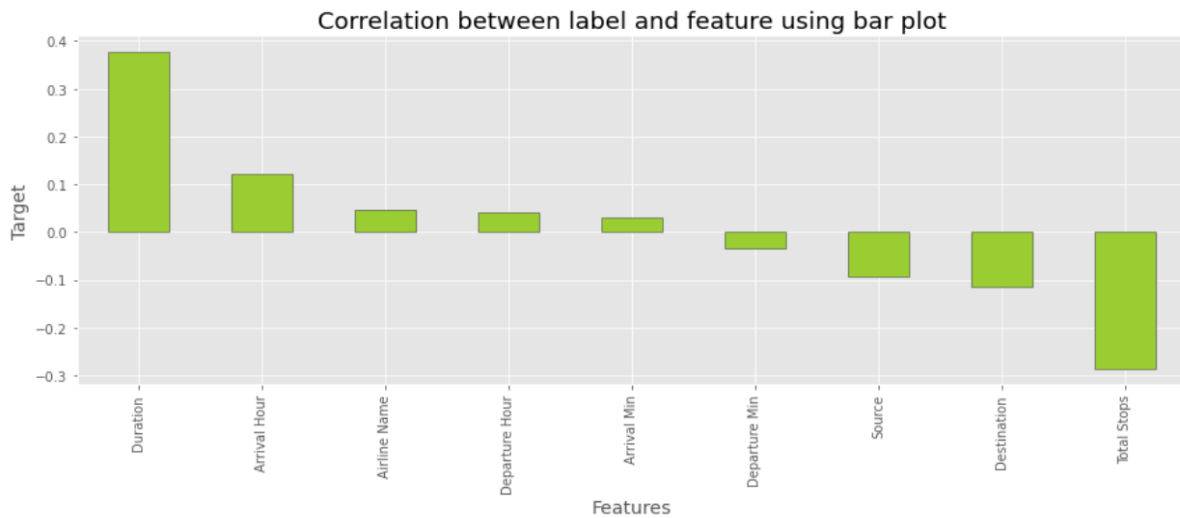
2.5 Hardware & Software Requirements & Tools Used

To build the machine learning projects it is important to have the following hardware and software requirements and tools.

Hardware required:

- Processor: core i3
- RAM: 8 GB or above
- ROM/SSD: 250 GB or above

Software required:

- Distribution: Anaconda Navigator
- Programming language: Python
- Browser based language shell: Jupyter Notebook
- Chrome: To scrape the data

Libraries required:

```python
# Preprocessing
import numpy as np
import pandas as pd
# Visualization
import seaborn as sns
import matplotlib.pyplot as plt
import os
import scipy as stats
from scipy.stats import zscore   # To remove outliers
# Evaluation Metrics
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import cross_val_score
from sklearn.metrics import r2_score
from sklearn import metrics
# ML Algorithms
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor,ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
import xgboost as xgb
from sklearn.model_selection import GridSearchCV
import warnings
%matplotlib inline
warnings.filterwarnings('ignore')
```

✔ **import numpy as np:** It is defined as a Python package used for performing the various numerical computations and processing of the multidimensional and single dimensional array elements. The calculations using Numpy arrays are faster than the normal Python array.

✔ **import pandas as pd:** Pandas is a Python library that is used for faster data analysis, data cleaning and data pre-processing. The data-frame term is coming from Pandas only.

✔ **import matplotlib.pyplot as plt:** Matplotlib and Seaborn acts as the backbone of data visualization through Python.

   Matplotlib: It is a Python library used for plotting graphs with the help of other libraries like Numpy and Pandas. It is a powerful tool for visualizing data in Python. It is used for creating statical interferences and plotting 2D graphs of arrays.

✔ **import seaborn as sns: Seaborn** is also a Python library used for plotting graphs with the help of Matplotlib, Pandas, and Numpy. It is built on the

roof of Matplotlib and is considered as a superset of the Matplotlib library. It helps in visualizing univariate and bivariate data.

With the above sufficient libraries, we can perform pre-processing, data cleaning and can build ML models.

3. MODEL/S DEVELOPMENT AND EVALUATION

3.1 Identification of possible Problem-solving approaches (Methods):

I have used both statistical and analytical approaches to solve the problem which mainly includes the pre-processing of the data also used EDA techniques and heat map to check the correlation of independent and dependent features. Removed skewness using square root transformation. Encoded data using Label Encoder. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Checked for the best random state to be used on our Regression Machine Learning model pertaining to the feature importance details. Finally created multiple regression models along with evaluation metrics.

For this particular project we need to predict flight ticket prices. In this dataset, "Price" is the target variable, which means our target column is continuous in nature so this is a regression problem. I have used many regression algorithms and predicted the flight ticket price. By doing various evaluations I have selected Gradient Boosting Regressor as best suitable algorithm to create our final model as it is giving high R2 score and low evaluation error among all the algorithms used. Performed hyper parameter tuning on best model. Then I saved my final model and loaded the same for predictions.

3.2 Testing of Identified Approaches (Algorithms)

Since "Price" is my target variable which is continuous in nature, from this I can conclude that it is a regression type problem hence I have used following regression algorithms. After the pre-processing and data cleaning I left with 11 columns including target and with the help of feature importance bar graph I used these independent features for model building and prediction. The algorithms used on training the data are as follows:

1. Linear Regression
2. Lasso Regressor
3. Ridge Regressor
4. Decision Tree Regressor
5. KNeighbors Regressor
6. Support Vector Regressor
7. Random Forest Regressor
8. Ada Boost Regressor
9. Gradient Boosting Regressor
10. Extreme Gradient Boosting Regressor (XGB)
11. Stochastic Gradient Descent Regressor(SGD)

3.3 Run and evaluate selected models

I have used 11 regression algorithms after choosing random state amongst 1-1000 number.

Model Building:

1. Linear Regression:

Linear Regression is a machine learning algorithm based on supervised learning. It is a model that assumes a linear relationship between the input variables (x) and the single output variable (y) i.e Y=bX+c

```python
from sklearn.linear_model import LinearRegression

for i in range(0,1000):
    x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=i,test_size=0.20)
    lr.fit(x_train,y_train)
    pred_train=lr.predict(x_train)
    pred_test=lr.predict(x_test)
    if round(r2_score(y_train,pred_train)*100,1)==round(r2_score(y_test,pred_test)*100,1):
        print("At random state:",i)
        print("Training r2_score is:",r2_score(y_train,pred_train*100))
        print("Testing r2_score is:",r2_score(y_test,pred_test*100))

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_test))
print('Mean Squared Error:',mean_squared_error(y_test,pred_test))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_test)))
```
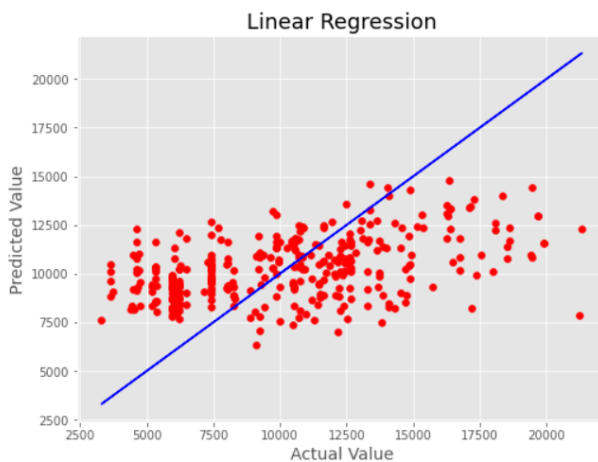
❖ Created Linear Regression model and checked for its evaluation metrics. The model is giving R2 score as -68136.05710272539%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.
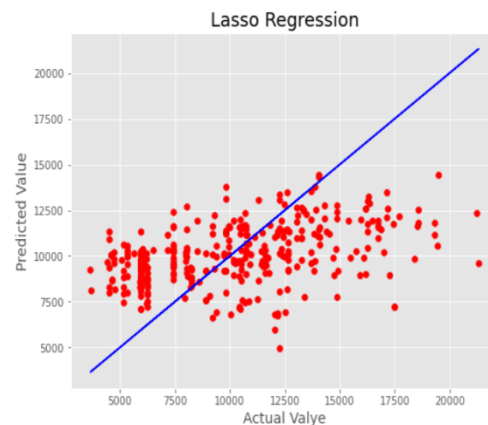

