FLIP ROBO

Project Report On

Flight Price Prediction

Submitted by:

VISHWAS PAI

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. 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 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://www.youtube.com/
3) https://github.com/
4) https://www.kaggle.com/
5) https://medium.com/
6) https://towardsdatascience.com/
7) https://www.analyticsvidhya.com/

TABLE OF CONTENTS

1. INTRODUCTION

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 during a particular time. 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 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 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 Machine Learning 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 yatra 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, decision trees and other Regressors 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. ANALYTICAL PROBLEM FRAMING

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 yatra website where I found more features of flights compared to other websites and I fetched 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 analysing 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 model 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 yatra website 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 5303 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 5303 rows and 9 columns including target variable "Price". The particular dataset contains both categorical and numerical data type. The data description is as follows:

1	Airline	The Name of airline
2	Departure_time	The time when the journey starts from the source
3	Time_of_arrival	Time of arrival at the destination
4	Duration	Total duration taken by the flight to reach the destination from the source
5	Source	The source from which the service begins
6	Destination	The destination where the service ends
7	Meal_availability	Availability of meals in the flight
8	Number_of_stops	Total stops between the source and destination
9	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:

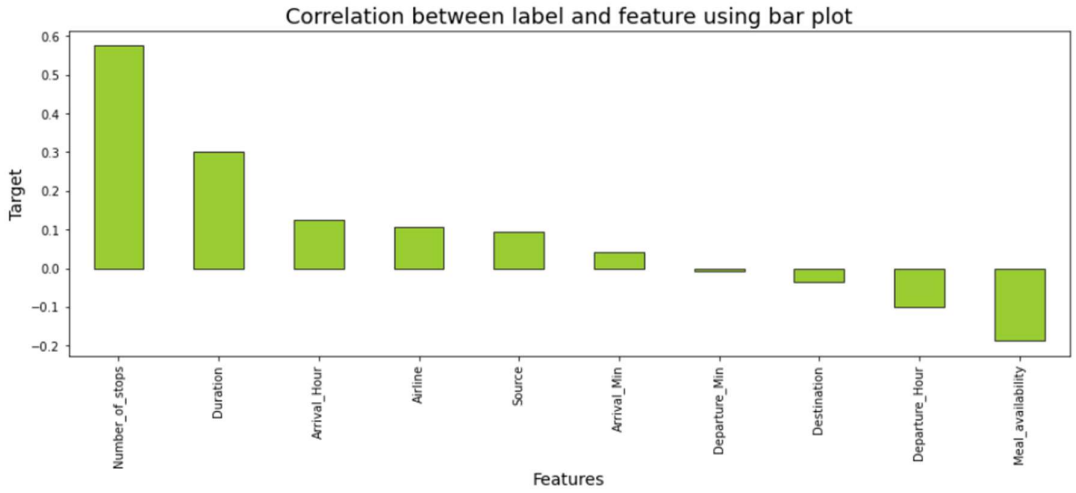
- Importing necessary libraries and loading collected dataset as a dataframe.

- 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 “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_Min and 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 "eCash 250" 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 Feature vs Label relativity.
- Separate feature and target 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.



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 i5 or above
- 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:

```
# 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 itself.
- **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 Extra Trees Regressor as best suitable algorithm to create our final model as it is giving highR2 score and low 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 Regressor
2. Lasso Regressor
3. Ridge Regressor
4. Elastic Net Regressor
5. Support Vector Regressor
6. Decision Tree Regressor
7. Random Forest Regressor
8. K Neighbors Regressor
9. SGD Regressor
10. Gradient Boosting Regressor
11. Ada Boost Regressor
12. Extra Trees Regressor
13. Extreme Gradient Boosting (XGB) Regressor

3.3 Run and evaluate selected models

I have used 13 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  0.7429758080082968  on Random_state  126
```

With the help of random state selection process we have found our random state to be 126 amongst 1-1000 with best accuracy as 74.2975% for test size=0.3 using Random Forest Regressor.

```
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=.20, 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  0.7589279830507123  on Random_state  53
```

With the help of random state selection process we have found our random state to be 53 amongst 1-1000 with best accuracy as 75.8927% for test size=0.2 using Random Forest Regressor.

Model Building:

I have imported all the required libraries of the Machine Learning algorithms and created a function which can run all the regressors and provide the results in a systematic way for easier reference.

```
# importing all the required libraries
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso,Ridge,ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import SGDRegressor
from sklearn.ensemble import AdaBoostRegressor
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.ensemble import GradientBoostingRegressor
import xgboost as xgb
```

```
# creating a function to run all the regressors

def regressor(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=53)

    # Training the model
    model.fit(x_train, y_train)

    # Predicting y_test
    pred = model.predict(x_test)

    # Root Mean Square Error (RMSE)
    rmse = mean_squared_error(y_test, pred, squared=False)
    print("Root Mean Square Error is:", rmse)

    # R2 score
    r2 = r2_score(y_test, pred, multioutput='variance_weighted')*100
    print("R2 Score is:", r2)

    # Cross Validation Score
    cv_score = (cross_val_score(model, x, y, cv=5).mean())*100
    print("Cross Validation Score is:", cv_score)

    # Result of r2 score - cv score
    result = r2 - cv_score
    print("R2 Score - Cross Validation Score is", result)
```

After creating the function, we will run all the models individually and get the Root Mean Square Error (RMSE), R2 score, Cross Validation Score and Result of r2 score - cv score as shown below after which we will choose the best regressor Algorithm which has the highest R2 score and Cross Validation score and least RMSE value.

