

18MIS0190

18MIS0271



**VIT<sup>®</sup>**  
**UNIVERSITY**  
(Estd. u/s 3 of UGC Act 1956)

Winter Semester (2020-21)

Soft Computing (SWE1011)

**Project title: Flight Pricing Prediction Web**  
**Application using Machine Learning**

Review-03

**TEAM MEMBERS**

S MONISH KUMAR	18MIS0190
H JAGADEESH	18MIS0271

*Guided by*  
**Prof SUBHASHINI R**

# TABLE OF CONTENTS

Chapter	Name	Pg. No
1.	<ul style="list-style-type: none"><li>▪ Abstract</li><li>▪ Introduction</li><li>▪ Objective</li><li>▪ Scope</li><li>▪ Problem Definition and Approach</li></ul>	3-5
2.	<ul style="list-style-type: none"><li>▪ Literature Survey</li></ul>	6-10
3.	<ul style="list-style-type: none"><li>▪ Current Status</li><li>▪ Dataset Collection</li><li>▪ Methods we used</li></ul>	11-14
4.	<ul style="list-style-type: none"><li>▪ Techniques</li><li>▪ Development Tools</li></ul>	15
5.	<ul style="list-style-type: none"><li>▪ Architecture Diagrams</li></ul>	16-17
6.	<ul style="list-style-type: none"><li>▪ Implementation</li><li>▪ Final Result (Website)</li></ul>	17-41
7.	<ul style="list-style-type: none"><li>▪ Conclusion</li><li>▪ References</li></ul>	42-43

## ABSTRACT

Flight Price Prediction is the system which predicts the Flight price in earlier. airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we simulate various models for computing expected future prices. Airline's use using sophisticated tactics which they call "Revenue management" or "Yield management".

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) Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit.

Airlines use various kinds of computational techniques to increase their revenue such as demand prediction and price discrimination. From the customer side, two kinds of models are proposed by different researchers to save money for customers: models that predict the optimal time to buy a ticket and models that predict the minimum ticket price. In this paper, we present a review of customer side and airlines side prediction models. Our review analysis shows that models on both sides rely on limited set of features such as historical ticket price data, ticket purchase date and departure date.

Features extracted from external factors engine query are not considered. Therefore, we introduce and discuss the concept of using social media data for ticket/demand prediction.

## INTRODUCTION

As someone who purchases flights frequently, I would like to be able to predict when the best time is to buy in order to get the best deal. I have heard some people claiming that certain days of the week are when prices are lowest, and it's likely that that demand for flights is higher at certain hours, which could indicate higher prices. It's possible this varies depending on various other aspects, like the length of the flight, the date and time of the flight, and how much time there is until the flight. In this project, I chose to focus on aspects that are visible on the consumer side and predict only the binary class of whether the price will increase or not, which is essentially whether one should buy now or wait. The input is the time of the day of the request, the time of the week of the request, the time of the day of the flight, the time of the week of the flight, the number of stops, the duration of the flight, the number of hours between the request.

On the airlines side, the main goal is increasing revenue and maximizing profit. According to airlines utilize various kinds of pricing strategies to determine optimal ticket prices: long-term pricing policies, yield pricing which describes the impact of production conditions on ticket prices, and dynamic pricing which is mainly associated with dynamic adjustment of ticket prices in response to various influencing factors.

On the other hand, dynamic pricing enables a more optimal forecasting of ticket prices based on vibrant factors such as changes in demand and price discrimination. However, dynamic pricing is challenging as it is highly influenced by various factors including internal factors, external factors, competition among airlines and strategic customers. Internal factors consist of features such as historical ticket price data, ticket purchase date and departure date, season, holidays, supply (number of available airlines and flights), fare class, availability of seats, recent market demand and flight distance. External factors include features such as occurrence of some event at the origin or destination city like terrorist attacks, natural disaster (hurricane, earthquake, tsunami, etc.).

## OBJECTIVE

In this system, we are going to predict or classify the Flight ticket price using Regression method in Jupyter Notebook. And then, we are going to compare our proposed model with existing model. Main objective is to deduct flight ticket at low cost for the beneficiary of customer using Random Forest Regression and Random Search CV Regressor.

## SCOPE

Moreover, development of accurate prize prediction of flight tickets using prediction models for saving money regarding delay of flight, air ticket price, flight score and flight fair. Because of accurate delay of flight prediction problem, it has a scope of predicting cheapest price for customers regarding delay in Future.

## PROBLEM DEFINITION AND APPROACH:

- After we have the data, we need to clean & prepare the data according to the model's requirements. In any machine learning problem, this is the step that is the most important and the most time consuming. We used various statistical techniques & logics and implemented them using built-in R packages.
- Data preparation is followed by analyzing the data, uncovering hidden trends and then applying various predictive & classification models on the training set.
- Having built various models, we now have to test the models on our testing set and come up with the most suitable metric to calculate the accuracy. Moreover, many a times, merging models and predicting a cumulative target variable proves to be more accurate

## LITERATURE SURVEY

S. No	Title	Author	Summary
1.	A Survey on Flight Pricing Prediction using Machine Learning	Supriya Rajankar , Neha Sakharkar	<p>Abstract: What is the best time to buy a flight ticket? The airline implements dynamic pricing for the flight ticket. According to the survey, flight ticket prices change during the morning and evening time of the day. Also, it changes with the holidays or festival season. 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.</p> <p>AND THE ACCURACY VALUE SCORE=67.1%</p>
2.	Survey on Air Price Prediction using Machine Learning Algorithms	Abhilash, Ranjana Y, Shilpa S.	<p>Abstract: The existing airfare prediction method uses very complicated methods and algorithms for the prediction. They consider several financial and commercial factors and the prices change dynamically which makes it difficult for customers to purchase the air ticket. Airlines implement dynamic pricing for their tickets, and base their pricing decisions on demand estimation models.</p> <p>AND THE ACCURACY VALUE SCORE=61.8%</p>
3.	INDIAN FLIGHT FARE PREDICTION: A PROPOSAL	Jaywrat Singh Champawat, Dr. K. Vijaya	<p>ABSTRACT The price of a ticket of an airline changes very rapidly these days, and the difference is a lot. It can vary even in a few hours for the same flight or even the some particular. Customers want to get the cheapest price possible while the airline companies want the maximum profit and revenue possible. To solve this problem, researchers have proposed different models to save consumer's money- minimum price predicting model and models that tell us an optimal time to buy a ticket while airlines use techniques such as demand prediction and price discrimination to maximize their revenue.</p> <p>AND THE ACCURACY VALUE SCORE=61.8%</p>

S. No	Title	Author	Summary
4.	Flight Price using Machine Learning	Snehanshu Sengupta	<p>Abstract: 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. As data scientists, we are going to prove that given the right data anything can be predicted. Here you will be provided with prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.</p> <p>AND THE ACCURACY VALUE SCORE=73.8%</p>
5.	Airline ticket price and demand prediction	NazarZaki, FazadKhan,Juhar Ahmed.	<p>Abstract: Nowadays, airline ticket prices can vary dynamically and significantly for the same flight, even for nearby seats within the same cabin. Customers are seeking to get the lowest price while airlines are trying to keep their overall revenue as high as possible and maximize their profit. Airlines use various kinds of computational to increase their revenue such as demand prediction and price discrimination.</p> <p>AND THE ACCURACY VALUE SCORE=78.2%</p>
6.	A Machine Learning Approach to Predict Price of Airlines Tickets	Pranita Rajure ,Samiksha Bankar.	<p>ABSTRACT Air passengers (the buyers) are often looking for the best time period to purchase airfares to get as much saving as possible while airlines (the sellers) always try to maximize their revenues by revising different prices for the same service. The airline implements dynamic pricing for the flight ticket. Flight ticket prices change during the morning and evening time of the day. Also, it changes with the holidays or festival season. The price of an airline ticket is affected by a number of times, fuel price, etc. The sellers have all the necessary information (for example historical sale, market demand, customer profile, and behaviors) to make the decision whether to increase or decrease airfares at different times.</p> <p>AND THE ACCURACY VALUE SCORE=71.2%</p>

S. No	Title	Author	Summary
7.	Airfare prices prediction using machine learning techniques	K. Tziridis; Th . Kalampokas G. A. Papakostas	<p>Abstract: This paper deals with the problem of airfare prices prediction. For this purpose, a set of features characterizing a typical flight is decided, supposing that these features affect the price of an air ticket. The features are applied to eight states of the art machine learning (ML) models, used to predict the air tickets prices, and the performance of the models is compared to each other.</p> <p>Along with the prediction accuracy of each model, this paper studies the dependency of the accuracy on the feature set used to represent an airfare. For the experiments a novel dataset consisting of 1814 data flights of the Aegean Airlines for a specific international destination (from Thessaloniki to Stuttgart) is constructed and used to train each ML model.</p> <p>AND THE ACCURACY VALUE SCORE=88%</p>
8.	Design and implementation of ticket price forecasting system	Yuling Li, and Zhichao Li	<p>Abstract: With the advent of the aviation travel industry, a large number of data mining technologies have been developed to increase profits for airlines in the past two decades. The implementation of the digital optimization strategy leads to price discrimination, for example, similar seats on the same flight are purchased at different prices, depending on the time of purchase, the supplier, and so on. Price fluctuations make the prediction of ticket prices have application value</p> <p>AND THE ACCURACY VALUE SCORE=78.2%</p>
9.	Airfare prices prediction using machine learning techniques	K . Tziridis , T. Kalampokas	<p>ABSTRACT Along with the prediction accuracy of each model, this paper studies the dependency of the accuracy on the feature set used to represent an airfare. For the experiments a novel dataset consisting of 1814 data flights of the Aegean Airlines for a specific international destination model.</p> <p>AND THE ACCURACY VALUE SCORE=88%</p>