2. Lasso Regressor:

Least Absolute Shrinkage and Selection Operator (Lasso) is a regression technique. Lasso is a powerful technique that performs regularisation and feature selection. In Lasso instead of squaring the slope like Ridge regression, the absolute value of the slope is added as a penalty term.

```python
from sklearn.linear_model import Lasso
ls = Lasso(alpha=10,random_state=0)
ls.fit(x_train,y_train)
ls.score(x_train,y_train)
pred_ls = ls.predict(x_test)

lss = r2_score(y_test,pred_ls)
for j in range(2,10):
    lsscore = cross_val_score(ls,x,y,cv=j)
    lsc = lsscore.mean()
    print("At cv:-",j)
    print("Cross validation score is:-",lsc*100 )
    print("R2_score is :-",lss*100)
    print("\n")
```

```
At cv:- 2
Cross validation score is:- -27.56906606329278
R2_score is :- 19.20716755028432
```


❖ Created Lasso Regressor model and checked for its evaluation metrics. The model is giving R2 score as 19.20%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

3. Ridge Regressor:

The Ridge Regression is supervised regression technique. It is an estimation procedure to manage collinearity without removing variables from the regression model. In multiple linear regression, the multicollinearity is a common problem that leads least square estimation to be unbiased, and its variances are far from the correct value.

Therefore, by adding a degree of bias to the regression model, Ridge Regression reduces the standard errors, and it shrinks the least square coefficients towards the origin of the parameter space.

```python
from sklearn.linear_model import Ridge
rd = Ridge(alpha=0.01, copy_X= True, fit_intercept= True, normalize=True, random_state= 0, tol= 0.001)
rd.fit(x_train,y_train)
rd.score(x_train,y_train)
pred_rd = rd.predict(x_test)

rds = r2_score(y_test,pred_rd)
print('R2 Score:',rds*100)

rdscore = cross_val_score(rd,x,y,cv=4)
rdc = rdscore.mean()
print('Cross Val Score:',rdc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_rd))
print('Mean Squared Error:',mean_squared_error(y_test,pred_rd))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_rd)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_rd, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('Ridge Regression',fontsize=18)
plt.show()
```
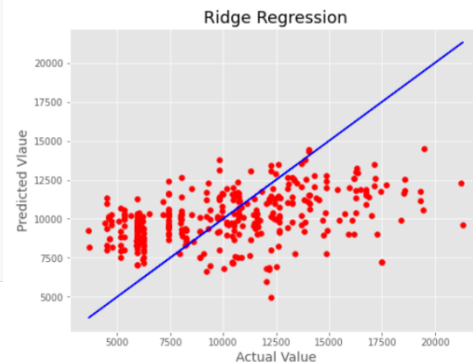
```
R2 Score: 19.14865937404404
Cross Val Score: -12.525023981868976
Error:
Mean Absolute Error: 2852.0803541780824
Mean Squared Error: 12101877.250081727
Root Mean Square Error: 3478.7752514472286
```


❖ Created Ridge Regressor model and checked for its evaluation metrics. The model is giving R2 score as 19.14%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

4. Decision Tree Regressor:

Decision Tree Regressor is a decision-making tool that uses a flowchart like tree structure. It observes features of an object and trains a model in the structure of a tree to predict data in the future to produce meaningful continuous output.

```python
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor(criterion='mse',splitter='best')
dtr.fit(x_train,y_train)
dtr.score(x_train,y_train)
pred_dtr = dtr.predict(x_test)

dtr_r2 = r2_score(y_test,pred_rd)
print('R2 Score:',dtr_r2*100)

dtrscore = cross_val_score(dtr,x,y,cv=4)
dtrcc = dtrscore.mean()
print('Cross Val Score:',dtrcc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_dtr))
print('Mean Squared Error:',mean_squared_error(y_test,pred_dtr))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_dtr)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_dtr, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('Decision Tree Regressor',fontsize=18)
plt.show()
```

```
R2 Score: 19.14865937404404
Cross Val Score: 21.128644911162077
Error:
Mean Absolute Error: 971.4985337243402
Mean Squared Error: 4731741.793255132
Root Mean Square Error: 2175.2567189311544
```

❖ Created Decision Tree Regressor model and checked for its evaluation metrics. The model is giving R2 score as 19.14%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

5. KNeighbors Regressor:

KNeighbors regressor is a supervised regression technique based on k-nearest neighbors. The target is predicted by local interpolation of the targets associated of the nearest neighbors in the training set.

```python
knr=KNeighborsRegressor(n_neighbors=5,algorithm='auto')
knr.fit(x_train,y_train)
knr.score(x_train,y_train)
pred_knr = knr.predict(x_test)

knr_r2 = r2_score(y_test,pred_knr)
print('R2 Score:',knr_r2*100)

knrscore = cross_val_score(knr,x,y,cv=4)
knrcc = knrscore.mean()
print('Cross Val Score:',knrcc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_knr))
print('Mean Squared Error:',mean_squared_error(y_test,pred_knr))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_knr)))

#Visualizing the Predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_knr, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('KNeighbors Regressor',fontsize=18)
plt.show()
```
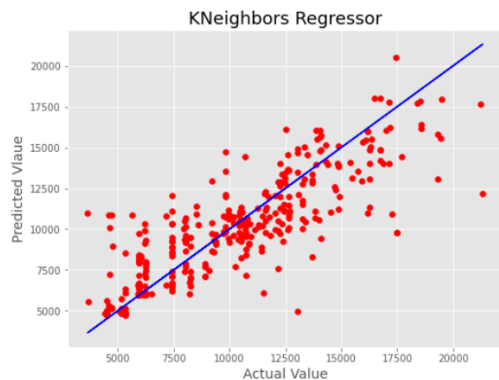
```
R2 Score: 69.31875965240464
Cross Val Score: 20.72445348968129
Error:
Mean Absolute Error: 1503.7149560117302
Mean Squared Error: 4592386.4921994135
Root Mean Square Error: 2142.985415769182
```


❖ Created KNeighbors Regressor model and checked for its evaluation metrics. The model is giving R2 score as 69.31%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

