FLIP ROBO

Project Report On

"Flight Price Prediction"

Submitted By :

Akanksha Raja Parmar

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 where in 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.

- ✓ https://www.google.com/
- ✓ https://scikit-learn.org/stable/index.html
- ✓ https://github.com
- ✓ https://www.youtube.com/user/krishnaik06
- ✓ https://www.analyticsvidhya.com/

INTRODUCTION

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 a Machine Learning model for flight ticket price prediction and analysis. This model will provide the

approximate selling price for the flight ticket based on different features.

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 travelers 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 –

- Time of purchase patterns (making sure last-minute purchases are expensive).

- 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.

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 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.

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 approach and work in an industry end to end.

ANALYTICAL PROBLEM FRAMING

Mathematical/ Analytical Modeling 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 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

Data Sources and their formats:

We have collected the dataset from the website www.yatra.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 3355 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 3355 rows and 9 columns including target variable “Price”. The particular dataset contains both categorical and numerical data type. The data description is as follows:

Variables	Definition
Airline	The name of airline.
Departure_time	The time when the journey starts from the source.
Time_of_arrival	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.
Meal_availability	Availability of meals in the flight.
Number_of_stops	Total stops between the source and destination.
Price	The price of the flight ticket.

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:

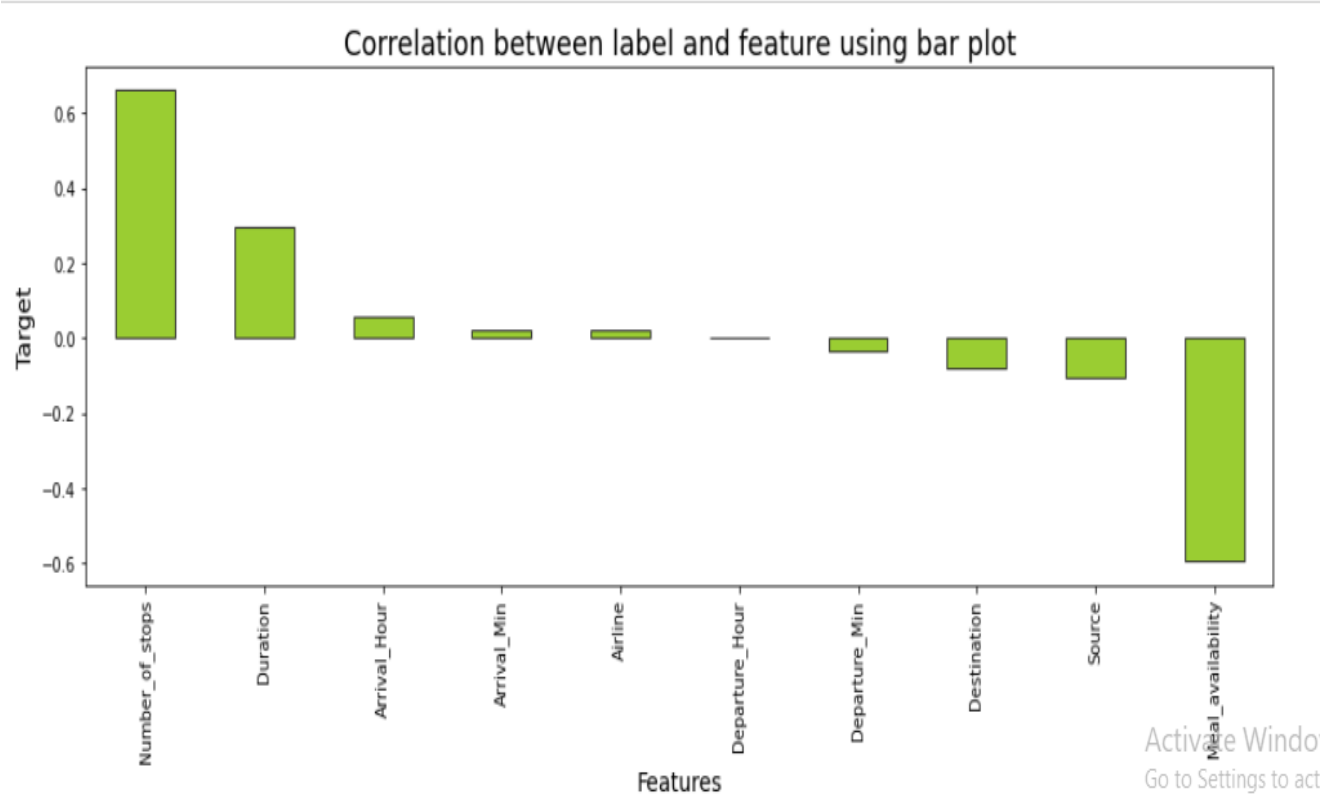
- Importing necessary libraries and loading collected dataset as a dataframe.
- Checked some statistical information like shape, number of unique values present, info, unique(), data types, value count function etc.
- Checked null valued and found no missing values in the dataset.
- Taking care of Timestamp variables by converting data types of “Departure_time” and “Time_of_arrival” 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_Minand, Arrival_Hour, Arrival_Min columns from Departure_time and Time_of_arrival 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 Meal_availability we observed “eCash250” entry which does not belong to meals so I have replaced it as “None” and grouped same categories.

- From the value count function of Number_of_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 Features. With the help of heatmap and correlation bar graph was able to understand the Feature vs Label relativity.
- Separated feature and label data and feature scaling is performed using Standard Scaler method to avoid any kind of data biasness.

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.



Libraries Required:

```
# 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

# 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

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 statistical 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.

MODEL/ S DEVELOPMENT AND EVALUATION

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 Extra Trees 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.

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:

- Decision Tree Regressor
- Random Forest Regressor
- Extra Trees Regressor
- Gradient Boosting Regressor
- Bagging Regressor

Run and evaluate selected models:

I have used 5 regression algorithms after choosing random state amongst 1-1000 number. I have used Random Forest Regressor to find best random state and the code is as below:

```
from sklearn.ensemble import RandomForestRegressor
maxAccu=0
maxRS=0
for i in range(1,200):
    x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=.30, random_state=i)
    mod = RandomForestRegressor()
    mod.fit(x_train, y_train)
    pred = mod.predict(x_test)
    acc=r2_score(y_test, pred)
    if acc>maxAccu:
        maxAccu=acc
        maxRS=i
print("Maximum r2 score is ",maxAccu," on Random_state ",maxRS)
```

```
Maximum r2 score is  0.7939986252640746  on Random_state  56
```

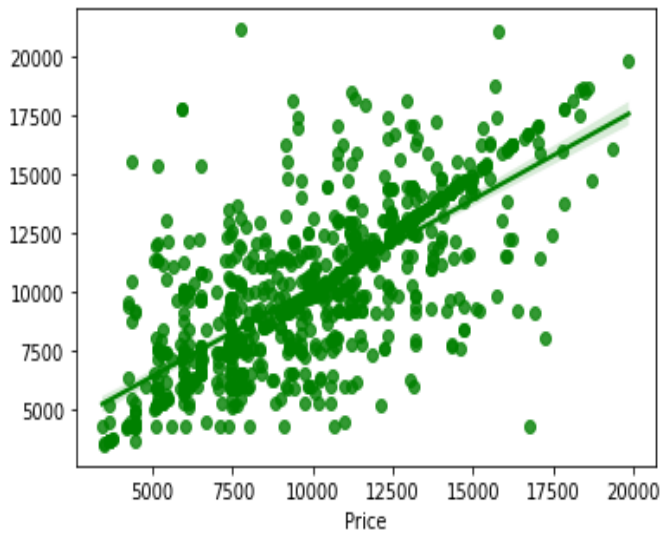
Model Building :

- ***Decision Tree Regressor***