S. No	Title	Author	Summary
10.	Airline Price Prediction using Machine Learning	Jaya Shukla, Aditi Srivastava	<p>Abstract: Flight ticket prices can be somewhat hard to guess, as flight prices fluctuate constantly, so purchasing at different times could mean large differences in price. From the customer side, models are used to predict optimal time to buy a ticket and to predict the minimum ticket price. Customers are seeking to urge all-time low prices while airlines are attempting to keep their overall revenue as high as possible and maximize their profit. Airlines use different kinds of techniques to increase their interests like demand prediction and price inequity. For this purpose a set of features distinguishing a flight price is decided, presuming that these attributes influence the machine learning models, and the performance of the model is compared to each other.</p> <p>AND THE ACCURACY VALUE SCORE=80%</p>
11.	Airfare Price Prediction Based on Reviews Using Machine Learning Techniques	Aakanksha V. Jain, Aanal S.Raval, Ruchi K. Oza	<p>Abstract: This paper deals with the problem of correct airfare prices prediction and their reviews. Along with the prediction accuracy of each model, this paper studies the dependency of the accuracy on the feature set used to represent an airfare. For the experiment, dataset consisting for reviews, I take twitter and Skytrax website (2017- 2020). And for price dataset I take different Indian</p> <p>AND THE ACCURACY VALUE SCORE=94.36%</p>
12.	Prediction of Airline Ticket Price	Ruixuan Ren, Yunzhe Yang, Shenli Yuan	<p>ABSTRACT 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. Therefore, it is challenging for consumers to predict the price change in the future airlines use techniques such as demand prediction and price discrimination to maximize their revenue.</p> <p>AND THE ACCURACY VALUE SCORE=82.61%</p>

S. No	Title	Author	Summary
13.	Analysis of dynamic pricing in airlines and predicting least fare	Purbid Bambroo Dr. Kavitha Sooda, Nitin Agrawal	Abstract: Airline companies have been using dynamic pricing to vary the ticket prices to maximize the profit for a limited number of seats. Though the algorithm is different for all the airlines and never disclosed, it is possible to predict the variation in ticket prices. There have been studies in the past for the same, none explicitly for the Indian market; considering the major holidays. Applying techniques from Machine learning model of neural networks and back-propagation, we could predict the upcoming surge or dip in the ticket prices. We aim at predicting if the price of the ticket will go down in the future or the current price is the lowest. <a href="#">AND THE ACCURACY VALUE SCORE=87.42%</a>
14.	Flight price prediction for users by machine learning techniques	Pavithra maria k Anitha k l	Abstract: People who frequently travel through flight will have better knowledge on best discount and right time to buy the ticket. For the business purpose many airline companies change prices according to the seasons or time duration. They will increase the price when people travel more. Estimating the highest prices of the airlines data for the route is collected with features such as Duration, Source, Destination, Arrival, Departure. Features are taken from chosen dataset and in this paper, we have used machine learning techniques and regression strategies for prediction of the price wherein the airline price ticket costs vary overtime. <a href="#">AND THE ACCURACY VALUE SCORE=80%</a>
15.	Dynamic Pricing in the Airline Industry	Supriya Rajankar, Neha Sakharkar	ABSTRACT This paper has considered so many different theories and data that it is worth emphasizing a few highlights. Dynamic price discrimination is primarily driven by customer dynamics rather than price discrimination over an existing set of customers. <a href="#">AND THE ACCURACY VALUE SCORE=82.61%</a>

## CURRENT STATUS

To develop the model for the flight price prediction, many conventional machine learning algorithms are evaluated. They are as follows:

- Linear regression
- Decision tree
- Random Forest Algorithm
- K-Nearest neighbors
- Multilayer Perceptron
- Support Vector Machine (SVM)
- Gradient Boosting. All these models are implemented in the scikit learn. To evaluate the performance of this model, certain parameters are considered.

They are as follows: R-squared value, Mean Absolute Error (MAE) and Mean Squared Error (MSE). The formulas for these three parameters are as follows:

$$R^2 = 1 - \frac{\sum_{n=1}^t (y_i - \hat{y}_i)^2}{\sum_{n=1}^t (y_i - \bar{y}_i)^2} \quad (1)$$

$$MAE = \frac{1}{n} \sum_{n=1}^t |y_i - \hat{y}_i| \quad (2)$$

$$MSE = \frac{1}{n} \sum_{n=1}^t (y_i - \hat{y}_i)^2 \quad (3)$$

Algorithm we used here was Random Forest algorithm other algorithms can also used currently for this flight price prediction. Our metrics values as well as comparison to other values of the algorithm will be followed after the description of Random Forest Algorithm used in our project.

## DATASETS COLLECTION

The collection of data is the most important aspect of this project. There are various sources of the data on different websites which are used to train the models. Websites give information about the multiple routes, times, airlines and fare. Various sources from APIs to consumer travel websites are available for data scraping. In this section details of the various sources and parameters that are collected are discussed.

The script extracts the information from the website and creates a CSV file as output. This file contains the information with features and its details. Now an important aspect is to select the features that might be needed for the flight prediction algorithm. Output collected from the website contains numerous variables for each flight but not all are required, so only the following feature is considered.

- Departure Time,
- Arrival Time,
- Destination,
- Source,
- Additional Info like offers, delay
- Name of the Airline,
- Number of Stops

All the collected data needed a lot of work so after the collection of data, it is needed to be clean and prepare according to the model requirements. All the unnecessary data is removed like duplicates and null values. In all machine learning this technology, this is the most important and time-consuming step. Various statistical techniques and logic built-in python are used to clean and prepare the data.

[https://www.kaggle.com/nikhilmittal/flight-fare-prediction-mh?select=Data\\_Train.xlsx](https://www.kaggle.com/nikhilmittal/flight-fare-prediction-mh?select=Data_Train.xlsx)

## METHODOLOGIES WE USED

1. Define the problem and look at the big picture.
2. Perform Exploratory Data Analysis (EDA) to gain insights.
3. Clean and prepare data to better expose latent patterns within it.
4. Explore many different machine learning models and pick the best ones.
5. Perform model cross-validations to ensure that the analysis is robust.

### Random Forest Algorithm

It is a supervised learning algorithm. The benefit of the random forest is, it very well may be utilized for both characterization and relapse issue which structure most of current machine learning framework. Random forest forms numerous decision trees, what's more, adds them together to get an increasingly exact and stable expectation. Random Forest has nearly the equivalent parameters as a decision tree or a towing classifier model. It is very simple to discover the significance of each element on the expectation when contrasted with others in this calculation. The regular component in these techniques is, for the  $k$ th tree, a random vector  $\theta_k$  is produced, autonomous of the past random vector's  $\theta_1, \dots, \theta_{k-1}$  however with the equivalent distribution, while a tree is developed utilizing the preparation set and bringing about a classifier.  $x$  is an information vector.

For a period, in stowing the random vector is created as the includes in  $N$  boxes where  $N$  is the number of models in the preparation set of information. In random split, choice includes various autonomous random whole numbers between 1 to  $K$ . The dimensionality and nature of  $\theta$  rely upon its utilization in the development of a tree. After countless trees are created, they select the most famous class. These methodologies are called as random forests.

## Extra Trees Regressor

Then to check the important feature selection we using Extra Trees Regressor i.e... This class 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.

## Randomized SearchCV

Finally, for hyperparameter tuning we choose Randomized SearchCV method implements a “fit” and a “score” method. It also implements “score\_samples”, “predict”, “predict\_proba”, “decision\_function”, “transform” and “inverse\_transform” if they are implemented in the estimator used.