6. Support Vector Regressor:

Support-vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for regression, classification or other tasks like outliers detection. Intuitively, a good separation is achieved by the hyperplane that has the largest distance to the nearest training-data point of any class (so-called functional margin), since in general the larger the margin, the lower the generalization error of the regressor.

```python
from sklearn.svm import SVR
svr=SVR()
svr.fit(x_train,y_train)
svr.score(x_train,y_train)
pred_svr = svr.predict(x_test)

svr_r2 = r2_score(y_test,pred_svr)
print('R2 Score:',svr_r2*100)

svr_score = cross_val_score(svr,x,y,cv=4)
svr_cc = svr_score.mean()
print('Cross Val Score:',svr_cc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_svr))
print('Mean Squared Error:',mean_squared_error(y_test,pred_svr))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_svr)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_svr, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('Support Vector Regressor',fontsize=18)
plt.show()
```

```
R2 Score: 0.5658674814917442
Cross Val Score: -16.37907783898055
Error:
Mean Absolute Error: 3207.585903677629
Mean Squared Error: 14883360.707330497
Root Mean Square Error: 3857.895891199048
```


❖ Created Support Vector Regression model and checked for its evaluation metrics. The model is giving R2 score as 0.5658%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

7. Random Forest Regressor:

Random forest Regressor is an ensemble technique capable of performing both regression and classification tasks with use of multiple decision trees and a technique called Bootstrap Aggregation. It improves the predictive accuracy and control over-fitting.

```
from sklearn.ensemble import RandomForestRegressor

rf=RandomForestRegressor(n_estimators=100,criterion='mse')
rf.fit(x_train,y_train)
rf.score(x_train,y_train)
pred_rf = rf.predict(x_test)

rf_r2 = r2_score(y_test,pred_rd)
print('R2 Score:',rf_r2*100)

rf_score = cross_val_score(rf,x,y,cv=4)
rf_cc = rf_score.mean()
print('Cross Val Score:',rf_cc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_rf))
print('Mean Squared Error:',mean_squared_error(y_test,pred_rf))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_rf)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_rf, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('Random Forest Regressor',fontsize=18)
plt.show()

R2 Score: 19.14865937404404
Cross Val Score: 56.332637721611
Error:
Mean Absolute Error: 806.7926585090537
Mean Squared Error: 2212514.0027945195
Root Mean Square Error: 1487.4521850447898
```
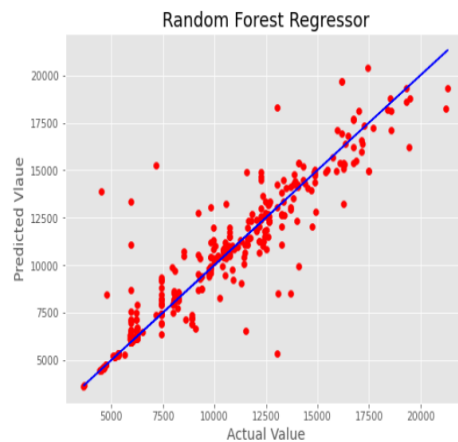

Random Forest Regressor

❖ Created Random Forest Regressor model and checked for it evaluation metrics. The model is giving R2 score as 19.14%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

8. Ada Boost Regressor:

An **AdaBoost regressor** is a meta-estimator that begins by fitting a regressor on the original dataset and then fits additional copies of the regressor on the same dataset but where the weights of instances are adjusted according to the error of the current prediction.

```python
from sklearn.ensemble import AdaBoostRegressor

ada=AdaBoostRegressor(n_estimators=50)
ada.fit(x_train,y_train)
ada.score(x_train,y_train)
pred_ada = ada.predict(x_test)

ada_r2 = r2_score(y_test,pred_ada)
print('R2 Score:',ada_r2*100)

ada_score = cross_val_score(ada,x,y,cv=4)
ada_cc = ada_score.mean()
print('Cross Val Score:',ada_cc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_ada))
print('Mean Squared Error:',mean_squared_error(y_test,pred_ada))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_ada)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_ada, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('Ada Boost Regressor',fontsize=18)
plt.show()
```
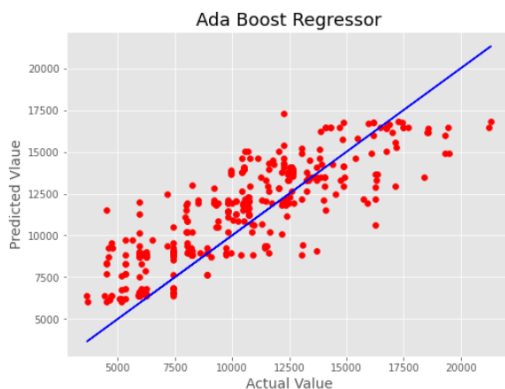
```
R2 Score: 66.30040014666312
Cross Val Score: 40.99171297649316
Error:
Mean Absolute Error: 1813.37779450608
Mean Squared Error: 5044176.356811437
Root Mean Square Error: 2245.924388044138
```


❖ Created Ada Boost Regressor model and checked for its evaluation metrics. The model is giving R2 score as 66.30%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

9. Gradient Boosting Regressor:

Gradient Boosting Regressor is also works for both numerical as well as categorical output variables. It produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. This model was chosen to account for non-linear relationships between the features & predicted price, by splitting the data into 100 regions.

```
: from sklearn.ensemble import GradientBoostingRegressor
  gbr= GradientBoostingRegressor(criterion='friedman_mse',loss='huber',n_estimato
  gbr.fit(x_train, y_train)
  gbr.score(x_train, y_train)
  pred_gbr =gbr.predict(x_test)

  gbrs= r2_score(y_test,pred_gbr)
  print('R2 Score:',gbrs*100)

  gbscore = cross_val_score(gbr,x,y,cv=4)
  gbrc= gbscore.mean()
  print('Cross Val Score:',gbrc*100)

  print('Error:')

  print('Mean Absolute Error:',mean_absolute_error(y_test,pred_gbr))
  print('Mean Squared Error:',mean_squared_error(y_test,pred_gbr))
  print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_gbr)))

  #Visualizing the predicted values
  plt.figure(figsize=(8,6))
  plt.scatter(x=y_test, y=pred_gbr, color='r')
  plt.plot(y_test,y_test, color='b')
  plt.xlabel('Actual Value',fontsize=14)
  plt.ylabel('Predicted Vlaue',fontsize=14)
  plt.title('Gradient Boosting Regressor',fontsize=18)
  plt.show()

  R2 Score: 87.01257531305733
  Cross Val Score: 56.836487303923164
  Error:
  Mean Absolute Error: 1023.6489600746621
  Mean Squared Error: 1943965.531544988
  Root Mean Square Error: 1394.2616438620794
```
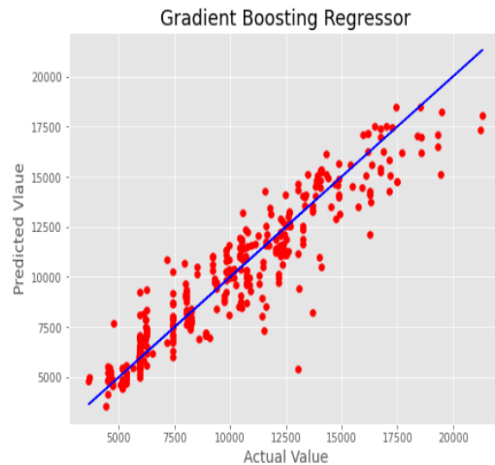