The 13 ML Algorithms are as shown below:

Linear Regression

```
model=LinearRegression()
regressor(model, x, y)
```

```
Root Mean Square Error is: 3102.8186536791527
R2 Score is: 42.603234346105936
Cross Validation Score is: 33.77319916855067
R2 Score - Cross Validation Score is 8.830035177555267
```

L1 -- Lasso Regression

```
model=Lasso(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 3102.818672494132
R2 Score is: 42.603233650016925
Cross Validation Score is: 33.773211683967105
R2 Score - Cross Validation Score is 8.83002196604982
```

L2 -- Ridge Regression

```
model=Ridge(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 3102.818626452565
R2 Score is: 42.60323535339528
Cross Validation Score is: 33.77319922834115
R2 Score - Cross Validation Score is 8.830036125054129
```

Elastic Net

```
model=ElasticNet(alpha=0.001)
regressor(model, x, y)
```

```
Root Mean Square Error is: 3102.7613800512713
R2 Score is: 42.605353252187726
Cross Validation Score is: 33.77331900837605
R2 Score - Cross Validation Score is 8.832034243811677
```

Support Vector Regression

```
model=SVR(kernel='rbf')
regressor(model, x, y)
```

```
Root Mean Square Error is: 4063.649859839265
R2 Score is: 1.5519329273662064
Cross Validation Score is: -5.9147846244282976
R2 Score - Cross Validation Score is 7.466717551794504
```

```
model=SVR(kernel='poly')
regressor(model, x, y)
```

```
Root Mean Square Error is: 4012.093509020631
R2 Score is: 4.034147452280323
Cross Validation Score is: -3.3886265089246232
R2 Score - Cross Validation Score is 7.422773961204946
```

```
model=SVR(kernel='linear')
regressor(model, x, y)
```

```
Root Mean Square Error is: 3284.6518314452705
R2 Score is: 35.67892171494081
Cross Validation Score is: 27.57679708465823
R2 Score - Cross Validation Score is 8.102124630282582
```

Decision Tree Regressor

```
model=DecisionTreeRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 2797.249382463219
R2 Score is: 53.35157273966329
Cross Validation Score is: -15.517361218161627
R2 Score - Cross Validation Score is 68.86893395782492
```

Ada Boost Regressor

```
model=AdaBoostRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 3777.64229864085
R2 Score is: 14.922190787491852
Cross Validation Score is: 12.405022993316894
R2 Score - Cross Validation Score is 2.5171677941749575
```

Extra Trees Regressor

```
model=ExtraTreesRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 1905.6529465530937
R2 Score is: 78.34978296442169
Cross Validation Score is: 40.144868435573514
R2 Score - Cross Validation Score is 38.20491452884818
```

Extreme Gradient Boosting Regressor (XGB)

```
from xgboost import XGBRegressor as xgb
model=xgb()
regressor(model, x, y)
```

```
Root Mean Square Error is: 2071.6548978729247
R2 Score is: 74.41358409017838
Cross Validation Score is: 34.17089069885046
R2 Score - Cross Validation Score is 40.24269339132792
```

Random Forest Regressor

```
model=RandomForestRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 1975.905508550808
R2 Score is: 76.72407351266581
Cross Validation Score is: 40.43148795397926
R2 Score - Cross Validation Score is 36.292585558686554
```

K Neighbors Regressor

```
model=KNeighborsRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 2761.4439108347187
R2 Score is: 54.53815197360785
Cross Validation Score is: 23.548905410763474
R2 Score - Cross Validation Score is 30.98924656284438
```

SGD Regressor

```
model=SGDRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 3104.9887472462897
R2 Score is: 42.522920333057655
Cross Validation Score is: 33.7810893175838
R2 Score - Cross Validation Score is 8.741831015473856
```

Gradient Boosting Regressor

```
model=GradientBoostingRegressor()
regressor(model, x, y)
```

```
Root Mean Square Error is: 2478.8654296531226
R2 Score is: 63.36632108788556
Cross Validation Score is: 44.73035371841555
R2 Score - Cross Validation Score is 18.63596736947001
```

Model Selection:

From the above created models, Extra Trees Regressor algorithm has high R2 score and less RMSE value. So, we can conclude that "Extra Trees Regressor" as the best fitting model. Let's try 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 paarameters in XGBRegressor
from sklearn.model_selection import GridSearchCV

# Extra Trees Regressor
parameters = {'n_estimators':[10,100,200],
              'criterion':['squared_error','mse','absolute_error','mae'],
              'min_samples_split': [1,2,3],
              'max_features':['auto','sqrt','log2'],
              'n_jobs':[-2,-1,1]}
```

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

```
# Running GridSearchCV
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],
                         'n_estimators': [10, 100, 200],
                         'n_jobs': [-2, -1, 1]})
```

```
GCV.best_params_          # printing best parameters found by GridSearchCV
```

```
{'criterion': 'mae',
 'max_features': 'auto',
 'min_samples_split': 2,
 'n_estimators': 100,
 'n_jobs': 1}
```

We got the best parameters using Gridsearch CV

```
final_model = ExtraTreesRegressor(criterion = 'mae', max_features = 'auto', min_samples_split = 2, n_estimators = 100, n_jobs = 1
```

```
final_fit = final_model.fit(x_train,y_train)   # final fit
```

```
final_pred = final_model.predict(x_test)   # predicting with best parameters
```

```
best_r2=r2_score(y_test,final_pred,multioutput='variance_weighted')*100   # checking final r2_score
print("R2 score for the Best Model is:", best_r2)
```