### Algorithm Evaluation Comparison

ML algorithms	R-squared	MAE	MSE
Random forest	0.67	0.08	0.04
Multilayer Perceptron	0.65	0.09	0.04
Gradient Boosting	0.47	0.13	0.06
Decision tree	0.45	0.09	0.06
K-nearest neighbour	0.38	0.14	0.07
SVM	0.19	0.15	0.08
AdaBoost	-0.12	0.21	0.11

## TECHNIQUES

### Libraries/Packages:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from sklearn.ensemble import ExtraTreesRegressor
```

```
from sklearn.model_selection import train_test_split
```

```
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn import metrics
```

```
from sklearn.model_selection import RandomizedSearchCV
```

```
import pickle
# open a file, where you want to store the data
file = open('flight_rf.pkl', 'wb')
```

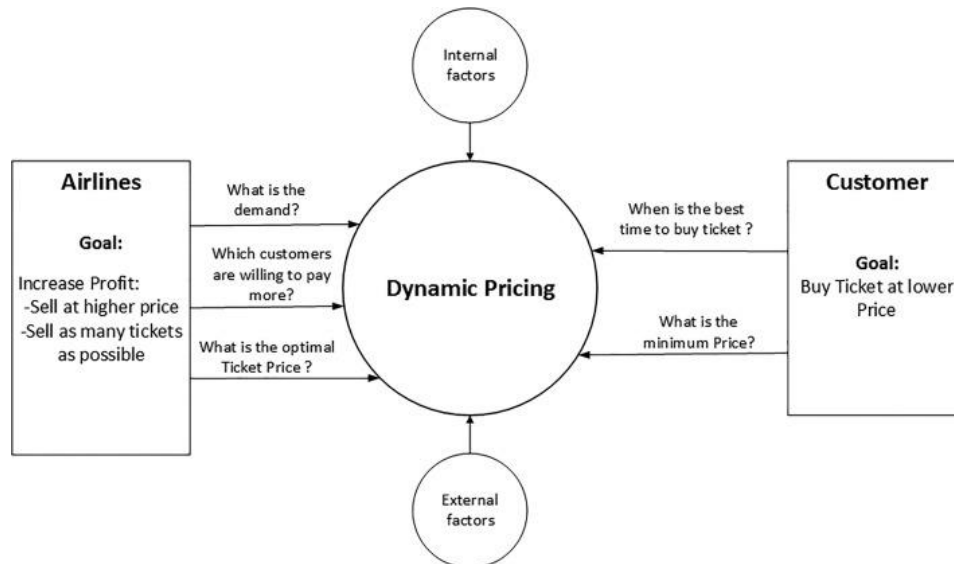
- Data visualization for data training in the model.
- Pickle for serialized the trained model and reuse it in python App
- Flask a micro web framework which is used for the code to run on a backend browser so that the all-packed model is made as a web application.
- For frontend design we using HTML CSS techniques.

### Development Tools:

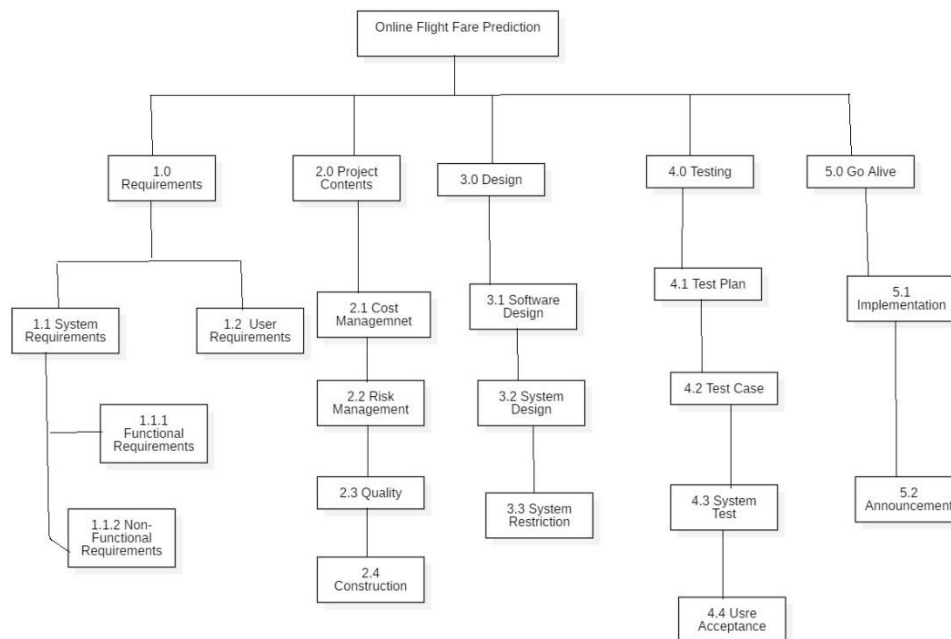
- Jupyter notebook collaborative work capabilities.
- Anaconda kit installer for installing all machine learning libraries.
- Sublime text tool for programming for frontend (HTML & CSS) as well as backend (python 3.8)
- Chrome or Firefox for deploy the project.

# ARCHITECTURE DIAGRAMS

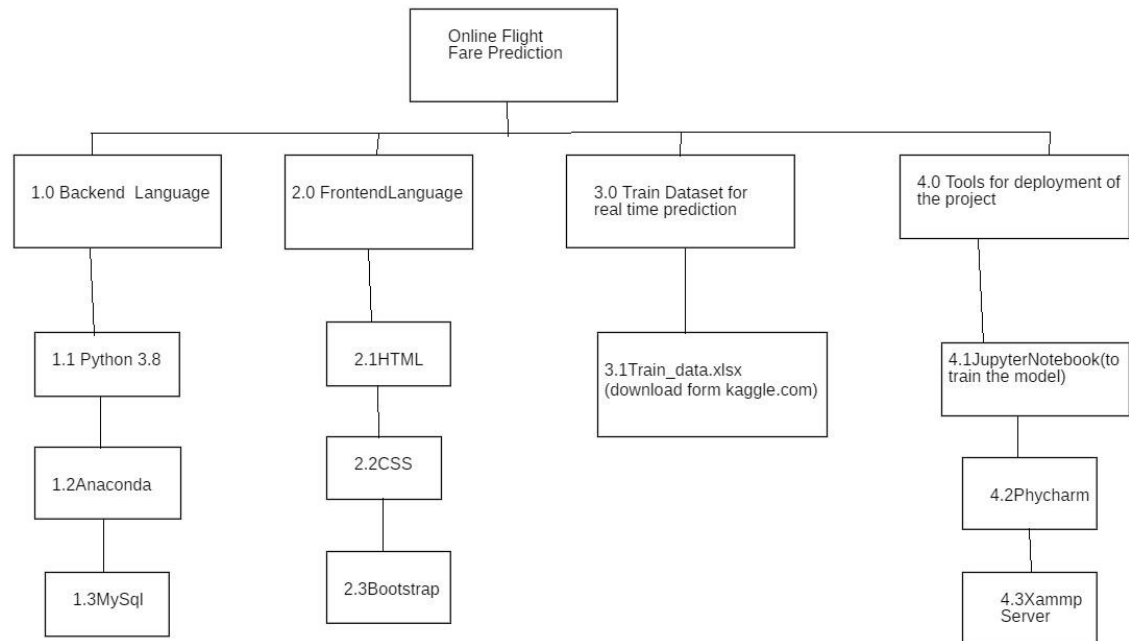
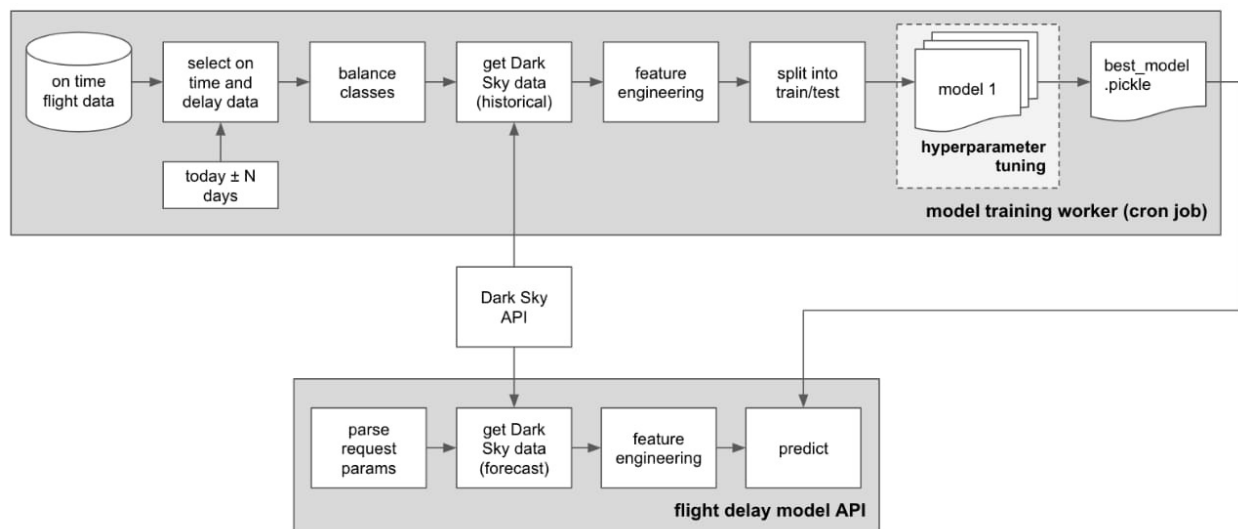
## Strategy Diagram:



## WorkBreakDown Diagram:





Product Breakdown Diagram:ML Diagram

18MIS0190

18MIS0271

# IMPLEMENTATION

## Dataset training and testing Models

### Flight Price Prediction