Gradient Boosting Regressor

❖ Created Gradient Boosting Regressor model and checked for its evaluation metrics. The model is giving R2 score as 87.01%.
❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

10.Extreme Gradient Boosting Regressor (XGB Regressor):

XGB Regressor is a popular supervised machine learning model and it is an implementation of Gradient Boosting trees algorithm. It is best known to provide better solutions than other machine learning algorithms.

```
import xgboost

xgb = xgboost.XGBRegressor()
xgb.fit(x_train,y_train)
xgb.score(x_train,y_train)
pred_xgb = xgb.predict(x_test)

xgbs= r2_score(y_test,pred_xgb)
print('R2 Score:',xgbs*100)

xgbscore = cross_val_score(xgb,x,y,cv=4)
xgbc= xgbscore.mean()
print('Cross Val Score:',xgbc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_xgb))
print('Mean Squared Error:',mean_squared_error(y_test,pred_xgb))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_xgb)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_xgb, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('XGB Regressor',fontsize=18)
plt.show()

R2 Score: 85.62161127973611
Cross Val Score: 50.405563109772565
Error:
Mean Absolute Error: 900.8290414222874
Mean Squared Error: 2152165.86391063
Root Mean Square Error: 1467.0261974179705
```


XGB Regressor

❖ Created XGB Regressor model and checked for its evaluation metrics. The model is giving R2 score as 85.62%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

11.Stochastic Gradient Descent (SGD) Regressor:

The **SGDRegressor** implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties to fit linear regression models. SGDRegressor is well suited for regression problems with a large number of training samples.

```
: from sklearn.linear_model import SGDRegressor
sgd=SGDRegressor()
sgd.fit(x_train,y_train)
sgd.score(x_train,y_train)
pred_sgd = sgd.predict(x_test)

sgd_r2 = r2_score(y_test,pred_sgd)
print('R2 Score:',sgd_r2*100)

sgd_score = cross_val_score(sgd,x,y,cv=2)
sgd_cc = sgd_score.mean()
print('Cross Val Score:',sgd_cc*100)

print('Error:')

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_sgd))
print('Mean Squared Error:',mean_squared_error(y_test,pred_sgd))
print('Root Mean Square Error:',np.sqrt(mean_squared_error(y_test,pred_sgd)))

#Visualizing the predicted values
plt.figure(figsize=(8,6))
plt.scatter(x=y_test, y=pred_sgd, color='r')
plt.plot(y_test,y_test, color='b')
plt.xlabel('Actual Value',fontsize=14)
plt.ylabel('Predicted Vlaue',fontsize=14)
plt.title('SGD Regressor',fontsize=18)
plt.show()

R2 Score: 19.106503605037084
Cross Val Score: -27.99802657557346
Error:
Mean Absolute Error: 2858.5354235662826
Mean Squared Error: 12108187.150918934
Root Mean Square Error: 3479.6820473886596
```
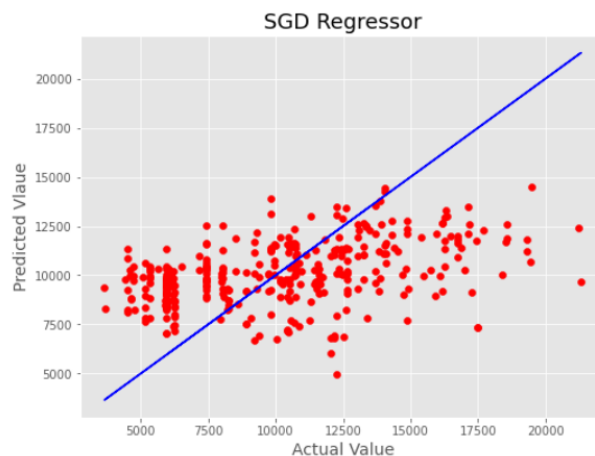

❖ Created SGDRegressor model and checked for its evaluation metrics. The model is giving R2 score as 19.10%.

❖ From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

Model Selection:

From the above created models, we can conclude that "Gradient Boosting Regressor" as the best fitting model as it is giving high R2 score and low MAE, MSE and RMSE values.

Saving the final model and predicting the flight ticket price

```
import pickle
filename = 'flight-price.pkl'
pickle.dump(gbr, open(filename, 'wb'))
```

3.4 Key Metrics for success in solving problem under consideration:

The essential step in any machine learning model is to evaluate the accuracy and determine the metrics error of the model. I have used Mean Absolute Error (MAE), Mean Squared Error (MSE), Root Mean Squared Error (RMSE) and R2 Score metrics for my model evaluation

❖ **Mean Absolute Error (MAE):** MAE is a popular error metric for regression problems which gives magnitude of absolute difference between actual and predicted values. The MAE can be calculated as follows:
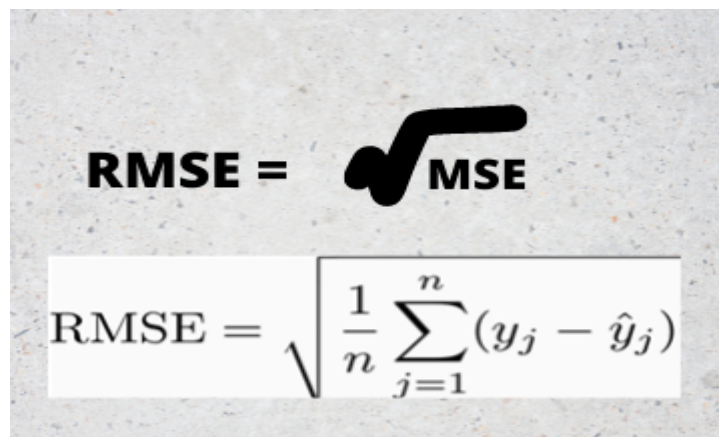

❖ **Mean Squared Error (MSE):** MSE is a most used and very simple metric with a little bit of change in mean absolute error. Mean squared error states that finding the squared difference between actual and predicted value. We perform squared to avoid the cancellation of negative terms and it is the benefit of MSE.

$$MSE = \frac{1}{n} \Sigma \underbrace{\left(y - \widehat{y} \right)}_{\text{The square of the difference between actual and predicted}}^{2}$$

❖ **Root Mean Squared Error (RMSE):** RMSE is an extension of the mean squared error. The square root of the error is calculated, which means that the units of the RMSE are the same as the original units of the target value that is being predicted.

$$RMSE = \sqrt{MSE}$$

$$RMSE = \sqrt{\frac{1}{n} \sum_{j=1}^{n} (y_j - \hat{y}_j)}$$

❖ **R2 Score:** I have used R2 score which gives the accurate value for the models used. On the basis of R2 score I have created final model.

3.5 Visualizations

I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. I have analysed the data using univariate, bivariate and multivariate analysis. In univariate analysis I have used distribution plot, pie plot and count plot and in bivariate analysis I have used bar plots, strip plots, box plots and boxen plots to get the relation between categorical variables and target column Price and used line plots, reg plot, box plot, bar plot, boxen plot and factor plot to understand the relation between continuous numerical variables and target

variable. Apart from these plots I have used pair plot (multivariate analysis) and box plots to get the insight from the features.