```
R2 score for the Best Model is: 78.33566428674997
```

```
final_cv_score = (cross_val_score(final_model, x, y, cv=5).mean())*100
print("Cross Validation Score is:", final_cv_score)
```

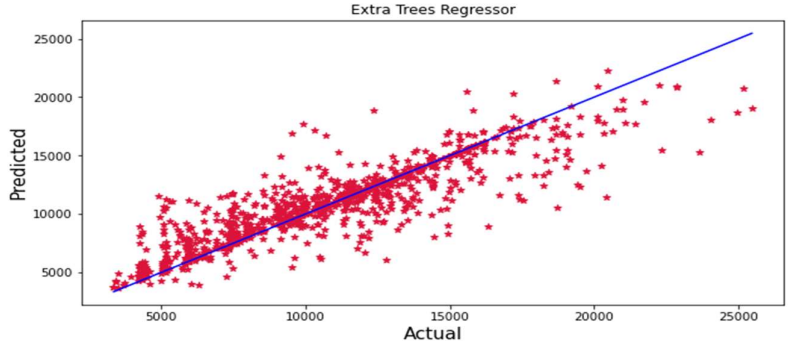
```
Cross Validation Score is: 39.749733127238315
```

```
final_rmse = mean_squared_error(y_test, final_pred, squared=False)
print("Root Mean Square Error is:", final_rmse)
```

```
Root Mean Square Error is: 1906.2742085634945
```

We used Hyper Parameter Tuning on the final model to obtained the best r2_score and CV score.

```
# Visualizing actual and predicted values
plt.figure(figsize=(10,5))
plt.scatter(y_test, final_pred, c='crimson',marker="*")
p1 = max(max(final_pred), max(y_test))
p2 = min(min(final_pred), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('Actual', fontsize=15)
plt.ylabel('Predicted', fontsize=15)
plt.title("Extra Trees Regressor")
plt.show()
```



We have successfully incorporated the hyper parameter tuning using best parameters of Extra Trees Regressor and obtained the R2 score as 78.335% which is very good.

Saving the Final model

```
# Saving the model using joblib library
import joblib
joblib.dump(final_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 predicting Flight Ticket Price

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

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

```
array([14016.75, 14377.  ,  7558.11, ...,  9599.2 , 10339.2 , 10442.01])
```

These are the predicted price of the flight tickets.