```
In [17]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

sns.set()
```

```
In [19]: pd.set_option('display.max_columns', None)
```

```
In [20]: train_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	1388
3	IndiGo	12/05/2019	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	01/03/2019	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	1330

```
In [21]: train_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10683 entries, 0 to 10682
Data columns (total 11 columns):
#   Column             Non-Null Count  Dtype  
---  -
0   Airline             10683 non-null object  
1   Date_of_Journey     10683 non-null object  
2   Source              10683 non-null object  
3   Destination         10683 non-null object  
4   Route               10682 non-null object  
5   Dep_Time            10683 non-null object  
6   Arrival_Time        10683 non-null object  
7   Duration            10683 non-null object  
8   Total_Stops         10682 non-null object  
9   Additional_Info     10683 non-null object  
10  Price               10683 non-null int64  
dtypes: int64(1), object(10)
memory usage: 918.2+ KB
```

18MIS0190

18MIS0271

```
In [22]: train_data["Duration"].value_counts()
```

```
2h 50m    550
1h 30m    386
2h 45m    337
2h 55m    337
2h 35m    329
...
41h 20m     1
35h 35m     1
30h 10m     1
3h 25m      1
37h 10m     1
Name: Duration, Length: 368, dtype: int64
```

```
In [23]: train_data.dropna(inplace = True)
```

```
In [24]: train_data.isnull().sum()
```

```
Airline      0
Date_of_Journey  0
Source       0
Destination  0
Route        0
Dep_Time     0
Arrival_Time  0
Duration     0
Total_Stops  0
Additional_Info  0
Price        0
dtype: int64
```

```
In [28]: # Since we have converted Date_of_Journey column into integers, Now we can drop as it is of no use.
```

```
train_data.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

```
In [29]: # Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time
```

```
# Extracting Hours
```

```
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour
```

```
# Extracting Minutes
```

```
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute
```

```
# Now we can drop Dep_Time as it is of no use
```

```
train_data.drop(["Dep_Time"], axis = 1, inplace = True)
```

```
In [29]: # Departure time is when a plane leaves the gate.
# Similar to Date_of_Journey we can extract values from Dep_Time
```

```
# Extracting Hours
```

```
train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour
```

```
# Extracting Minutes
```

```
train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute
```

```
# Now we can drop Dep_Time as it is of no use
```

```
train_data.drop(["Dep_Time"], axis = 1, inplace = True)
```

18MIS0190

18MIS0271

In [30]: train\_data.head()

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_mo
0	IndiGo	Banglore	New Delhi	BLR → DEL	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	3
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	13:15	7h 25m	2 stops	No info	7662	1	5
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	04:25 10 Jun	19h	2 stops	No info	13882	9	6
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	23:30	5h 25m	1 stop	No info	6218	12	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	21:35	4h 45m	1 stop	No info	13302	1	3

```
In [31]: # Arrival time is when the plane pulls up to the gate.
# Similar to Date_of_Journey we can extract values from Arrival_Time

# Extracting Hours
train_data["Arrival_hour"] = pd.to_datetime(train_data.Arrival_Time).dt.hour

# Extracting Minutes
train_data["Arrival_min"] = pd.to_datetime(train_data.Arrival_Time).dt.minute

# Now we can drop Arrival_Time as it is of no use
train_data.drop(["Arrival_Time"], axis = 1, inplace = True)
```

In [32]: train\_data.head()

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16

```

In [33]: # Time taken by plane to reach destination is called Duration
# It is the difference between Departure Time and Arrival time

# Assigning and converting Duration column into list
duration = list(train_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2: # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m" # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i] # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0])) # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1])) # Extracts only minutes from duration

In [34]: # Adding duration_hours and duration_mins list to train_data dataframe

train_data["Duration_hours"] = duration_hours
train_data["Duration_mins"] = duration_mins

In [35]: train_data.drop(["Duration"], axis = 1, inplace = True)

In [36]: train_data.head()

```

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min
0	IndiGo	Banglore	New Delhi	BLR → DEL	non-stop	No info	3897	24	3	22	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	2 stops	No info	7662	1	5	5	50
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	2 stops	No info	13882	9	6	9	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	1 stop	No info	6218	12	5	18	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	1 stop	No info	13302	1	3	16	50

18MIS0190

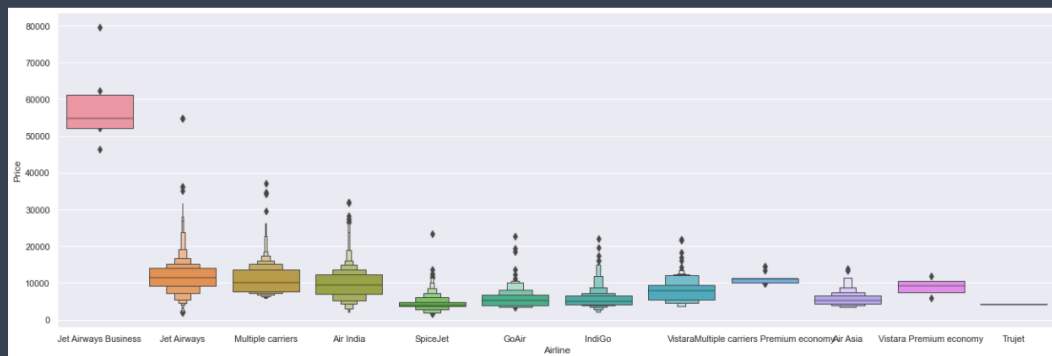
18MIS0271

```
In [37]: train_data["Airline"].value_counts()
```

```
Jet Airways      3849
IndiGo           2053
Air India        1751
Multiple carriers 1196
SpiceJet         818
Vistara          479
Air Asia         319
GoAir           194
Multiple carriers Premium economy 13
Jet Airways Business 6
Vistara Premium economy 3
Trujet          1
Name: Airline, dtype: int64
```

```
In [38]: # From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median

# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = train_data.sort_values("Price", ascending = False), kind="box",
plt.show()
```



```
In [39]: # As Airline is Nominal Categorical data we will perform OneHotEncoding
```

```
Airline = train_data[["Airline"]]

Airline = pd.get_dummies(Airline, drop_first= True)

Airline.head()
```

	Airline_Air India	Airline_GoAir	Airline_IndiGo	Airline_Jet Airways	Airline_Jet Airways Business	Airline_Multiple carriers	Airline_Multiple carriers Premium economy	Airline_SpiceJet	Airline_Vistara
0	0	0	1	0	0	0	0	0	0
1	1	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0

18MIS0190

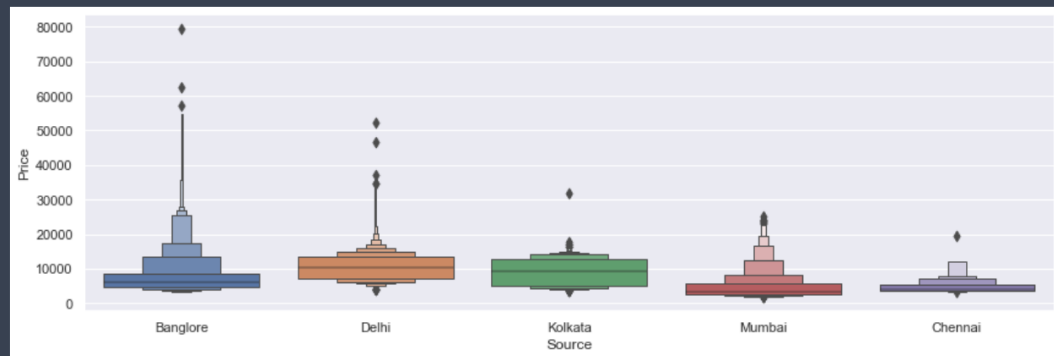
18MIS0271

```
In [40]: train_data["Source"].value_counts()
```

```
Delhi      4536
Kolkata    2871
Banglore   2197
Mumbai     697
Chennai    381
Name: Source, dtype: int64
```

```
In [41]: # Source vs Price
```

```
sns.catplot(y = "Price", x = "Source", data = train_data.sort_values("Price", ascending = False), kind="box",
plt.show())
```



```
In [42]: # As Source is Nominal Categorical data we will perform OneHotEncoding
```

```
Source = train_data[["Source"]]
```

```
Source = pd.get_dummies(Source, drop_first= True)
```

```
Source.head()
```

	Source_Chennai	Source_Delhi	Source_Kolkata	Source_Mumbai
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	0

```
In [43]: train_data["Destination"].value_counts()
```

```
Cochin      4536
Banglore    2871
Delhi       1265
New Delhi   932
Hyderabad   697
Kolkata     381
Name: Destination, dtype: int64
```

18MIS0190

18MIS0271

```
In [43]: train_data["Destination"].value_counts()
```

```
Cochin      4536
Banglore    2871
Delhi        1265
New Delhi    932
Hyderabad    697
Kolkata      381
Name: Destination, dtype: int64
```

```
In [44]: # As Destination is Nominal Categorical data we will perform OneHotEncoding
```

```
Destination = train_data[["Destination"]]
```

```
Destination = pd.get_dummies(Destination, drop_first = True)
```

```
Destination.head()
```

	Destination_Cochin	Destination_Delhi	Destination_Hyderabad	Destination_Kolkata	Destination_New Delhi
0	0	0	0	0	1
1	0	0	0	0	0
2	1	0	0	0	0
3	0	0	0	0	0
4	0	0	0	0	1