Univariate Analysis: Univariate analysis is the simplest way to analyse data. "Uni" means one and this means that the data has only one kind of variable. The major reason for univariate analysis is to use the data to describe. The analysis will take data, summarise it, and then find some pattern in the data. Mainly we will get the counts of the values present in the features.

<u>Univariate Analysis for Numerical Variables:</u>

Visualizing the distribution of skewness in numerical variables
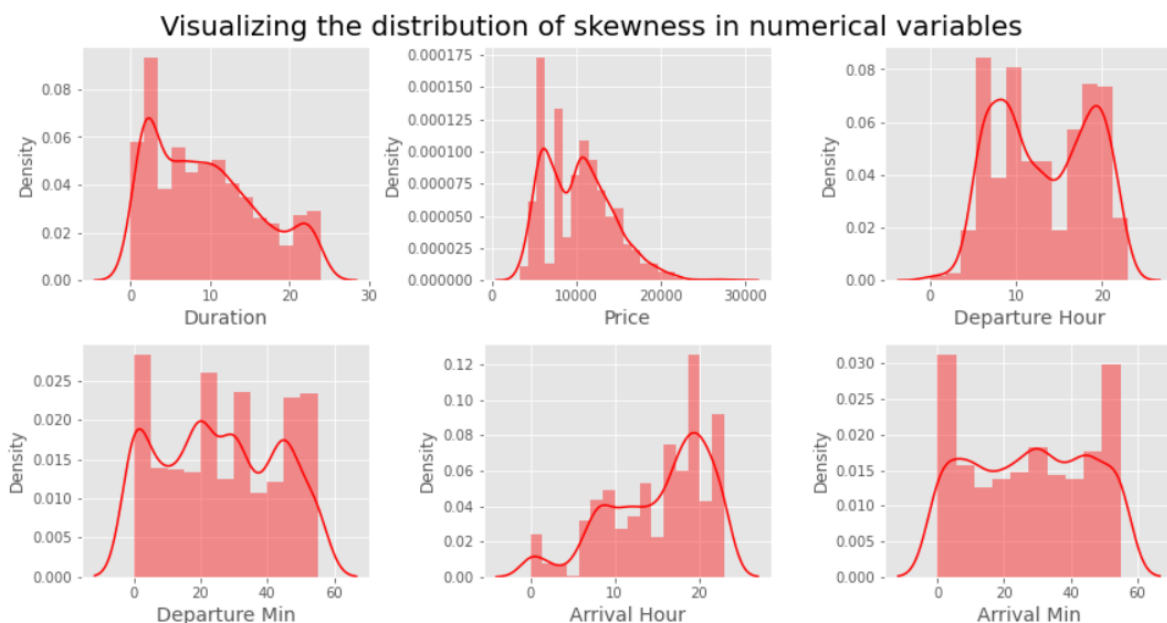
Observations:

Above plot shows how the data has been distributed in each of the columns.

✔ From the distribution plot we can observe the columns are somewhat distributed normally as they have no proper bell shape curve.
✔ Since there is presence of skewness in the data, we need to remove skewness in the numerical columns to overcome with any kind of data biasness.

<u>Univariate Analysis for Categorical Variables:</u>

Airline Name

Source

Destination
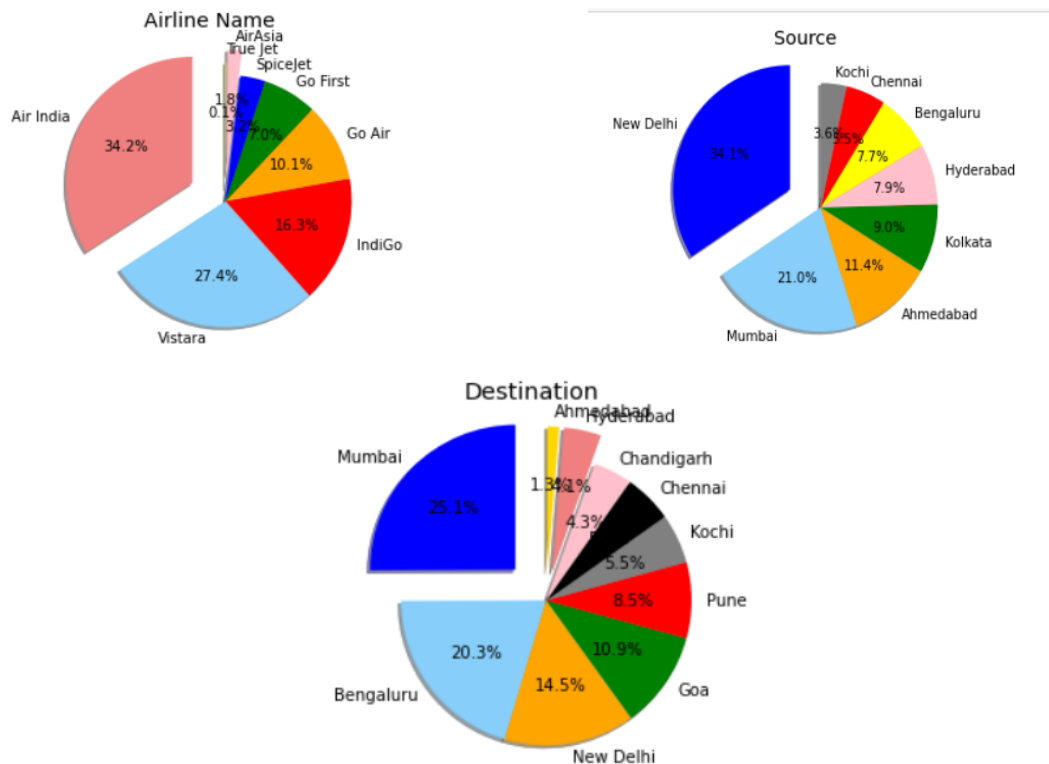
Observations:

✔ **Airline:** From the pie plot we can infer that there are greater number of flights of "Air India", "Vistara" and "Indigo" compared to others. Also, the count of True jet flight is very less.

✔ **Source:** From the count plot we can observe greater number of flights are from Mumbai, New Delhi, Ahmedabad and Kolkata. Only few flights are from Kochi and Chennai.

✔ **Destination:** More number of flights are heading towards Mumbai, Bengaluru, New Delhi andGoa. Only few flights are travelling to Hyderabad and Ahmedabad.

Bivariate Analysis: Bivariate analysis is one of the statistical analyses where two variables are observed, Bi means two variables. One variable here is dependent while the other is independent. These variables are usually denoted by X and Y. We can also analyse the data using both independent variables. Bivariate analysis is finding some kind of empirical relationship between two variables using different plotting techniques.