Creating DataFrame for the predicted values

```
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	...	1051	1052	1053	1054	1055	
Predicted	14016.75	14377.0	7558.11	8545.39	16745.955	8264.21	7418.1	7140.91	11061.13	8498.84	...	16484.0	10010.12	11401.52	17014.38	12644.33	818
Actual	19028.00	14377.0	4262.00	7907.00	10721.000	8055.00	7133.0	6489.00	12486.00	8577.00	...	17304.0	6932.00	14839.00	16140.00	12616.00	798

2 rows × 1061 columns

The graph shows how our final 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.

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 Root Mean Squared Error (RMSE) , R2 Score and Cross Validation for my model evaluation:

- **Cross Validation**:
  Cross-validation helps to find out the over fitting and under fitting of the model. In the cross validation the model is made to run on different subsets of the dataset which will get multiple measures of the model. If

we take 5 folds, the data will be divided into 5 pieces where each part being 20% of full dataset. While running the Cross-validation the 1st part (20%) of the 5 parts will be kept out as a holdout set for validation and everything else is used for training data. This way we will get the first estimate of the model quality of the dataset.

In the similar way further iterations are made for the second 20% of the dataset is held as a holdout set and remaining 4 parts are used for training data during process. This way we will get the second estimate of the model quality of the dataset. These steps are repeated during the cross-validation process to get the remaining estimate of the model quality and their mean are taken for the final CV score.