```
# checking R2 score for Decision Tree Regressor
DTR = DecisionTreeRegressor()
DTR.fit(x_train,y_train)

# prediction
predDTR = DTR.predict(x_test)
R2_score = r2_score(y_test,predDTR)*100
print('R2_Score:',R2_score)

# evaluation metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predDTR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predDTR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predDTR)))

# visualizing the predicted values
sns.regplot(y_test,predDTR,color="g")
plt.show()
```

```
R2_Score: 44.73452497550957
Mean Absolute Error: 1437.711022840119
Mean Squared Error: 5939949.040218471
Root Mean Squared Error: 2437.2010668425514
```

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.



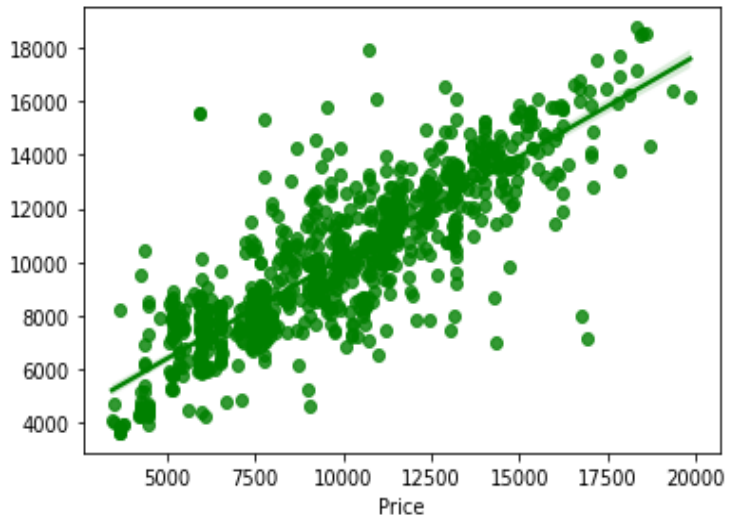
- Created Decision Tree Regressor model and checked for its evaluation metrics. The model is giving R2 score as 44.73%.
- From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that the model has given.

- ***Random Forest Regressor:***

```
# checking R2 score for Random Forest Regressor
RFR = RandomForestRegressor()
RFR.fit(x_train,y_train)

# prediction
predRFR = RFR.predict(x_test)
R2_score = r2_score(y_test,predRFR)*100
print('R2_Score:',R2_score)

# evaluation metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predRFR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predRFR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predRFR)))

# visualizing the predicted values
sns.regplot(y_test,predRFR,color="g")
plt.show()
```

```
R2_Score: 72.49667780575086
Mean Absolute Error: 1146.1626107722138
Mean Squared Error: 2956064.924768207
Root Mean Squared Error: 1719.3210650626622
```

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



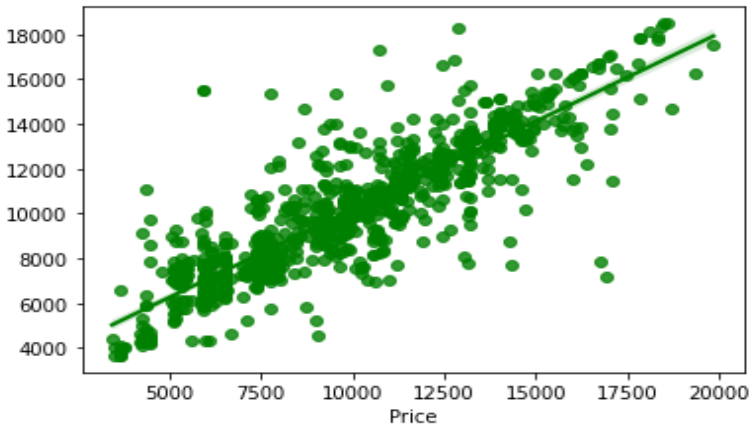
- Created Random Forest Regressor model and checked for it's evaluation metrics. The model is giving R2 score as 72.49%.
- From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

- ***Extra Trees Regressor:***

```
# checking R2 score for Extra Tree Regressor
XT = ExtraTreesRegressor()
XT.fit(x_train,y_train)

# prediction
predXT = XT.predict(x_test)
R2_score = r2_score(y_test,predXT)*100
print('R2_Score:',R2_score)

# evaluation metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predXT))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predXT))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predXT)))

# visualizing the predicted values
sns.regplot(y_test,predXT,color="g")
plt.show()
```

```
R2_Score: 75.2985903207323
Mean Absolute Error: 1019.6317461105594
Mean Squared Error: 2654914.567392924
Root Mean Squared Error: 1629.3908577725983
```

The Extra Trees implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.



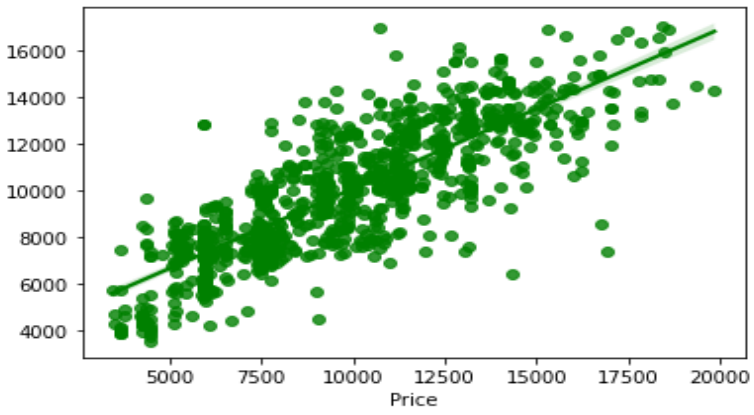
- Created Extra Trees Regressor model and checked for its evaluation metrics . The model is giving R2 score as 75.29%.
- From the graph we can observe how our model is mapping . In the graph we can observe the straight line which is our actual dataset and dots are the predictions that our model has given.

- ***Gradient Boosting Regressor:***

```
# checking R2 score for Gradient Boosting Regressor
GB = GradientBoostingRegressor()
GB.fit(x_train,y_train)

# prediction
predGB = GB.predict(x_test)
R2_score = r2_score(y_test,predGB)*100
print('R2_Score:',R2_score)

# evaluation metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predGB))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predGB))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predGB)))