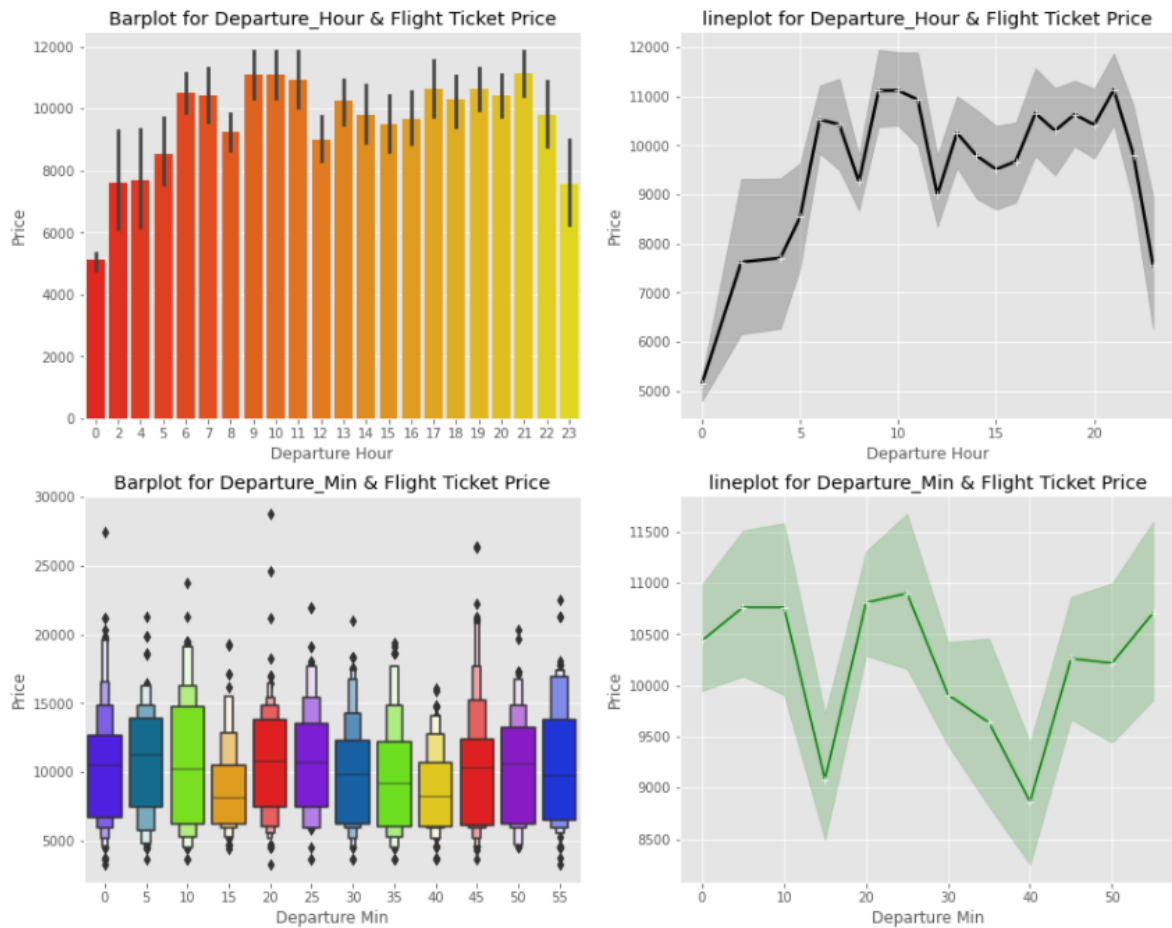## Visualizing Categorical Variables vs Target Variable Price

Visualizing Categorical Variables vs Target Variable Price
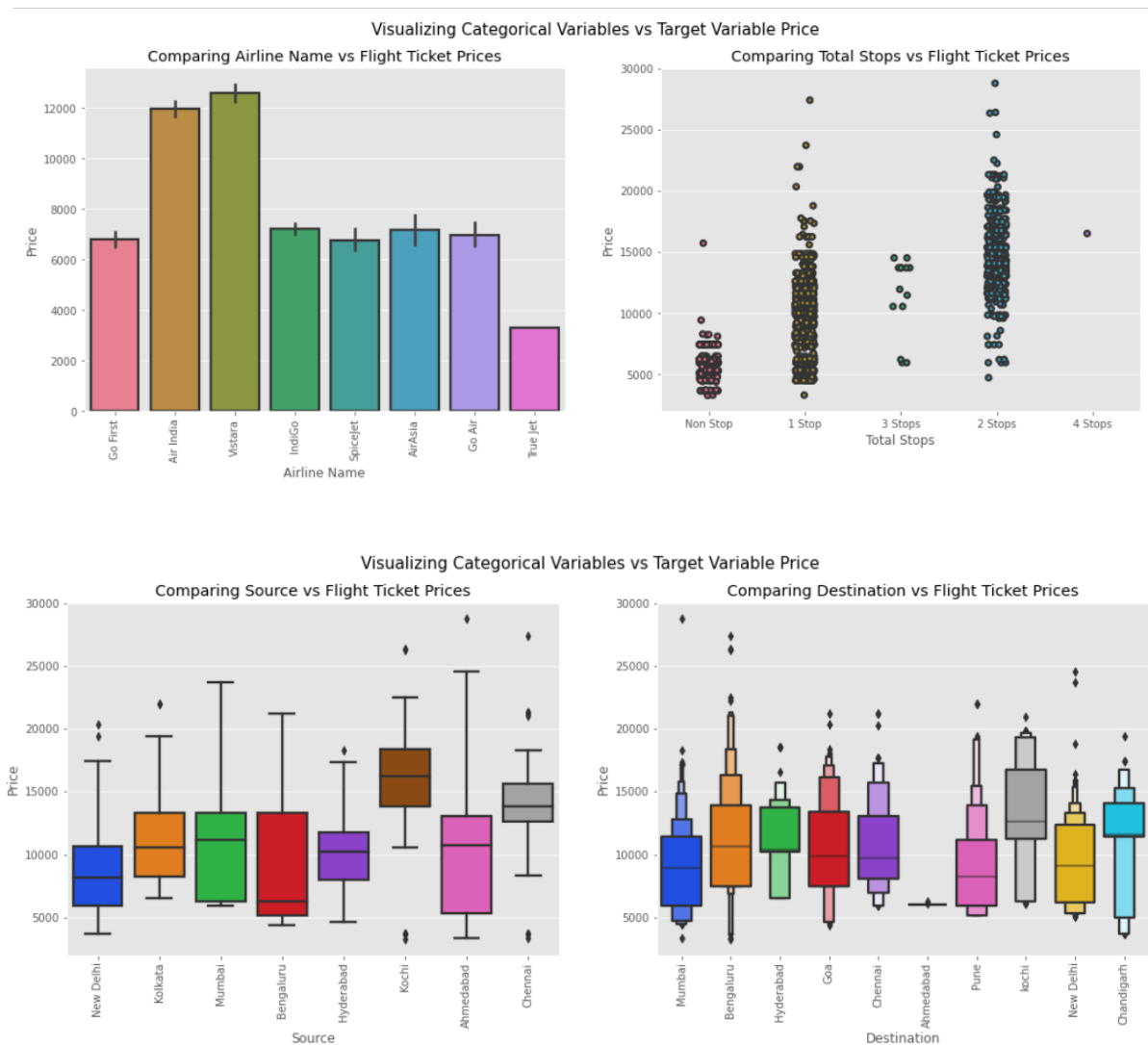
Observations:

✔ **Airline vs Price:** From the bar plot we can notice "Vistara" and "Air India" airlines have highest ticket prices compared to other airlines.

✔ **Total Stops vs Price:** From the strip plot we can notice the flights which have 1 and 2 stops between source and destination have highest ticket prices compared to others. The airlines which have 4 stops during the journey have very less ticket price. So, we can say as the stops increases, ticket price decreases.

✔ **Source vs Price:** From the box plot we can observe the flights from Kochi are having somewhat higher prices compared to other sources.