- **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.
- **R2 Score**:
  It is a statistical measure that represents the goodness of fit of a regression model. The ideal value for R2 Score is 1. The closer the value of R2 Score to 1, the better is the model fitted.
- **Hyperparameter Tuning**:
  There is a list of different machine learning models. They all are different in some way or the other, but what makes them different is nothing but input parameters for the model. These input parameters are named as Hyperparameters. These hyperparameters will define the architecture of the model, and the best part about these is that you get a choice to select these for your model. You must select from a specific list of hyperparameters for a given model as it varies from model to model.

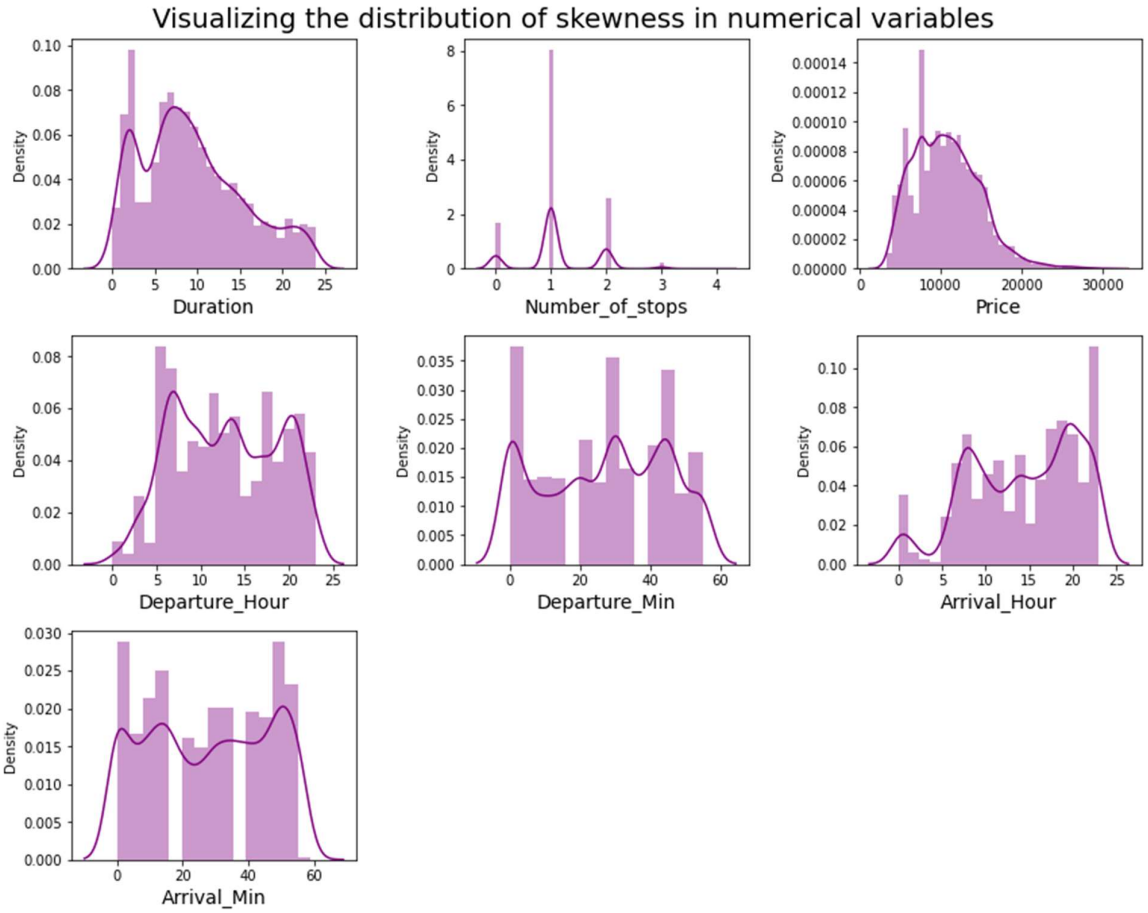
  We are not aware of optimal values for hyperparameters which would generate the best model output. So, what we tell the model is to explore and select the optimal model architecture automatically. This selection procedure for hyperparameter is known as Hyperparameter Tuning. We can do tuning by using GridSearchCV.

3.5 Visualizations

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.

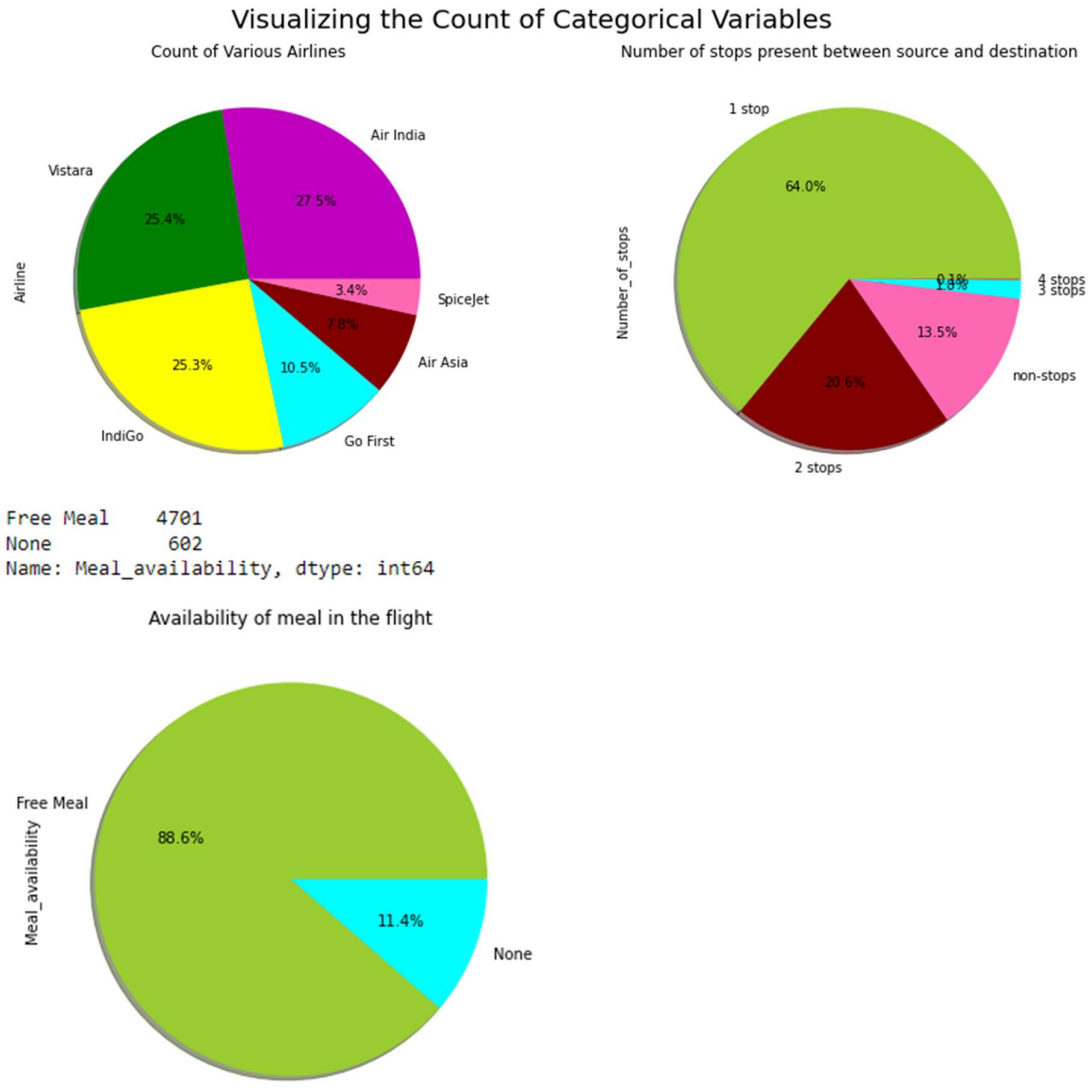
Univariate Analysis for Numerical Variables:

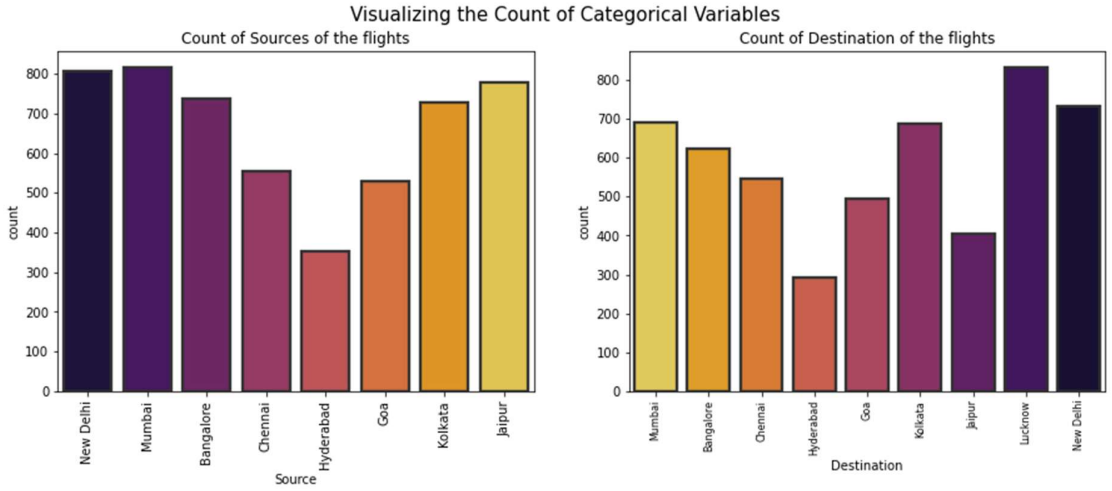


Above plot shows how the data has been distributed in each of the columns.

- 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:



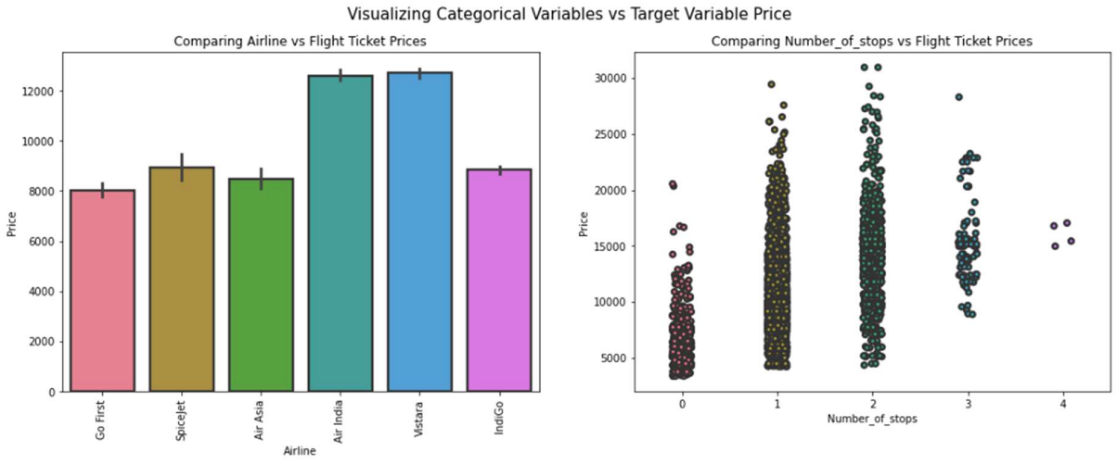


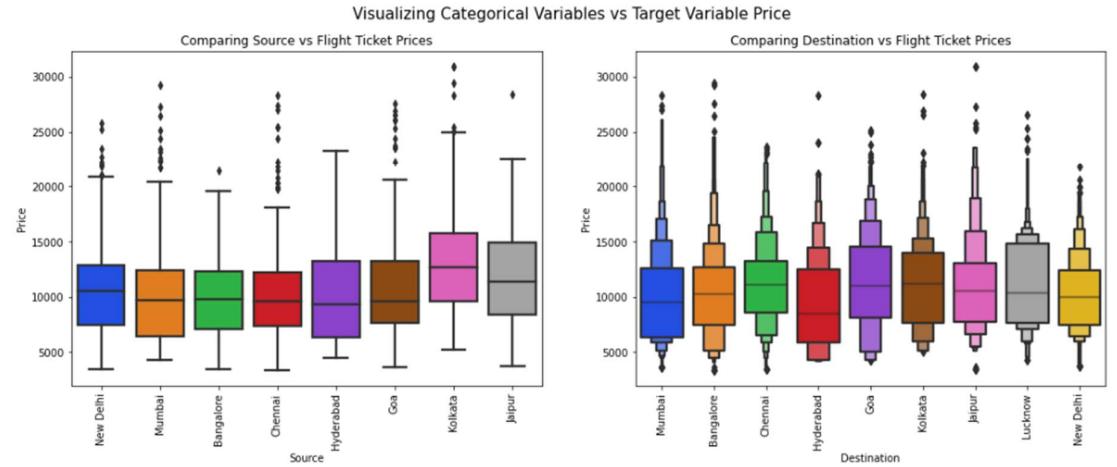
- **Airline**: From the pie plot we can infer that there are a greater number of flights of "Air India", "Vistara" and "Indigo" compared to others. Also, the count of Spicejet flights is very less.
- **Number_of_stops**: From the above pie plot we can infer that 64% of the flights have only 1 stop during the journey and some of the flights (20.6%) have 2 stops where only few flights have 3 and 4 stops.
- **Meal_availability**: Most of the flights providing free meals and only few flights are not providing any meals.

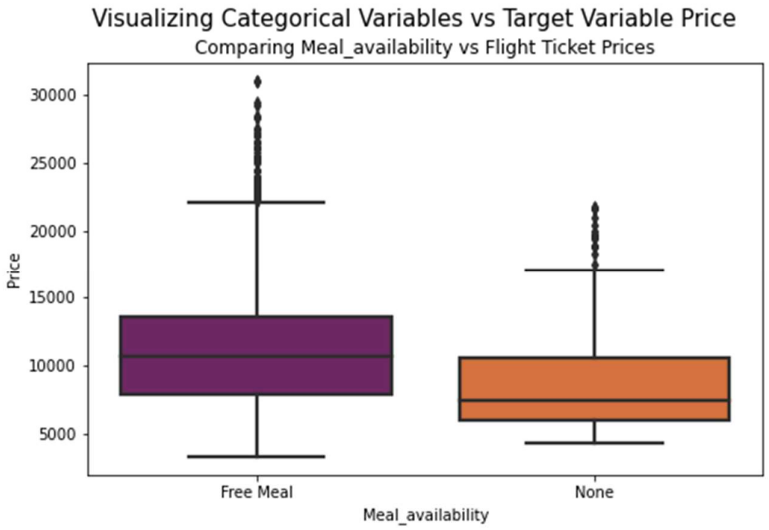
- **Source**: From the count plot we can observe a greater number of flights are from Mumbai, New Delhi, Jaipur, Kolkata and Bangalore. Only few flights are from Hyderabad.
- **Destination**: More number of flights are heading towards Lucknow, New Delhi and Kolkata. Only few flights are travelling to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

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





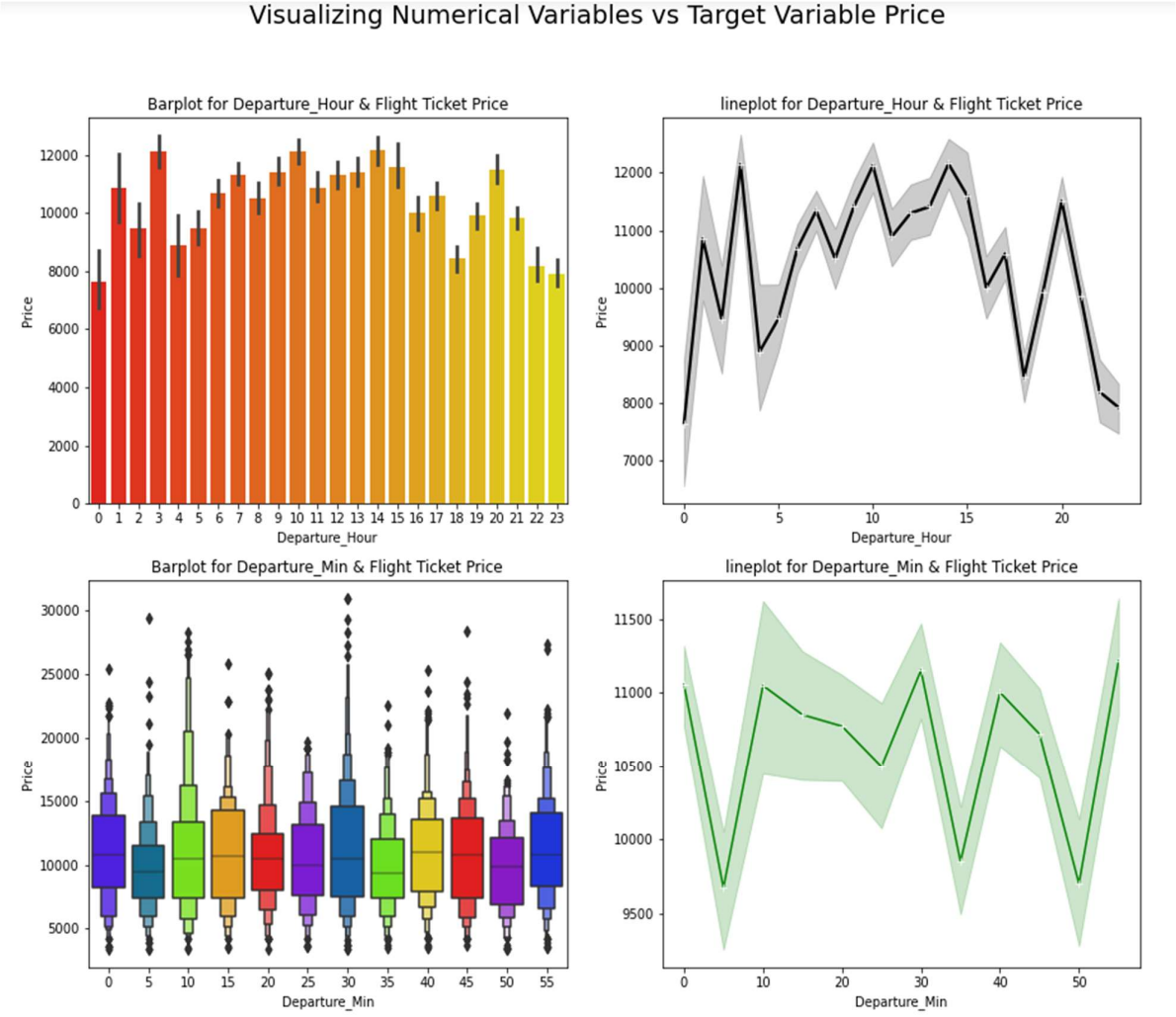


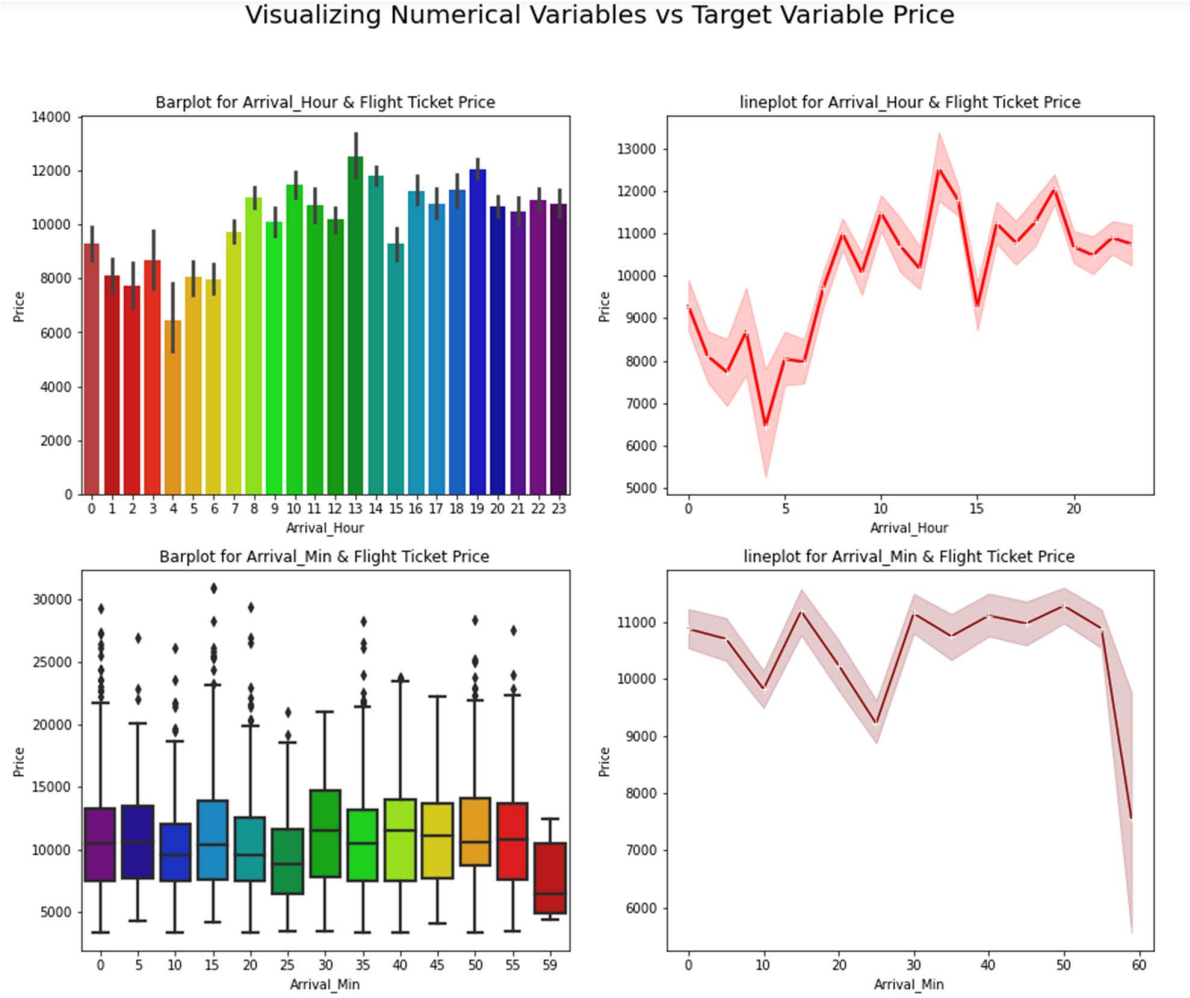
- **Airline vs Price**: From the bar plot we can notice "Vistara" and "Air India" airlines have highest ticket prices compared to other airlines.
- **Number_of_stops vs Price**: From the strip plot we can notice the flights which have 1 and 2 stops between source and destination have highest

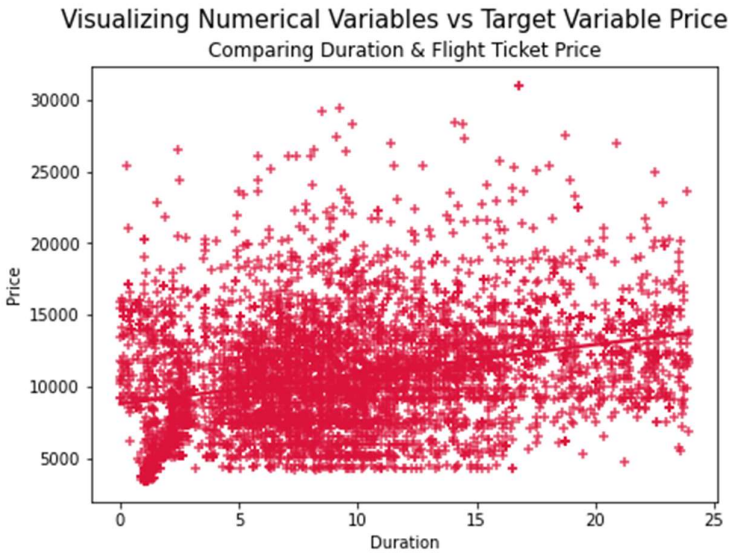
ticket prices compared to others. The airlines which have 4 stops during the journey have very less ticket price. So, we can say as the stops increases, ticket price decreases.