# visualizing the predicted values
sns.regplot(y_test,predGB,color="g")
plt.show()
```

```
R2_Score: 67.8161302471119
Mean Absolute Error: 1395.0415578268264
Mean Squared Error: 3459131.5132000274
Root Mean Squared Error: 1859.8740584243944
```

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.



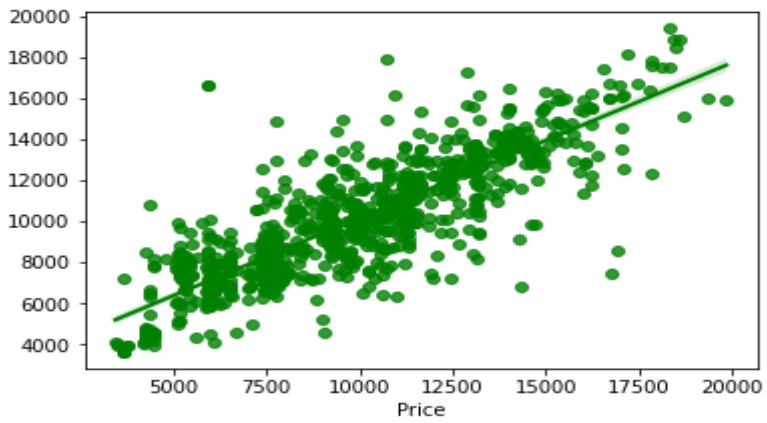
- Created GradientBoosting Regressor model and checked for its evaluation metrics. The model is giving R2 score as 67.81%.
- From the graph we can observe how our model is mapping. In the graph we can observe the straight line which is our actual dataset and the dots are the predictions that our model has given.

- ***Bagging Regressor :***

```
# checking R2 score for Bagging Regressor
BR = BaggingRegressor()
BR.fit(x_train,y_train)

# prediction
predBR = BR.predict(x_test)
R2_score = r2_score(y_test,predBR)*100
print('R2_Score:',R2_score)

# evaluation metrics
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, predBR))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, predBR))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, predBR)))

# visualizing the predicted values
sns.regplot(y_test,predBR,color="g")
plt.show()
```

```
R2_Score: 70.44793984453162
Mean Absolute Error: 1200.2781843760345
Mean Squared Error: 3176263.8659880306
Root Mean Squared Error: 1782.2075821822862
```

A Bagging Regressor is an ensemble meta-estimator that fits base regressors each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction.



- Created Bagging Regressor model and checked for its evaluation metrics. The model is giving R2 score as 70.44%.
- 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 "Extra Trees Regressor" as the best fitting model as it is giving high R2 score and low MAE, MSE and RMSE values. Then I tried to increase our model score by tuning the best model using different types of hyper parameters.

Hyper Parameter Tuning:

```
# let's use the GridSearchCV to find the best parameters in ExtraTrees Regressor
from sklearn.model_selection import GridSearchCV

# Extra Trees Regressor
parameter = {'n_estimators':[10,100,1000],
             'criterion':['squared_error','mse','absolute_error','mae'],
             'min_samples_split': [1,2,3,4],
             'max_features':['auto','sqrt','log2'],
             'n_jobs':[-2,-1,1,2]}
```

I have used 5 Extra Trees Regressor parameters to be saved under the variable "parameter" that will be used in Grid Search CV for finding the best output.

```
GCV=GridSearchCV(ExtraTreesRegressor(),parameter,cv=5)
```

Assigning a variable to the GridSearchCV function after entering all the necessary inputs.

```
# running Grid Search CV
GCV.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=ExtraTreesRegressor(),
             param_grid={'criterion': ['squared_error', 'mse', 'absolute_error',
                                       'mae'],
                         'max_features': ['auto', 'sqrt', 'log2'],
                         'min_samples_split': [1, 2, 3, 4],
                         'n_estimators': [10, 100, 1000],
                         'n_jobs': [-2, -1, 1, 2]})
```


Now we use our training data set to make the GridSearchCV aware of all the hyper parameters that needs to be applied on our best model.

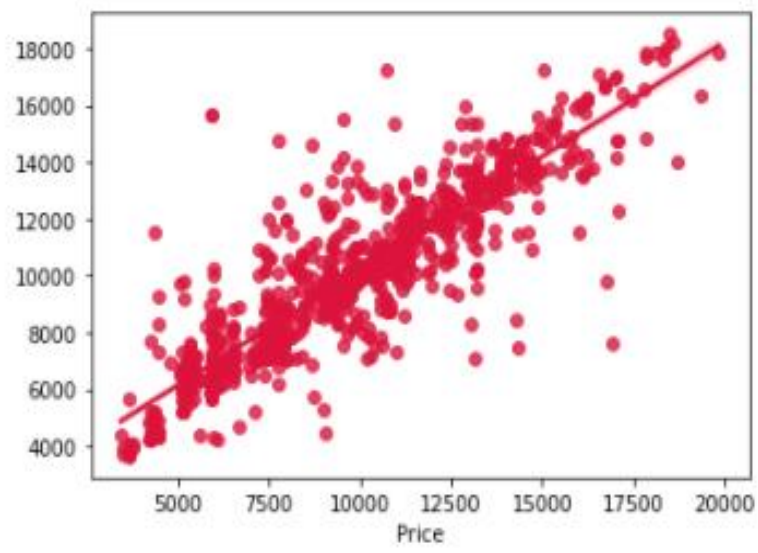
```
# finding best parameters
GCV.best_params_
```

```
{'criterion': 'absolute_error',
 'max_features': 'auto',
 'min_samples_split': 2,
 'n_estimators': 1000,
 'n_jobs': -2}
```

This gives us the list of best parameters which will be used further in our final model creation.

```
# creating final model
Flight_price_model = ExtraTreesRegressor(criterion='mae',max_features='auto',min_samples_split=3,n_estimators=1000,n_jobs=-1)

# prediction
Flight_price_model.fit(x_train, y_train)
pred = Flight_price_model.predict(x_test)
print('R2_Score:',r2_score(y_test,pred)*100)

#  metrics evaluation
print('Mean Absolute Error:',metrics.mean_absolute_error(y_test, pred))
print('Mean Squared Error:',metrics.mean_squared_error(y_test, pred))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test, pred)))

# visualizing the predicted values
sns.regplot(y_test,pred,color="crimson")
plt.show()
```

```
R2_Score: 78.25541368261959
Mean Absolute Error: 925.7143704071499
Mean Squared Error: 2337114.347947512
Root Mean Squared Error: 1528.7623582321457
```

- We have successfully incorporated the hyper parameter tuning using best parameters of Extra Trees Regressor and the R2 score of the model has been increased after hyperparameter tuning and received the R2 score as 78.25 % which is very good.
- From the graph we can observe how our final model is mapping. In the graph we can observe the best fit line which is our actual dataset and the dots are the predictions that our best final model has given.

Saving the final model

```
# saving the model using joblib library
import joblib
joblib.dump(Flight_price_model,"Flight_Ticket_Price_Prediction.pkl")
```

```
['Flight_Ticket_Price_Prediction.pkl']
```

I am using the joblib option to save the final regression model in the form of .pkl .

Loading the saved model and prediction Flight Ticket Price

```
# loading the saved model
Model=joblib.load("Flight_Ticket_Price_Prediction.pkl")