✔ **Destination va Price:** From the boxen plot we can notice that the flights travelling to Bengaluru have higher flight ticket prices.

# Visualizing Numerical Variables vs Target Variable Price

Observations:

✔ **Departure Hour vs Price:** From the bar plot and line plot we can see that there are some flights departing in the early morning having most expensive ticket prices compared to late morning flights. We can also observe the flight ticket prices are higher during afternoon (may fluctuate) and it decreases in the evening.

✔ **Departure Min vs Price:** The boxen plot and line plot gives there is no significant difference between price and departure min.

Visualizing Numerical Variables vs Target Variable Price
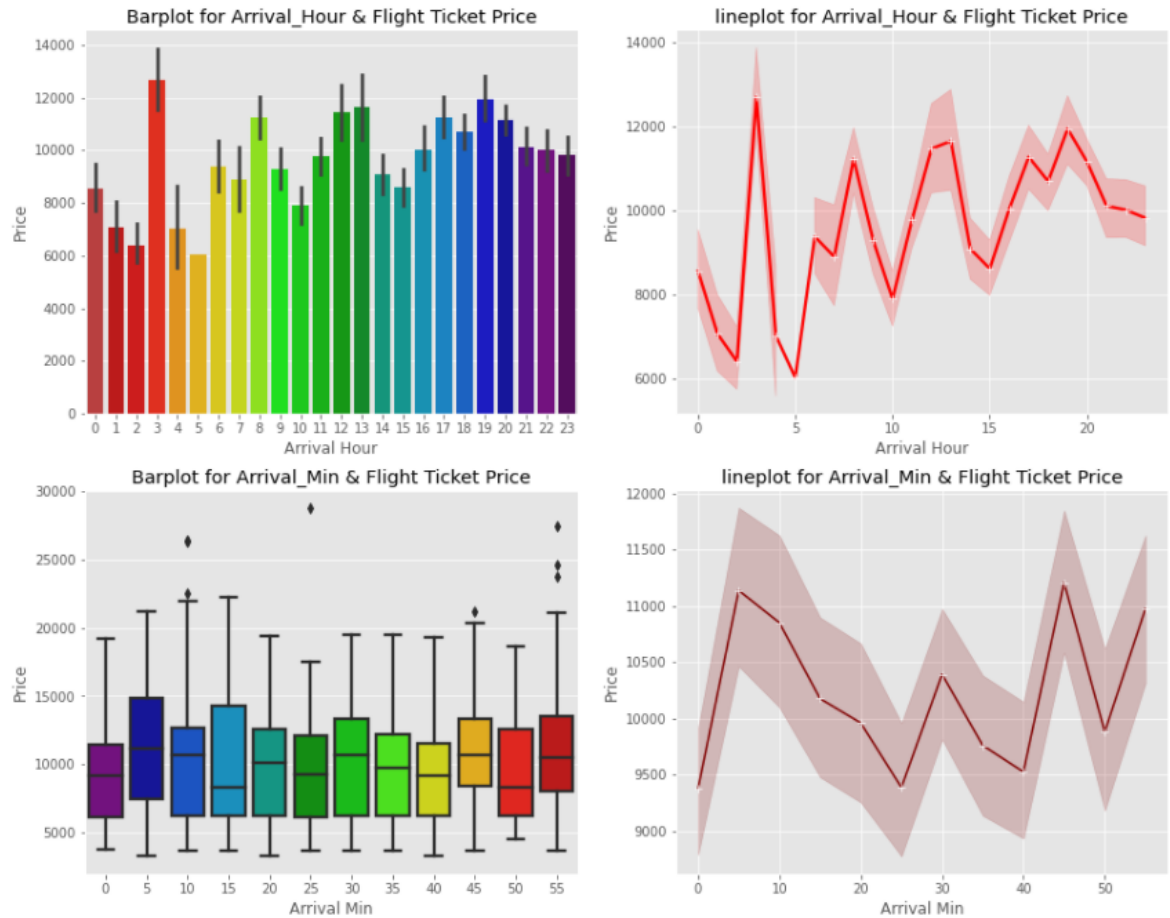
Observations:

✔ **Arrival Hour vs Price:** From the bar plot and line plot we can observe that very few flights are arriving in the early morning that is 0 to 6 AM they have very less ticket price. Also, the flights which are arriving in the afternoon and evening have somewhat higher price. So, we can conlude this column has some positive correlation with price.

✔ **Arrival Min vs Price:** There is no significant difference between this feature and price. We can say flight ticket prices are not much dependent on the Arrival min.

Visualizing Numerical Variables vs Target Variable Price
Comparing Duration & Flight Ticket Price
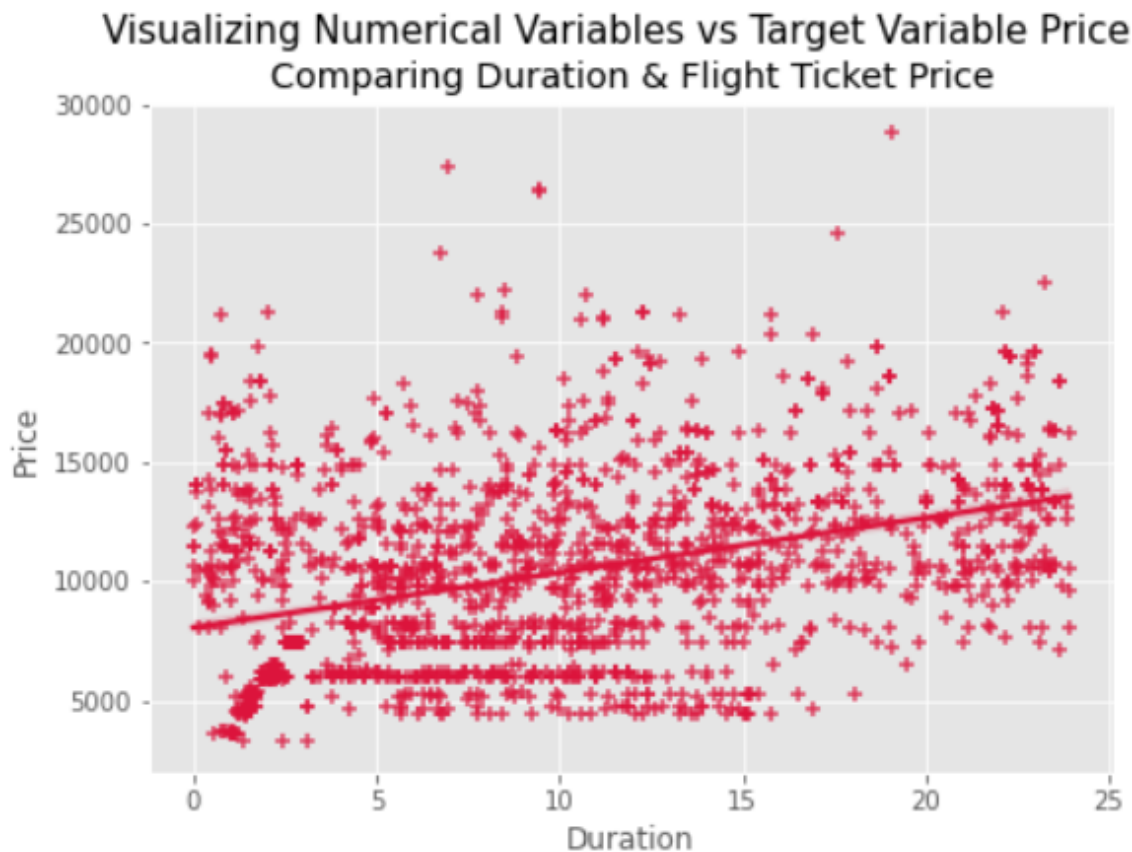
✔ **Duration vs Price:** From the reg plot we can observe some positive linear relation between Duration and Price. Flights having 1-12 hours of duration, they have ticket price of around 15000.