- **Source vs Price**: From the box plot we can observe the flights from Kolkata are having somewhat higher prices compared to other sources.
- **Destination vs Price**: From the boxen plot we can notice that the flights travelling to Goa have higher flight ticket prices.
- **Meal_availability vs Price**: The boxplot shows the flights having Free meal facility have high ticket prices.

Visualizing Numerical Variables vs Target Variable Price



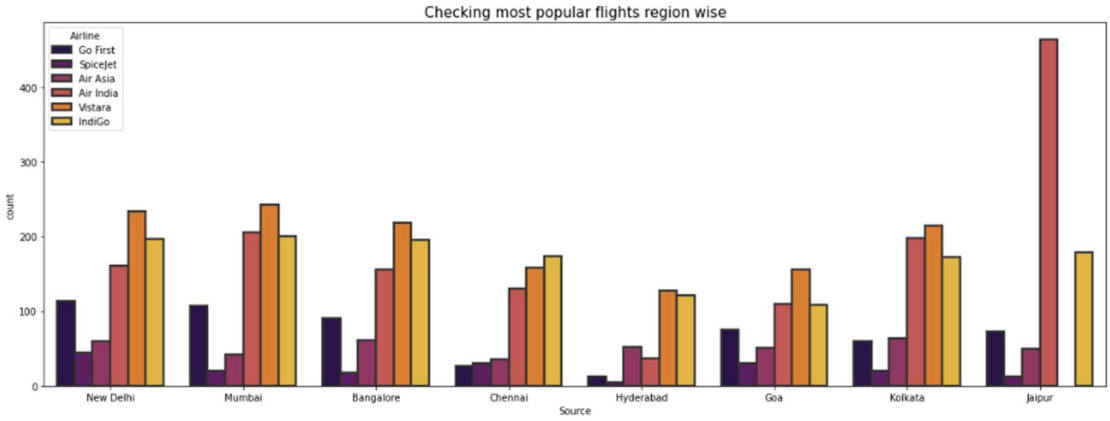


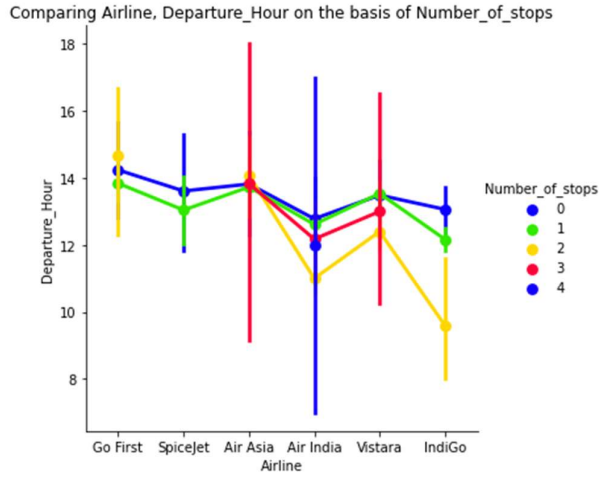


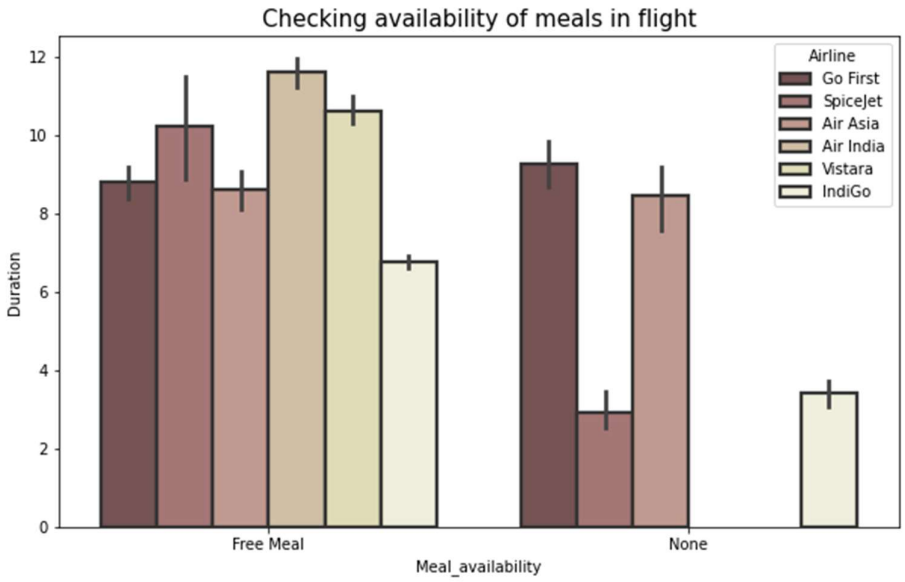
- **Departure_Hour vs Price**: From the bar plot and line plot we can see that there are some flights departing in the early morning 3 AM having most

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.
- **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 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. Flights having 1-12 hours of duration, they have ticket price of around 10000.







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

Airline vs Departure Hour: Above plot gives the relation between Airline and Departure hour based on Number of stops. Air India and Air Asia flights are departing in the evening and they have less than 4 stops during the journey.

Meal_Availability vs Duration: All the airlines provide free meals during the journey having the duration below 11 hours.

3.6 Interpretation of the Results

Visualizations: In univariate analysis we 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 we 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 we found some of the features skewed to right as well as to left. We 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. We have performed few processing steps which we 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, we performed train test split to build the model. We have built multiple regression models to get the accurate R2 score and evaluation metrics like RMSE. I got Extra Trees Regressor as the best model which gave the best R2 score. After tuning the best model, the R2 score of Extra Trees Regressor is 78.335% and also got low RMSE. Finally, we saved the final model and got the good predictions results for price of flight tickets.

4.CONCLUSION

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 flight price by building ML models. We Performed hyper parameter Tuning on the best model and got the best model's R2 score as 78.335%. 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 (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.

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 (bar and box) we came to know that early morning and late-night flights 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 increase 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.

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 after model building and hyper parameter tuning.

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 scrap the 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.

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 same 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).