```
In [45]: train_data["Route"]
```

```
0          BLR → DEL
1    CCU → IXR → BBI → BLR
2    DEL → LKO → BOM → COK
3    CCU → NAG → BLR
4    BLR → NAG → DEL
...
10678    CCU → BLR
10679    CCU → BLR
10680    BLR → DEL
10681    BLR → DEL
10682    DEL → GOI → BOM → COK
Name: Route, Length: 10682, dtype: object
```

```
In [46]: # Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
```

```
train_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)
```

```
In [47]: train_data["Total_Stops"].value_counts()
```

```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```



18MIS0190

18MIS0271

```

In [48]: # As this is case of Ordinal Categorical type we perform LabelEncoder
# Here Values are assigned with corresponding keys

train_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)

In [49]: train_data.head()

```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35

```

In [50]: # Concatenate dataframe --> train_data + Airline + Source + Destination

data_train = pd.concat([train_data, Airline, Source, Destination], axis = 1)

In [51]: data_train.head()

```

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	10
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	15
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	25
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	30
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	35

```

In [52]: data_train.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)

In [53]: data_train.head()

```

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_min
0	0	3897	24	3	22	20	1	10	2	50
1	2	7662	1	5	5	50	13	15	7	25
2	2	13882	9	6	9	25	4	25	19	0
3	1	6218	12	5	18	5	23	30	5	25
4	1	13302	1	3	16	50	21	35	4	45

18MIS0190

18MIS0271

```
In [54]: data_train.shape
```

```
(10682, 30)
```

## Test set

```
In [55]: test_data = pd.read_excel(r"C:\Users\Jagadeesh Himayan\Desktop\SC\FPP\Test_set.xlsx")
```

```
In [56]: test_data.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
0	Jet Airways	6/06/2019	Delhi	Cochin	DEL → BOM → COK	17:30	04:25 07 Jun	10h 55m	1 stop	No info
1	IndiGo	12/05/2019	Kolkata	Banglore	CCU → MAA → BLR	06:20	10:20	4h	1 stop	No info
2	Jet Airways	21/05/2019	Delhi	Cochin	DEL → BOM → COK	19:15	19:00 22 May	23h 45m	1 stop	In-flight meal not included
3	Multiple carriers	21/05/2019	Delhi	Cochin	DEL → BOM → COK	08:00	21:00	13h	1 stop	No info
4	Air Asia	24/06/2019	Banglore	Delhi	BLR → DEL	23:55	02:45 25 Jun	2h 50m	non-stop	No info

```
In [57]: # Preprocessing
```

```
print("Test data Info")
```

```
print("-"*75)
```

```
print(test_data.info())
```

```
print()
```

```
print()
```

```
print("Null values :")
```

```
print("-"*75)
```

```
test_data.dropna(inplace = True)
```

```
print(test_data.isnull().sum())
```

```
# EDA
```

```
# Date_of_Journey
```

```
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey, format="%d/%m/%Y").dt.day
```

```
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
```

```
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)
```

```
# Dep_Time
```

```
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
```

```
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
```

```

# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)

# Duration
duration = list(test_data["Duration"])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:    # Check if duration contains only hour or mins
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m"    # Adds 0 minute
        else:
            duration[i] = "0h " + duration[i]            # Adds 0 hour

duration_hours = []
duration_mins = []
for i in range(len(duration)):
    duration_hours.append(int(duration[i].split(sep = "h")[0]))    # Extract hours from duration
    duration_mins.append(int(duration[i].split(sep = "m")[0].split()[-1]))    # Extracts only minutes from d

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)

# Categorical data

print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)

print()

print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)

print()

```

```

print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

# Replacing Total_Stops
test_data.replace({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4}, inplace = True)

# Concatenate dataframe --> test_data + Airline + Source + Destination
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)

data_test.drop(["Airline", "Source", "Destination"], axis = 1, inplace = True)

print()
print()

print("Shape of test data : ", data_test.shape)

```

```

Test data Info
-----
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2671 entries, 0 to 2670
Data columns (total 10 columns):
#   Column          Non-Null Count  Dtype
---  ---
0   Airline          2671 non-null   object
1   Date_of_Journey  2671 non-null   object
2   Source           2671 non-null   object
3   Destination      2671 non-null   object
4   Route            2671 non-null   object
5   Dep_Time         2671 non-null   object
6   Arrival_Time     2671 non-null   object
7   Duration         2671 non-null   object
8   Total_Stops      2671 non-null   object
9   Additional_Info  2671 non-null   object
dtypes: object(10)
memory usage: 208.8+ KB
None

```

Null values :

```

-----
Airline          0
Date_of_Journey  0
Source           0
Destination      0
Route            0
Dep_Time         0
Arrival_Time     0
Duration         0
Total_Stops      0
Additional_Info  0
dtype: int64
Airline
-----

```

18MIS0190

18MIS0271

```
-----
Jet Airways      897
Indigo           511
Air India        440
Multiple carriers 347
SpiceJet         208
Vistara          129
Air Asia         86
GoAir            46
Multiple carriers Premium economy 3
Vistara Premium economy 2
Jet Airways Business 2
Name: Airline, dtype: int64
```

Source

```
-----
Delhi      1145
Kolkata    710
Bangalore  555
Mumbai     186
Chennai    75
Name: Source, dtype: int64
```

Destination

```
-----
Cochin      1145
Bangalore   710
Delhi       317
New Delhi   238
Hyderabad   186
Kolkata     75
Name: Destination, dtype: int64
```

Shape of test data : (2671, 28)

```
In [58]: data_test.head()
```

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins
0	1	6	6	17	30	4	25	10	55
1	1	12	5	6	20	10	20	4	0
2	1	21	5	19	15	19	0	23	45
3	1	21	5	8	0	21	0	13	0
4	0	24	6	23	55	2	45	2	50

```
In [59]: data_train.shape
```

(10682, 30)

```
In [60]: data_train.columns
```

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Journey_month', 'Dep_hour',
      'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
      'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
      'Airline_Jet Airways', 'Airline_Jet Airways Business',
      'Airline_Multiple carriers',
      'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
      'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
      'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
      'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
      'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```

```
In [61]: X = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Journey_month', 'Dep_hour',
    'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
    'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
    'Airline_Jet Airways', 'Airline_Jet Airways Business',
    'Airline_Multiple carriers',
    'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
    'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
    'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
    'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
    'Destination_Kolkata', 'Destination_New Delhi']]

X.head()
```

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins
0	0	24	3	22	20	1	10	2	50
1	2	1	5	5	50	13	15	7	25
2	2	9	6	9	25	4	25	19	0
3	1	12	5	18	5	23	30	5	25
4	1	1	3	16	50	21	35	4	45

```
In [62]: y = data_train.iloc[:, 1]
y.head()
```

```
0    3897
1    7662
2   13882
3    6218
4   13302
Name: Price, dtype: int64
```

```
In [63]: # Finds correlation between Independent and dependent attributes

plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()
```

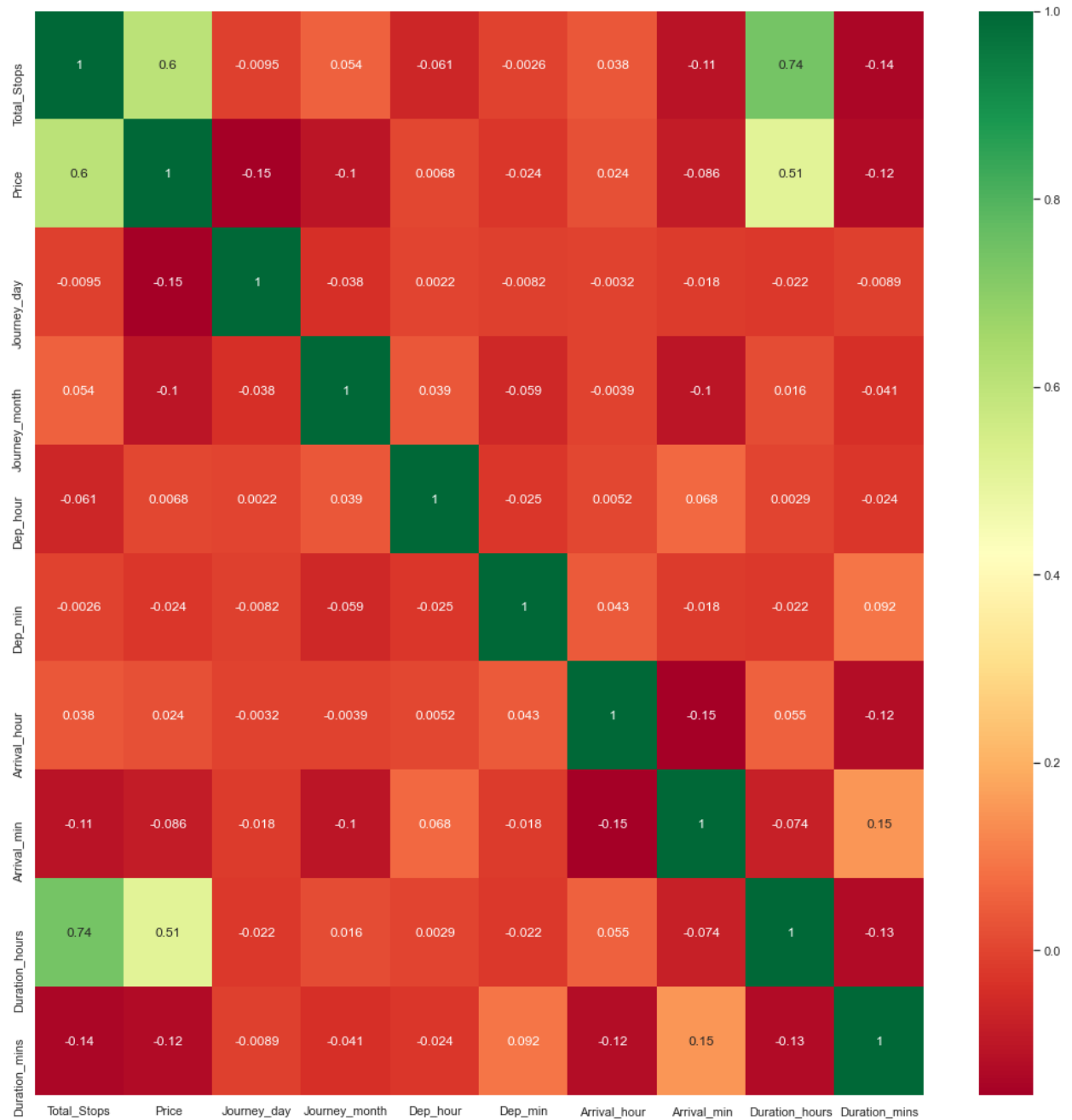
```
In [63]: # Finds correlation between Independent and dependent attributes