Checking most popular flights region wise
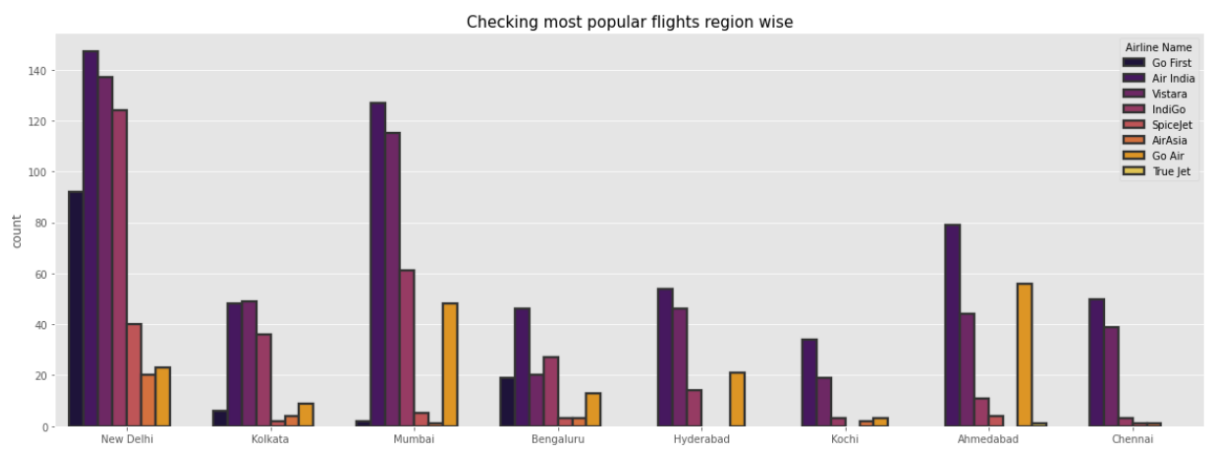
Observations:

☐ The plot showing the region wise count of airlines shows that New Delhi, Kolkata, Mumbai and Bengaluru Source is not having True jet flight,

Hyderabad source is not having Go First, SpiceJet,AirAsia and True Jet flights, Kochi is not having Go First,SpiceJet and True jet flights, Ahmedabad is not having Go FirstAirAsia flights, Chennai is not having Go First,Go Air and True jet flights. Air India flights are in higher count in almost all the regions except Kolkata. Kolkata has Vistara flights in higher count.

3.6 Interpretation of the Results

Visualizations: In univariate analysis I have used count plots and pie plots to visualize the counts in categorical variables and distribution plot to visualize the numerical variables. In bivariate analysis I have used bar plots, strip plots, line plots, reg plots, box plots, and boxen plots to check the relation between label and the features. Used pair plot to check the pairwise relation between the features. The heat map and bar plot helped me to understand the correlation between dependent and independent features. Detected outliers and skewness with the help of box plots and distribution plots respectively. And I found some of the features skewed to right as well as to left. I got to know the count of each column using bar plots.

Pre-processing: The dataset should be cleaned and scaled to build the ML models to get good predictions. I have performed few processing steps which I have already mentioned in the pre-processing steps where all the important features are present in the dataset and ready for model building.

Model building: After cleaning and processing data, I performed train test split to build the model. I have built multiple regression models to get the accurate R2 score, and evaluation metrics like MAE, MSE and RMSE. I got Gradient Boosting Regressor as the best model which gives 87.01% R2 score. Finally, I saved my final model and got the good predictions results for price of flight tickets.

4.<u>CONCLUSION</u>

4.1 Key Findings and Conclusions of the Study

The case study aims to give an idea of applying Machine Learning algorithms to predict the price of the flight tickets. After the completion of this project, we got an insight of how to collect data, pre-processing the data, analyse the data, cleaning the data and building a model.

 In this study, we have used multiple machine learning models to predict the flight ticket price. We have gone through the data analysis by performing feature engineering, finding the relation between features and label through visualizations. And got the important feature and we used these features to predict the car price by building ML models. We have also got good prediction results of ticket price.

Findings:

1.Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time?

> Flight ticket prices change during the morning and evening time of the day. From the distribution plots we came to know that the prices of the flight tickets are going up and down, they are not fixed at a time. Also, from this graph we found prices are increasing in large amounts. Some flights are departing in the early morning having most expensive ticket prices compared to late morning flights. As the time goes the flight ticket fares increased and midnight flight fares are very less (say after 10 PM). Also from categorical and numerical plots we found that the prices are tending to go up as the time is approaching from morning to evening.

2.What is the best time to buy so that the consumer can save the most by taking the least risk?

> From the categorical plots (bar and box) we came to know that early morning and late night flights are cheaper compared to working hours.

3.Does price increase as we get near to departure date?

> From the categorical plots we found that the flight ticket prices increases as the person get near to departure time. That is last minute flights are very expensive.

4.Is Indigo cheaper than Jet Airways?

From the bar plot we got to know that both Indigo is bit expensive than SpiceJet airways.

5.Are morning flights expensive?

Not all flights are expensive during morning, only few flights departing in the early morning are expensive. Apart from this the flight ticket fares are less compared to other timing flight fares.

4.2 Learning Outcomes of the Study in respect of Data Science

While working on this project I learned many things about the features of flights and about the flight ticket selling web platforms and got the idea that how the machine learning models have helped to predict the price of flight tickets. I found that the project was quite interesting as the dataset contains several types of data. I used several types of plotting to visualize the relation between target and features. This graphical representation helped me to understand which features are important and how these features describe price of tickets. Data cleaning was one of the important and crucial things in this project where I dealt with features having string values, features extraction and selection. Finally got Extra Trees Regressor as best model.

The challenges I faced while working on this project was when I was scrapping the real time data from websites like yaatra, makemytrip, tripodeal and vimaansafar. Finally, our aim was achieved by predicting the flight ticket price and built flight price evaluation model that could help the buyers to understand the future flight ticket prices.

4.3 Limitations of this work and scope for future work

Limitations: The main limitation of this study is the low number of records that have been used. In the dataset our data is not properly distributed in some of the columns many of the values in the columns are having string values which I had taken care. Due to some reasons our models may not make

the right patterns and the performance of the model also reduces. So that issues need to be taken care.

Future work: The greatest shortcoming of this work is the shortage of data. Anyone wishing to expand upon it should seek alternative sources of historical data manually over a period of time. Additionally, a more varied set of flights should be explored, since it is entirely plausible that airlines vary their pricing strategy according to the characteristics of the flight (for example, fares for regional flights out of small airports may behave differently than the major, well flown routes we considered here). Finally, it would be interesting to compare our system's accuracy against that of the commercial systems available today (preferably over a period of time).