Flight Fare Prediction

# Introduction

The flight ticket buying system is to purchase a ticket many days prior to flight takeoff so as to stay away from the effect of the most extreme charge. Mostly, aviation routes don’t agree this procedure. Plane organizations may diminish the cost at the time, they need to build the market and at the time when the tickets are less accessible. They may maximize the costs. So the cost may rely upon different factors. To foresee the costs this venture uses AI to exhibit the ways of flight tickets after some time. All organizations have the privilege and opportunity to change its ticket costs at any time. Explorer can set aside cash by booking a ticket at the least costs. People who had travelled by flight frequently are aware of price fluctuations. The airlines use complex policies of Revenue Management for execution of distinctive evaluating systems. The evaluating system as a result changes the charge depending on time, season, and festive days to change the header or footer on successive pages. The ultimate aim of the airways is to earn profit whereas the customer searches for the minimum rate. Customers usually try to buy the ticket well in advance of departure date so as to avoid hike in airfare as date comes closer. But actually this is not the fact. The customer may wind up by giving more than they ought to for the same seat

Main Objective

***The objective of this project is to predict the fare of a flight journey based on various parameters such as source, destination, no. of stops, arrival and departure time etc.***

This project is all about estimating the minimum airfare, data for a specific air route has been collected including the features like departure time, arrival time and airways over a specific period. Features are extracted from the collected data to apply Machine Learning (ML) models. In this project the machine learning **regression** methods to **predict the prices** at the given time.

Dataset

This is a kaggle flight fare dataset whch has prices of flight tickets for various airlines between the months of March and June of 2019 and between various cities.

**Datafile**

**https://www.kaggle.com/nikhilmittal/flight-fare-prediction-mh**

* Size of training set: 10683 records
* Size of test set: 2671 records
* Data is in form of excel file we have to use pandas read\_excel to load the data

## Attribute Information:

***Airline***: The name of the airline.(*categorical*)

***Date\_of\_Journey***: The date of the journey.(*categorical*)

***Source***: The source from which the service begins. .(*categorical*)

***Destination***: The destination where the service ends. .(*categorical*)

***Route***: The route taken by the flight to reach the destination. .(*categorical*)

***Dep\_Time***: The time when the journey starts from the source. .(*categorical*)

***Arrival\_Time***: Time of arrival at the destination. .(*categorical*)

***Duration***: Total duration of the flight. .(*categorical*)

***Total\_Stops***: Total stops between the source and destination. .(*categorical*)

***Additional\_Info***: Additional information about the flight.(*categorical*)

Target

***Price***: The price of the ticket(*numerical*)

Work flow in short

1. Getting the system ready and loading the data
2. Understanding the data
3. Missing Value Treatment
4. Exploratory Data Analysis (EDA)
5. Data Preprocessing
6. Feature Selection
7. Model Building:

* Random Forest

1. Hyperparameter Tuning
2. Saving Model as pickle file
3. Deployment in Heroku

**Loading Data**

Since data is in form of excel file we have to use pandas read\_excel to load the data

**Understanding the data and cleaning**

Except target variable ,all other features are categorical variables

It also contain missing values

**EDA(Exploratory data Analysis)**

**Data Preprocessing**

* Removing the null values from the data
* **Date\_of\_Journey** is a object data type,  
  Therefore, converting this datatype into timestamp so as to use this column properly for prediction.For this we use pandas **to\_datetime** to convert object data type to datetime dtype.
* **.dt.day method will extract only day of that date**
* **.dt.month method will extract only month of that date**
* Dropping **Date\_of\_Journey** as we have converted it to integer
* instead we have 2 columns **Journey\_day** and **Journey\_month**
* **Dep\_Time** is Departure time is when a plane leaves the gate.

* Extracting hour and minute values from Dep\_Time,
* And we get 2 columns **Dept\_min** and **Dept\_hour**
* Dropping **Dep\_Time** column
* **Arrival\_Time** is Departure time is when a plane leaves the gate.

* Extracting hour and minute values from Arrival\_Time,
* And we get 2 columns **Arrival \_min** and **Arrival \_hour**
* Dropping **Arrival \_Time** column
* **Duration** is time taken by plane to reach destination .It is the difference between Departure Time and Arrival time
* Assigning and converting Duration column into list
* Check if duration contains only hour or minutes
* Extracting hours as **Duration\_hours** and minutes as from duration**Duration\_mins**
* Dropping Duration

## **Handling Categorical Data**

There are many ways to handle categorical data. Some of them categorical data are,

1. **Nominal data** --> data are not in any order --> **OneHotEncoder** is used in this case
2. **Ordinal data** --> data are in order --> **LabelEncoder** is used in this case

Remaining categorical columns are **Airline, Source ,Destination, Additional\_Info ,Route and Total\_Stops**

* **Airlines :**Ploting a catplot
  + From graph we can see that **Jet Airways** Business have the highest Price.
  + Apart from the first Airline almost all are having similar median
  + As Airline is Nominal Categorical data we will perform **OneHotEncoding**
* **Source :** Ploting a catplot
* As Source is Nominal Categorical data we will perform **OneHotEncoding**
* **Destination :** Ploting a catplot
* As Destination is Nominal Categorical data we will perform **OneHotEncoding**
* **Additional\_Info** contains almost 80% no\_info,so dropping it
* **Route and Total\_Stops** are related to each other,so dropping **Route**
* **Total\_Stops :**
  + - * As this is case of Ordinal Categorical type we perform LabelEncoder
      * Here Values are assigned with corresponding keys
      * Replacing ({"non-stop": 0, "1 stop": 1, "2 stops": 2, "3 stops": 3, "4 stops": 4})
* Concatenate dataframe --> **train\_data + Airline + Source + Destination**

**And dropping the original colmns**

* Same preprocessing is done to **test data** also

**Feature Selection**

* Finding out the best feature which will contribute and have good relation with target variable.Following are some of the feature selection methods

1**. heatmap**

**2. feature\_importance\_**

**3. SelectKBest**

* Finding correlation between Independent and dependent attributes using heatmap
* Highly correlated features are **Duration\_hours &Total\_Stops,**
* **Duration\_hours & Price,and Price and Total\_Stops**
* Finding **feature importance** using **ExtraTreeRegressor**-This basically helps to find the important features ,ie ,those features which are highly correlated with the target variable

## **Fitting model using Random Forest**

* + As this is a regression problem we fit RandomForestRegressor to our train data and test data

Train Data Score -- 93.36

Test Data Score – 79.81

* + - Ploting the results using seaborn and its also shows a Gaussian Distribution(good result)
    - Also calculated the metrics like mean absolute error,Mean squared error and root mean square error
* **MAE**: 1177.186
* **MSE**: 4362789.078
* **RMSE**: 2088.729

**Hyperparameter tuning**

* To improve the model score we do hyperparameter tuning using Randomized Search Cross validation(5 fold cross validation)
* Following are the hyperparameter of Random Forest
* **'n\_estimators': n\_estimators,**
* **'max\_features': max\_features,**
* **'max\_depth': max\_depth,**
* **'min\_samples\_split': min\_samples\_split,**
* **'min\_samples\_leaf': min\_samples\_leaf}**
* By applying the best parameter values scores are increased to

MAE: 1165.606

MSE: 4062650.691

RMSE: 2015.601

**Saving the model**

* Saving the model for future use

**Deployment in Heroku**

**link**