plt.figure(figsize = (18,18))
sns.heatmap(train_data.corr(), annot = True, cmap = "RdYlGn")

plt.show()
```

18MIS0190

18MIS0271



```
In [65]: # Important feature using ExtraTreesRegressor

from sklearn.ensemble import ExtraTreesRegressor
selection = ExtraTreesRegressor()
selection.fit(X, y)

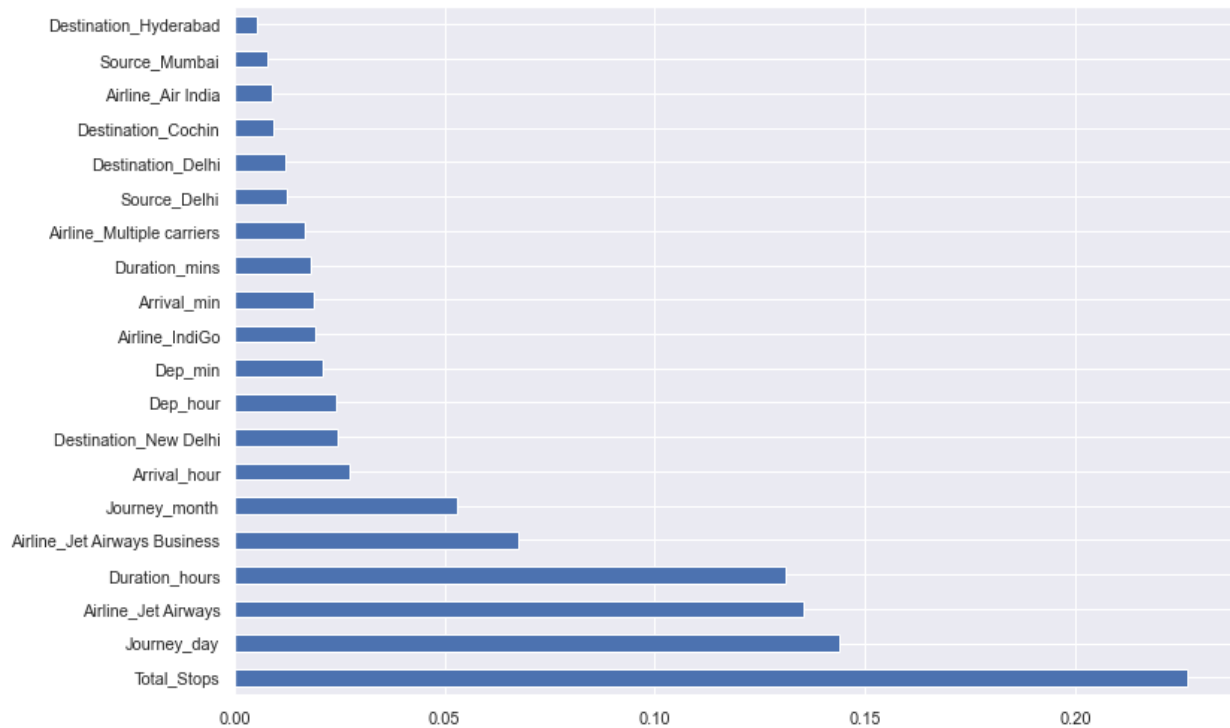
ExtraTreesRegressor()
```

```
In [66]: print(selection.feature_importances_)

[2.26540623e-01 1.43961577e-01 5.29076531e-02 2.42399448e-02
 2.08833407e-02 2.75666673e-02 1.89930471e-02 1.31286136e-01
 1.80410571e-02 8.95877297e-03 1.99263805e-03 1.90783858e-02
 1.35371354e-01 6.73905711e-02 1.68636195e-02 7.76800358e-04
 3.67578276e-03 1.01749653e-04 5.08306942e-03 9.57706528e-05
 4.65345219e-04 1.25357981e-02 3.14501574e-03 8.03025081e-03
 9.35643195e-03 1.21541090e-02 5.37541815e-03 4.97604100e-04
 2.46314662e-02]

In [67]: #plot graph of feature importances for better visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()
```





## Fitting model using Random Forest

1. Split dataset into train and test set in order to prediction w.r.t  $X_{test}$
2. If needed do scaling of data
  - Scaling is not done in Random forest
3. Import model
4. Fit the data
5. Predict w.r.t  $X_{test}$
6. In regression check **RSME** Score
7. Plot graph

```
In [68]: from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

```
In [69]: from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train, y_train)
```

```
RandomForestRegressor()
```

```
In [70]: y_pred = reg_rf.predict(X_test)
```

```
In [71]: reg_rf.score(X_train, y_train)
```

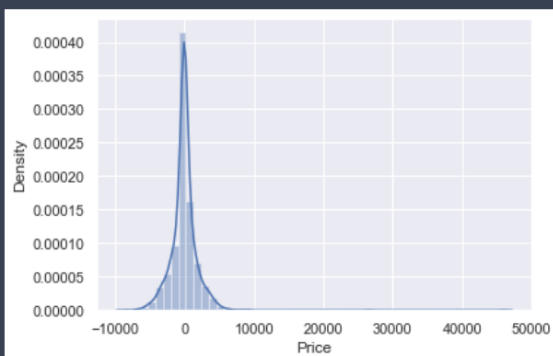
```
0.9537937407505536
```

```
In [72]: reg_rf.score(X_test, y_test)
```

```
0.7972044665410827
```

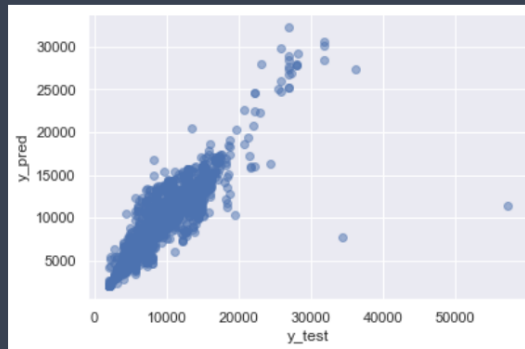
```
In [78]: sns.distplot(y_test-y_pred)
plt.show()
```

C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).  
warnings.warn(msg, FutureWarning)



In [79]:

```
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



In [80]:

```
from sklearn import metrics
```

In [81]:

```
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 1178.3934876524454
MSE: 4372688.714729612
RMSE: 2090.8974904890522
```

In [82]:

```
# RMSE/(max(DV)-min(DV))
```

```
2090.5509/(max(y)-min(y))
```

```
0.026887077025966846
```

In [83]:

```
metrics.r2_score(y_test, y_pred)
```

```
0.7972044665410827
```

## Hyperparameter Tuning

- Choose following method for hyperparameter tuning
  1. **RandomizedSearchCV** --> Fast
  2. **GridSearchCV**
- Assign hyperparameters in form of dictionary
- Fit the model
- Check best paramters and best score

In [84]:

```
from sklearn.model_selection import RandomizedSearchCV
```

```

In [85]: #Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]

In [86]: # Create the random grid

random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

In [87]: # Random search of parameters, using 5 fold cross validation,
# search across 100 different combinations
rf_random = RandomizedSearchCV(estimator = reg_rf, param_distributions = random_grid, scoring='neg_mean_squa

In [88]: rf_random.fit(X_train,y_train)

Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.