# prediction
prediction = Model.predict(x_test)
prediction
```

```
array([10245.5445,  7152.346 ,  9148.6705, ...,  3638.9505,  6545.44  ,
       11236.295 ])
```

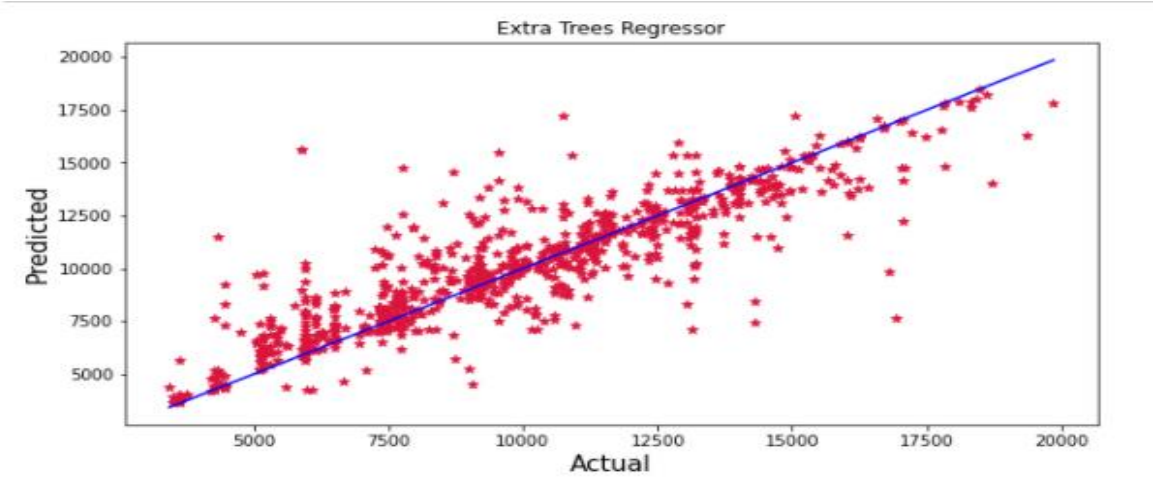
These are the predicted price of the flight tickets.

```
Predicted_Flight_Ticket_Price = pd.DataFrame([Model.predict(x_test)[:],y_test[:]],index=["Predicted","Actual"])
Predicted_Flight_Ticket_Price
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	
Predicted	10245.5445	7152.346	9148.6705	11567.3035	11492.2705	11541.952	17067.6095	9101.2125	17828.0	14141.438	3729.023	7140.9235	13108.967	438
Actual	8160.0000	10264.000	9141.0000	9538.0000	11520.0000	11319.000	16569.0000	9252.0000	17828.0	14388.000	3636.000	5499.0000	14802.000	447

2 rows × 1007 columns

Using regression model, we have got the predicted price of the flight tickets. From the above output we can observe that predicted values are almost near to the actual values.



This graph shows how our model is mapping. The plot gives the linear relation between predicted and actual price of the flight tickets. The blue line is the best fitting line which gives the actual values/data and red dots gives the predicted values/data.

Visualizations :

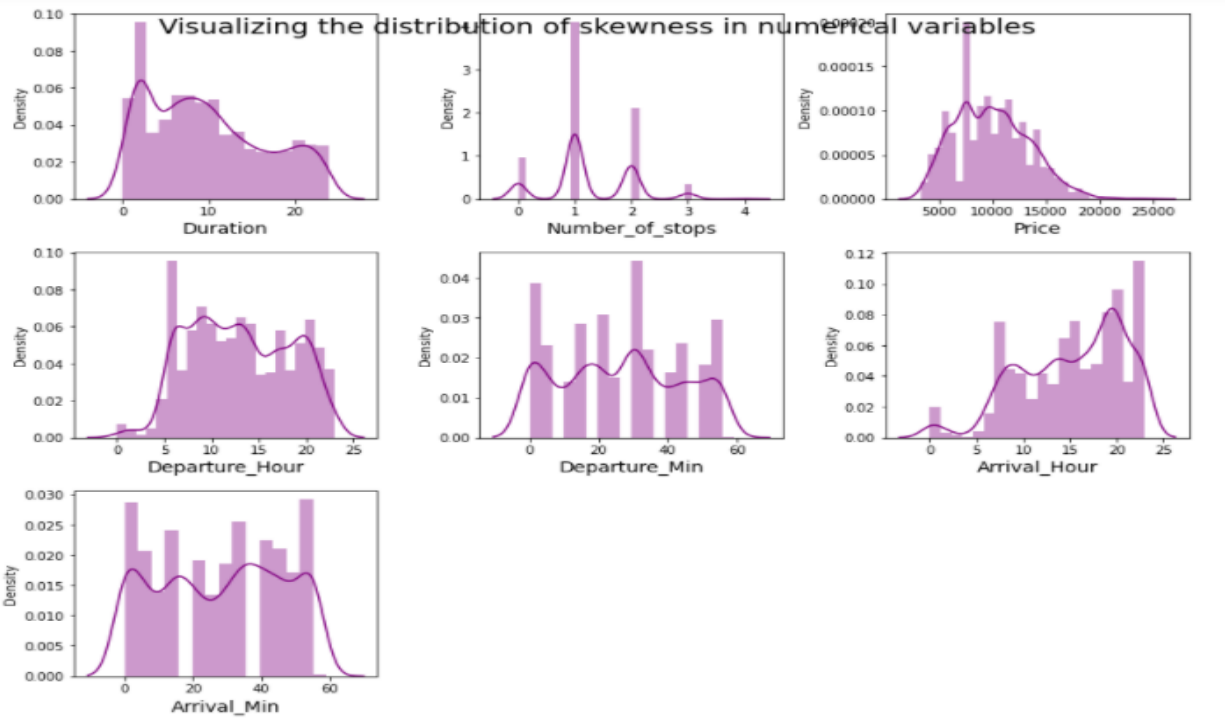
I used pandas profiling to get the over viewed visualization on the pre-processed data. Pandas is an open-source Python module with which we can do an exploratory data analysis to get detailed description of the features and it helps in visualizing and understanding the distribution of each variable. 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.

Univariate Analysis for Numerical Variables:



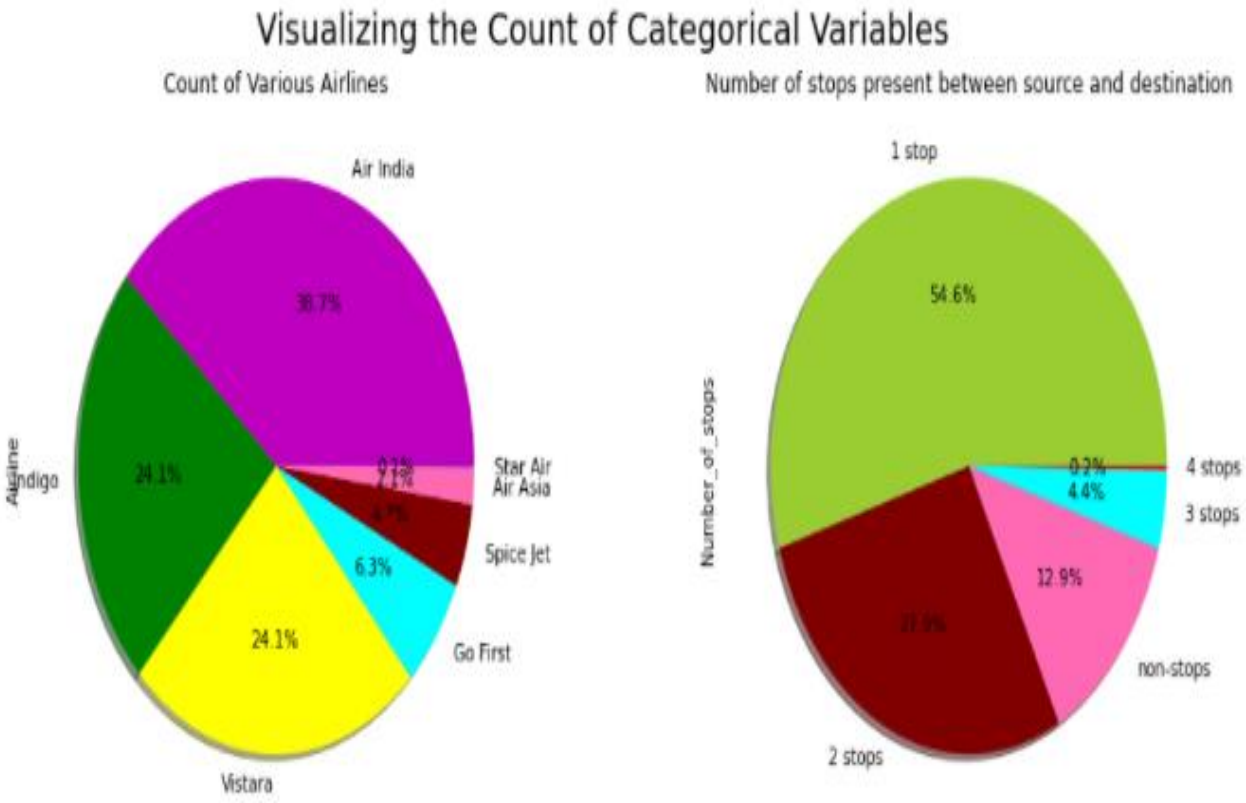
Observation:

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.
- ✓ The columns like “Duration”, “Number_of_stops” and “Price” are skewed to right as the mean value in these columns are much greater than the median(50%).
- ✓ Also, the data in the column Arrival_Hour skewed to left since the mean values is less than the median.

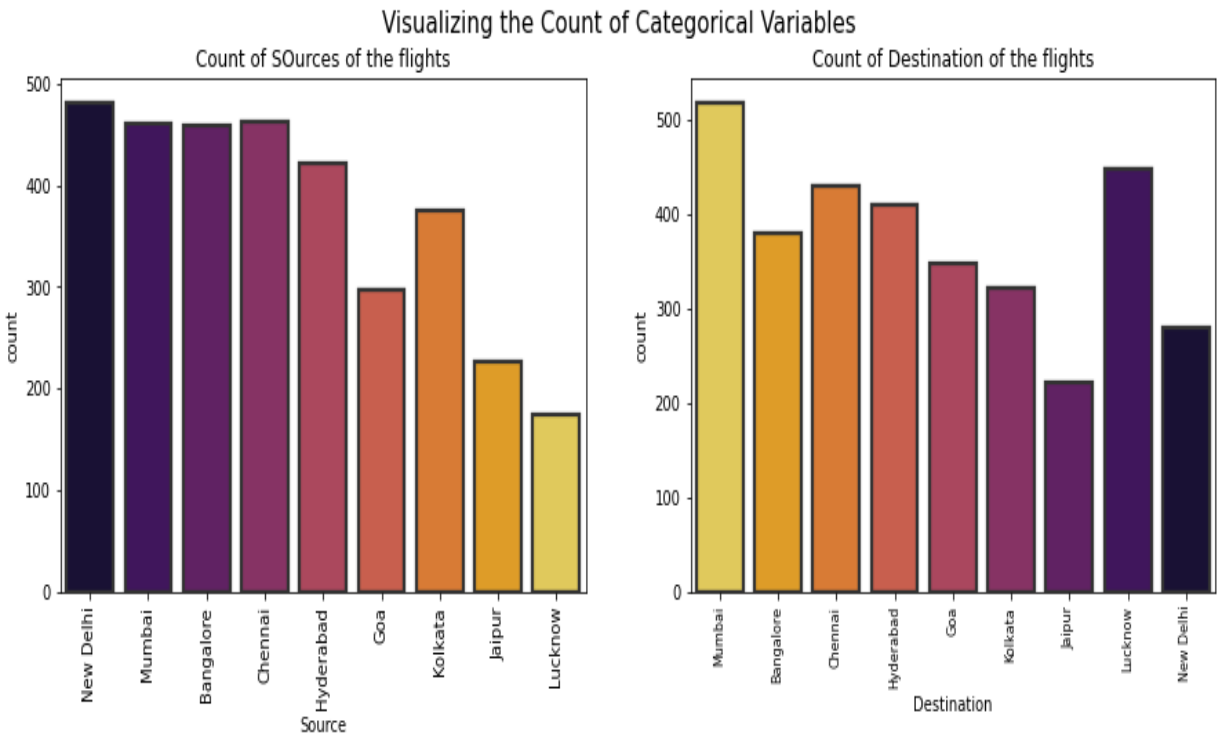
- ✓ Since there is presence of skewness in the data, we need to remove skewness in the numerical columns to overcome with any kind of data biasness.

Univariate Analysis for Categorical Variables:



Observations:

- ***Airline*** : From the pie plot we can infer that there are more number of flights of "Air India", "Indigo" and "Vistara" compared to others. Also, the count of "Star Air" flights are very less.
- ***Number_of_stops*** : From the above pie plot we can infer that 54.6% of the flights have only 1 stop during the journey and some of the flights (27.9%) have 2 stops where only few flights have 3, 4 stops.

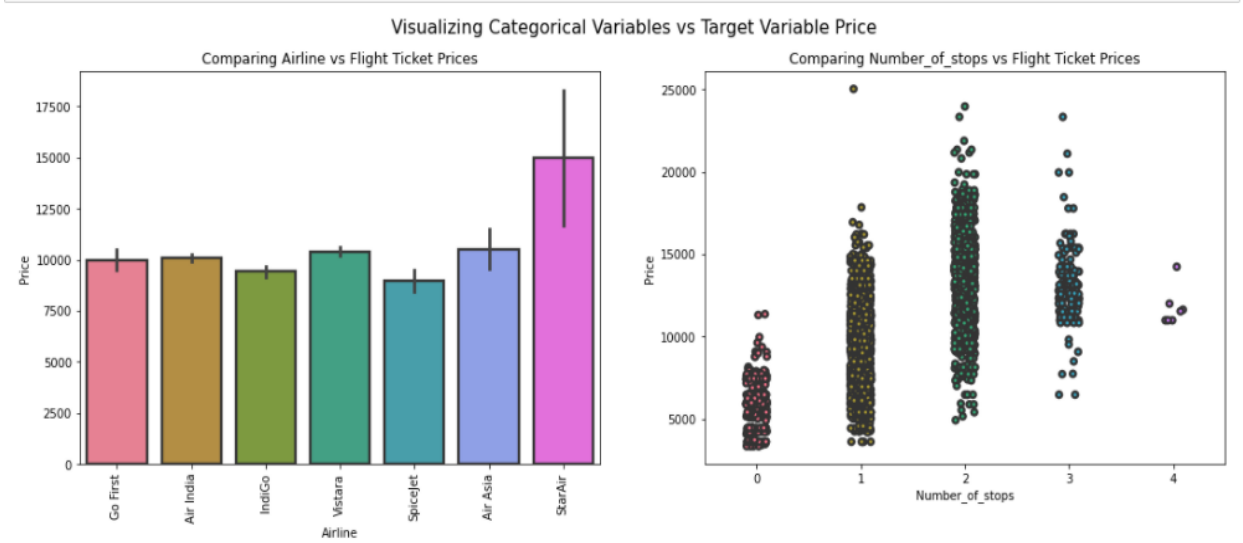


Observations:

- ***Source***: From the count plot we can observe more number of flights are from New Delhi, Chennai, Mumbai, Bangalore and Hyderabad. Only few flights are from Lucknow.
- ***Destination***: More number of flights are heading towards Mumbai, Lucknow and Chennai. Only few flights are travelling to Jaipur.

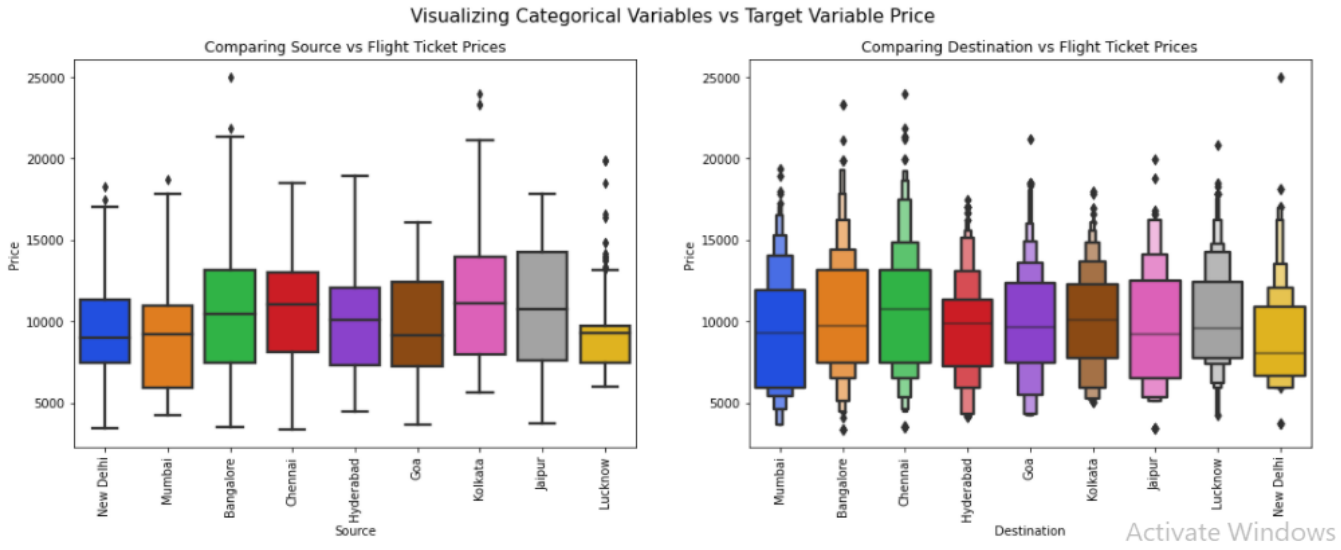
Bivariate Analysis: Bivariate analysis is one of the statistical analyses where two variables are observed, Bi means two variables. One variable here is dependent while the other is independent. These variables are usually denoted by X and Y . We can also analyse the data using both idependent variables.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

Visualizing Categorical Variables vs Target Variable:



Observations:

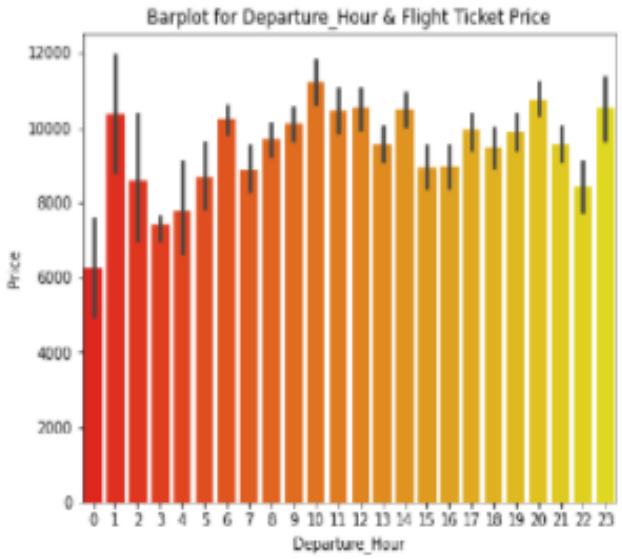
- ***Airline vs Price***: From the bar plot we can notice "Star Air" and "Vistara" airlines have highest ticket prices compared to other airlines.
- ***Number_of_stops vs Price***: From the strip plot we can notice the flights which have 1 and 2 stops between source and destination have highest ticket 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.

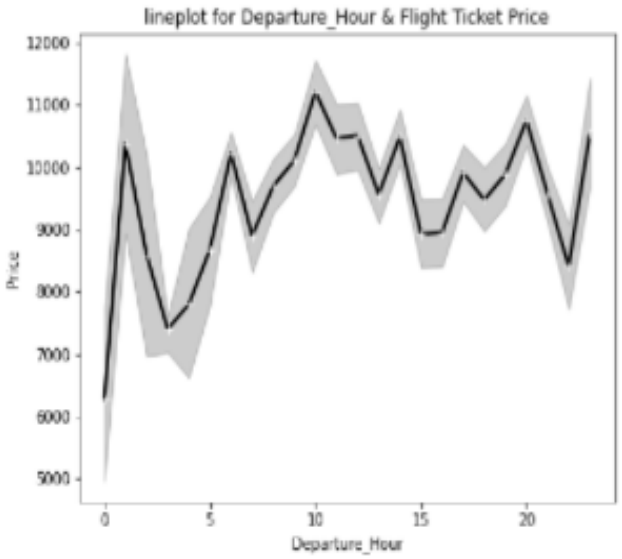


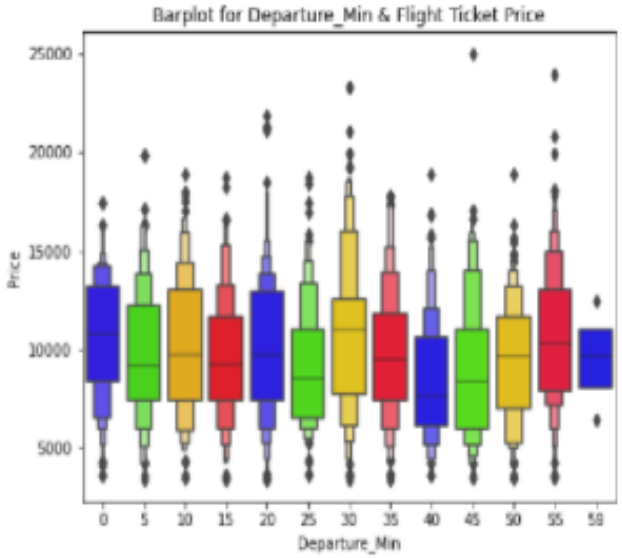
Observations:

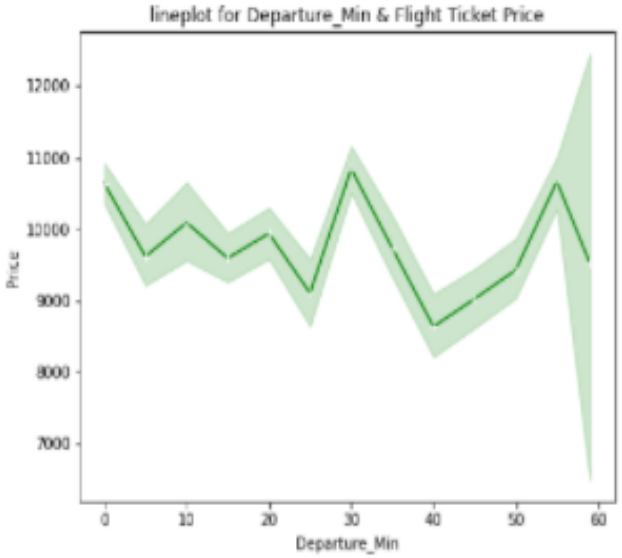
- ***Source vs Price***: From the box plot we can observe the flights from Jaipur are having somewhat higher prices compared to other sources.
- ***Destination vs Price***: From the boxen plot we can notice that the flights travelling to Banglore have higher flight ticket prices.

Visualizing Numerical Variables vs Target Variable Price:



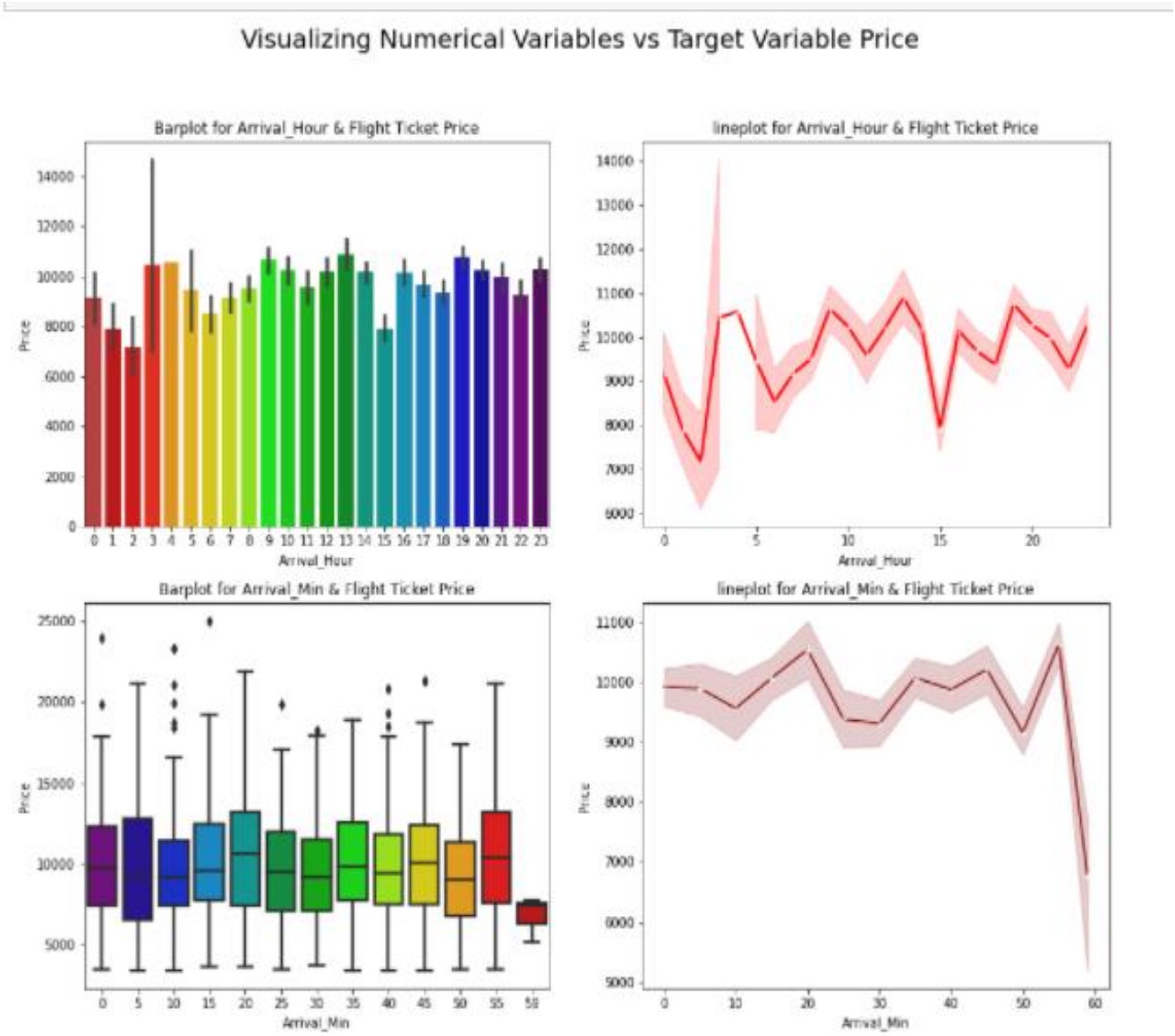






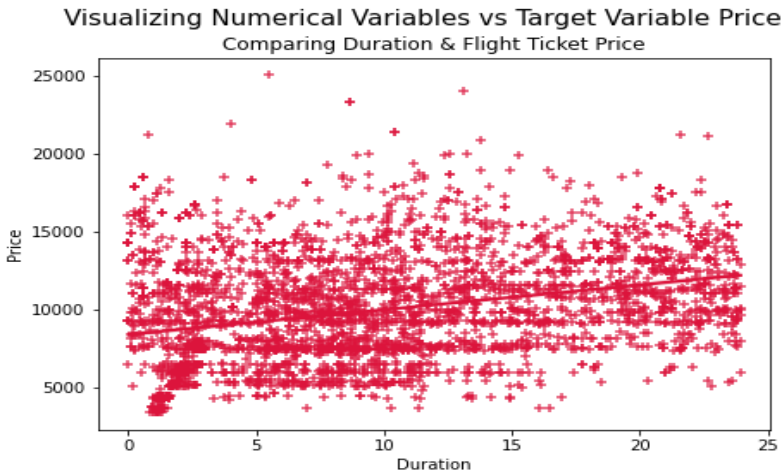
Observations:

- ***Departure_Hour vs Price***: From the bar plot and line plot we can see that there are some flights departing in the early morning 3 AM 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.