[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total= 3.9s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10

[Parallel(n_jobs=1)]: Done 1 out of 1 | elapsed: 3.8s remaining: 0.0s

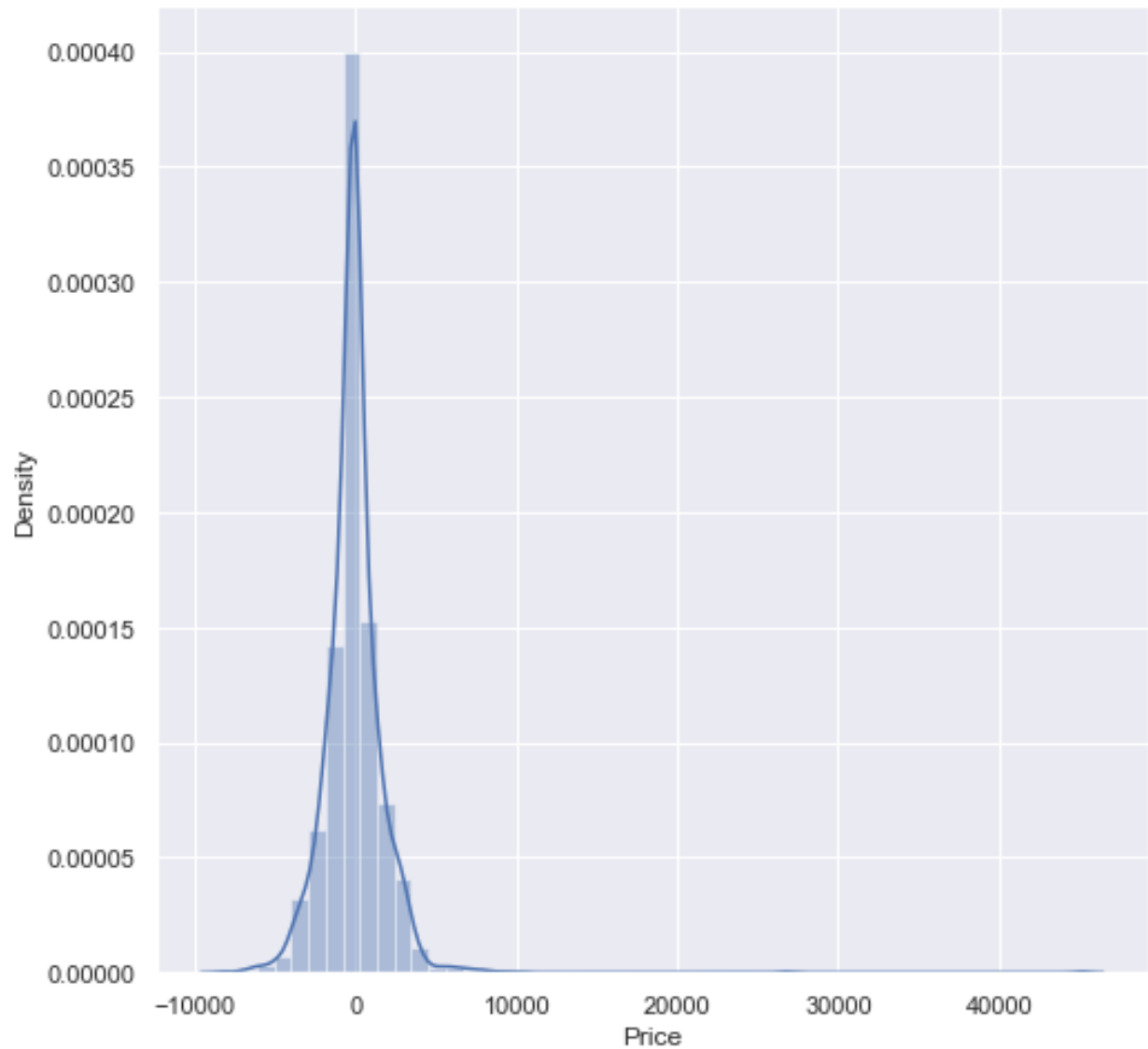
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total= 4.0s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total= 3.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total= 3.9s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10, total= 4.5s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total= 6.3s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total= 5.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total= 6.5s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total= 7.2s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=15, total= 7.3s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total= 4.0s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total= 3.7s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15, total= 3.6s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_features=auto, max_depth=15

```

18MIS0190

18MIS0271

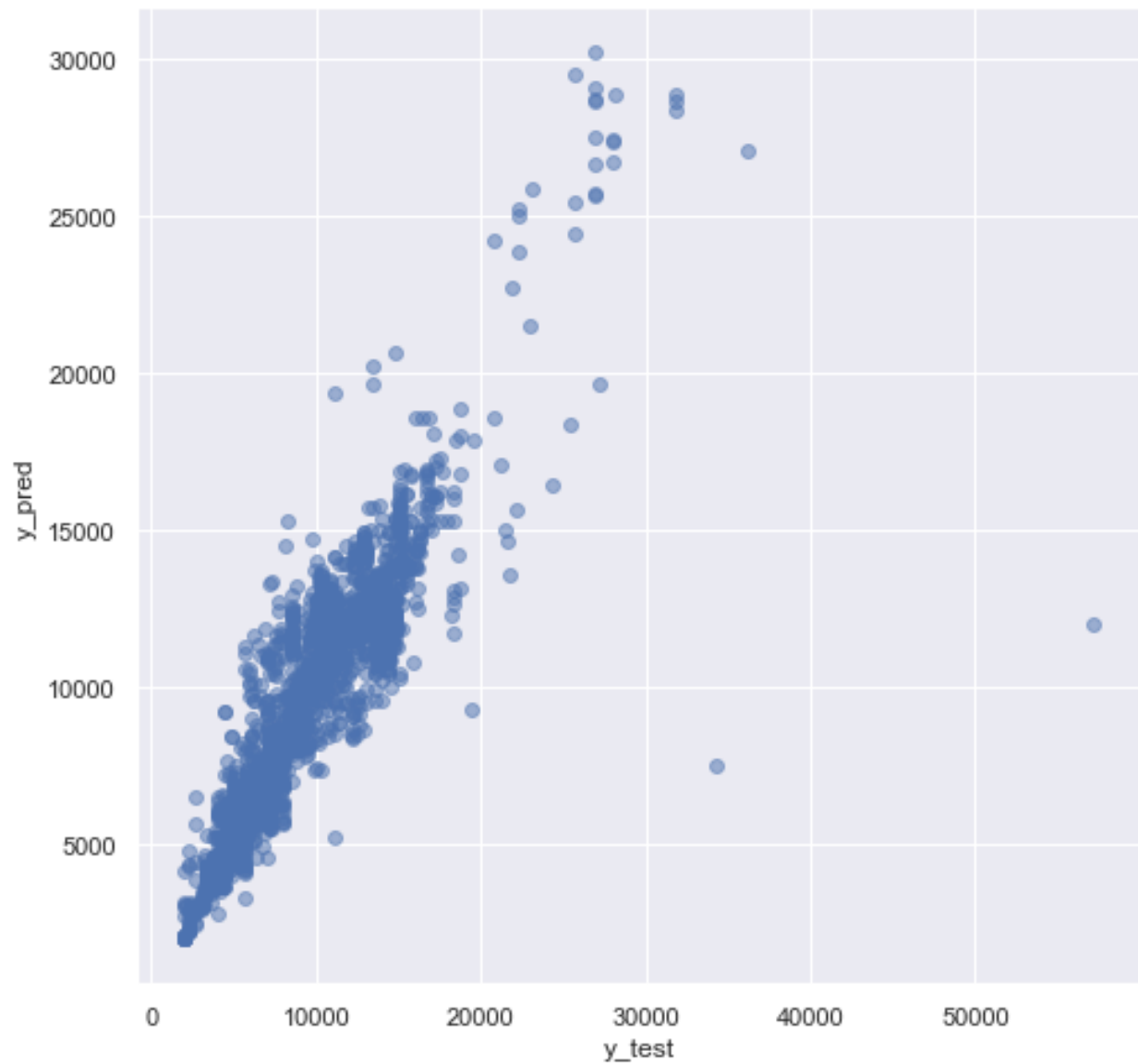
```
In [89]: rf_random.best_params_  
  
{'n_estimators': 700,  
 'min_samples_split': 15,  
 'min_samples_leaf': 1,  
 'max_features': 'auto',  
 'max_depth': 20}  
  
In [90]: prediction = rf_random.predict(X_test)  
  
In [91]: plt.figure(figsize = (8,8))  
sns.distplot(y_test-prediction)  
plt.show()
```



18MIS0190

18MIS0271

```
In [92]: plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



```
In [93]: print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 1166.293199169378
MSE: 4053297.8975700624
RMSE: 2013.2803822543106
```

## Save the model to reuse it again

```
In [94]: import pickle
         # open a file, where you want to store the data
         file = open('flight_rf.pkl', 'wb')

         # dump information to that file
         pickle.dump(reg_rf, file)
```

```
In [95]: model = open('flight_rf.pkl', 'rb')
         forest = pickle.load(model)
```

```
In [96]: y_prediction = forest.predict(X_test)
```

```
In [97]: metrics.r2_score(y_test, y_prediction)
```

```
0.7972044665410827
```

---

## Final Result (Website)



After training and testing the dataset model serialized the model and reuse it in our App python file.