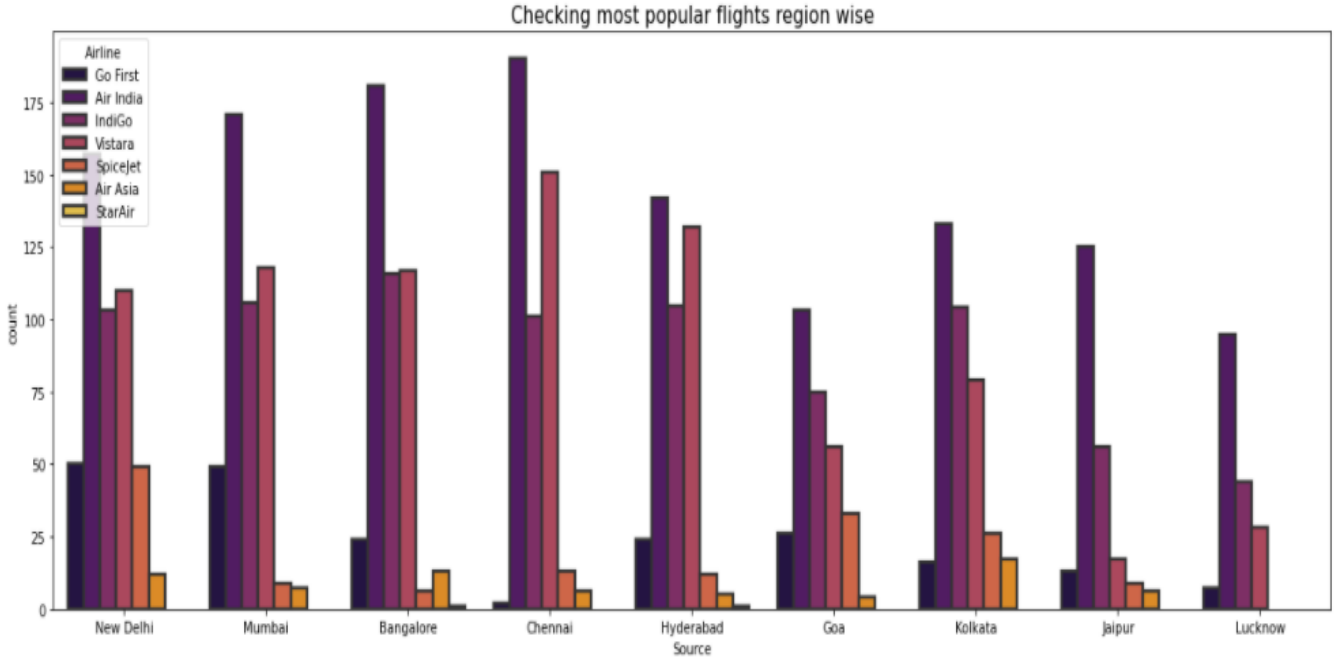


Observation:

- ***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 aftermoon and evening have somewhat higher price. So, we can conclude 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.

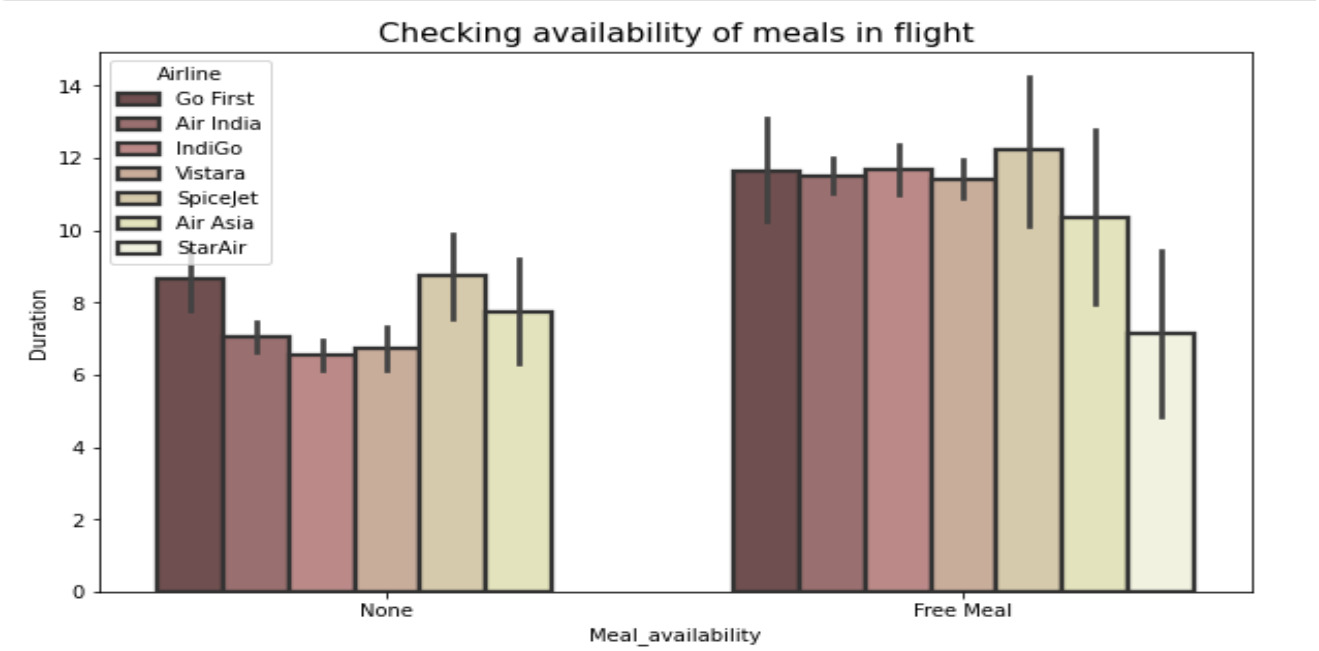


Duration vs Price: From the reg plot we can observe some positive linear relation between Duration and Price. Flight having 1-12 hours of duration, they have ticket price of around 15000.



Observations:

- ***Source vs Airline***: The plot showing the region wise count of airlines which tells us that Jaipur source is not having much flights and it has Air India flights in higher count compared to other sources. Other sources have Air India, IndiGo and Vistara flights with higher count.



Observations:

- All the airlines provides free meals during the journey having the duration below 11 hours.

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 pair wise 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 Extra Trees Regressor as the best model which gives 75.29% R2 score. After tuning the best model the R2 score of Extra Trees Regressor has been increased to 78.25% and also got low evaluation metrics. Finally, I saved my final model and got the good predictions results for price of flight tickets.

CONCLUSION

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. Performed hyper parameter tuning on the best model and the best model's R2 score in creased and was gibing R2 score as 78.25% . 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?***

- 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.

2. ***Do they tend to go up or down over time?***

- Some flights are departing in the early morning 3 AM 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 . 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.

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

- From the categorical plots we came to know that early morning and late night fights are cheaper compared to working hours.

4. ***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.

5. ***Is Indigo cheaper than Jet Airways?***

- From the bar plot we got to know that both Indigo and Spicejet airways almost having same ticket fares.

6. ***Are morning flights expensive?***

- Not all flights are expensive during morning. Only few flights departing in the early morning 3 AM are expensive. Apart from this the flight ticket fares are less compared to other timing flight fares.

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 yatra website, it took so much time to gather data. 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.

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).

****