Then using flask web micro framework, we used it as our backend to predict the flight price prediction by getting the inputs from the users.

Using HTML CSS, we designed the front-end by clicking some button it redirects to prediction page and display the predicted the prices in that page.

18MIS0190

18MIS0271






YAASSSS!  
MAKE IT EASY TO FLY BY  
USING OUR TOOL


○○○●○○

### Our great service


Flight Price Prediction is the system which predicts the Flight price in earlier. Airline ticket purchasing from the consumer's perspective is challenging principally because buyers have insufficient information for reasoning about future price movements. In this project we simulate various models for computing expected future prices.



**Honeymoon** ★★★★★  
📍 World ⌚ 7day 💰 \$ 2600



**mountain tour** ★★★★★  
📍 World ⌚ 7day 💰 \$ 2600



**Medical tour** ★★★★★  
📍 World ⌚ 7day 💰 \$ 2600

● ● ●

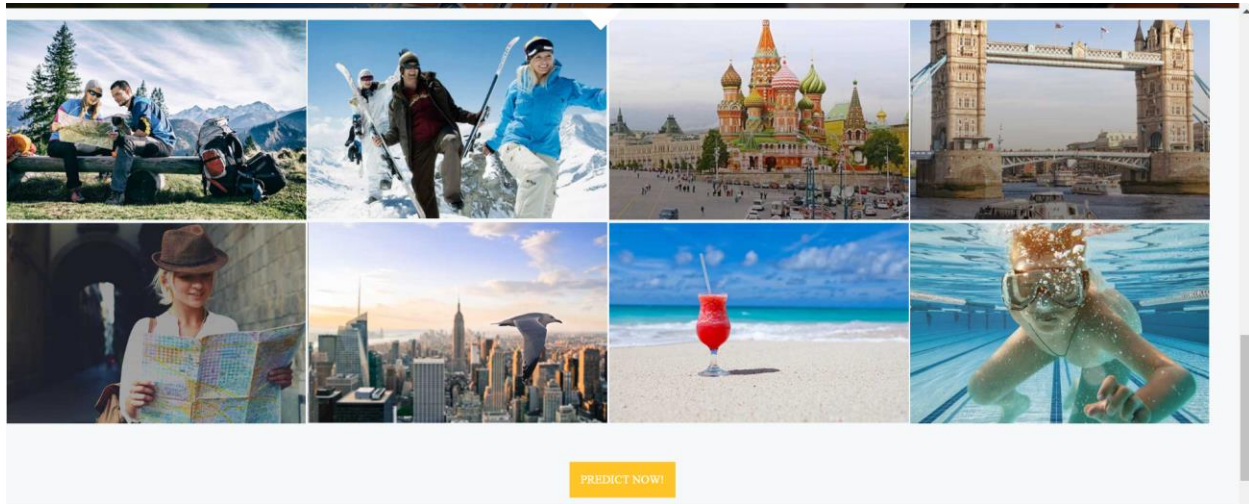
GREAT EXPERIENCE  
FOR TRAVELING AND TOURISM

Use our service to predict the prices!

TAKE YOUR TOUR

18MIS0190

18MIS0271



PREDICT NOW!

### Development Team

H JAGADEESH 18MIS0271  
S MONISH KUMAR 18MIS0255

#### Get in touch



#### OUR EMAIL ADDRESS

jagadeesh.h218@vit.student.ac.in  
moishmuar.s218@vit.student.ac.in

#### Drop your message

Your Name

Your Feedback

Send




18MIS0190


18MIS0271

FLIGHT PRICE


Departure Date

06/04/2021 12:00 PM 


Arrival Date

06/05/2021 12:00 AM 


Source

Chennai 


Destination

Kolkata 

Stopage


2 

Which Airline you want to travel?


IndiGo 

Submit


Departure Date

mm/dd/yyyy --:-- -- 


Arrival Date

mm/dd/yyyy --:-- -- 


Source

Delhi 


Destination

Cochin 

Stopage

Non-Stop 

Which Airline you want to travel?

Jet Airways 

Submit

Your Flight price is Rs. 11414.56

©2021 JAGADEESH MONISH

## CONCLUSION

In the proposed paper the overall survey for the dynamic price changes in the flight tickets is presented. this gives the information about the highs and lows in the airfares according to the days, weekend and time of the day that is morning, evening and night also, the machine learning models in the computational intelligence field that are evaluated before on different datasets are studied. their accuracy and performances are evaluated and compared in order to get better result. For the prediction of the ticket prices perfectly different prediction models are tested for the better prediction accuracy. As the pricing models of the company are developed in order to maximize the revenue management. So, to get result with maximum accuracy regression analysis is used. From the studies, the feature that influences the prices of the ticket are to be considered. In future the details about number of available seats can improve the performance of the model.

## REFERENCES

- [1] B. Smith, J. Leimkuhler, R. Darrow, and Samuels, "Yield management at american airlines," *Interfaces*, vol. 22, pp. 8–31, 1992.
- [3] T. Janssen, "A linear quantile mixed regression model for prediction of airline ticket prices," Bachelor Thesis, Radboud University, 2014.
- [4] R. Ren, Y. Yang and S. Yuan, "Prediction of airline ticket price," Technical Report, Stanford University, 2015
- [5] M. Papadakis, "Predicting Airfare Prices," 2014.
- [6] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5-32, 2001.
- [7] Viet Hoang Vu, Quang Tran Minh and Phu H. Phung, "An Airfare Prediction Model for Developing Markets", IEEE paper 2018.
- [8] S.B. Kotsiantis, "Decision trees: a recent overview," *Artificial Intelligence Review*, vol. 39, no. 4, pp. 261-283, 2013.
- [9] L. Breiman, "Random forests," *Machine Learning*, vol. 45, pp. 5-32, 2001.
- [10] S. Haykin, *Neural Networks – A Comprehensive Foundation*. Prentice Hall, 2nd Edition, 1999.
- [11] H. Drucker, C.J.C. Burges, L. Kaufman, A. Smola and V. Vapnik, "Support vector regression machines," *Advances in neural information processing systems*, vol. 9, pp. 155-161, 1997.
- [12] [www.Makemytrip.com](http://www.Makemytrip.com)
- [13] K. Tziridis, Th. Kalampokas, G.A. Papakotas and K.I. Diamantaras, "Airfare Prices Prediction Using Machine Learning Techniques", *EUSIPCO* 2017.
- [14] Yudong Tan and Huimin Zhou, "A Bayesian Predictor of Airline Class Seats Based on Multinomial Event Model", *IEEE on Big Data* 2016.
- [15] Wohlfarth, T. Clemencon, S. Roueff, "A Data mining approach to travel price forecasting", 10 th international conference on machine learning Honolulu 2011.
- [16] Dominguez-Menchero, J. Santo, Riviera, "optimal purchase timing in airline markets", 2014
- [17] Bingchuan Liu, Yudong Tan and Huimin Zhou, "A Bayesian predictor of Airline class Seats Based on Multinomial Event Model," *International conference on Big Data* 2016.
- [18] K. Tziridis, K.I. Diamantaras, "Airfare Prices Prediction Using machine Learning Technique", *European signal processing conference* 2017.
- [19] Viet Hong Vu, Phu Phung, "An airfare Prediction model for developing Markets", *IEEE* 2018
- [20] William Grooves and Maria Gini, "A regression model for predicting optimal purchase timing for airline tickets", *University of Minnesota* 2011.

**REFERENCE LINK (Literature Survey):**

1. <https://www.ijert.org/a-survey-on-flight-pricing-prediction-using-machine-learning>
2. <https://medium.com/code-to-express/flight-price-prediction-7c83616a13bb>
3. [https://www.academia.edu/45565822/A\\_Machine\\_Learning\\_Approach\\_to\\_Predict\\_Price\\_of\\_Airlines\\_Tickets](https://www.academia.edu/45565822/A_Machine_Learning_Approach_to_Predict_Price_of_Airlines_Tickets)
4. <https://ieeexplore.ieee.org/document/8081365>
5. <https://www.semanticscholar.org/paper/Airfare-prices-prediction-using-machine-learning-Tziridis->
6. <https://iarjset.com/papers/flight-price-prediction-for-users-by-machine-learning-techniques/>
7. <https://www.google.com/url?q=https://www.ijariit.com/manuscripts/v5i3/V5I3I150.pdf&usg=AOvVaw2LnUztFamuW>
8. <https://www.google.com/url?q=http://cs229.stanford.edu/>
9. <https://www.google.com/url?q=https://www.ijirset.com/upload/2020/march/>
10. <https://www.google.com/url?q=http://indusedu.org/pdfs/IJREISS/>
11. <https://www.google.com/url?q=https://aip.scitation.org/doi/pdf/>
12. <https://www.google.com/url?q=https://www.eurasip.org/Proceedings/Eusipco/Eusipco2017/papers/>
13. <https://www.google.com/url?q=https://ijireeice.com/wp-content/uploads/2019/03/>
14. [https://www.google.com/url?q=http://www.ijates.com/images/short\\_pdf/](https://www.google.com/url?q=http://www.ijates.com/images/short_pdf/)
15. <https://ieeexplore.ieee.org/document/8081365>

**THANK YOU**