FLIP ROBO

# flight_Price_prediction

Submitted by:

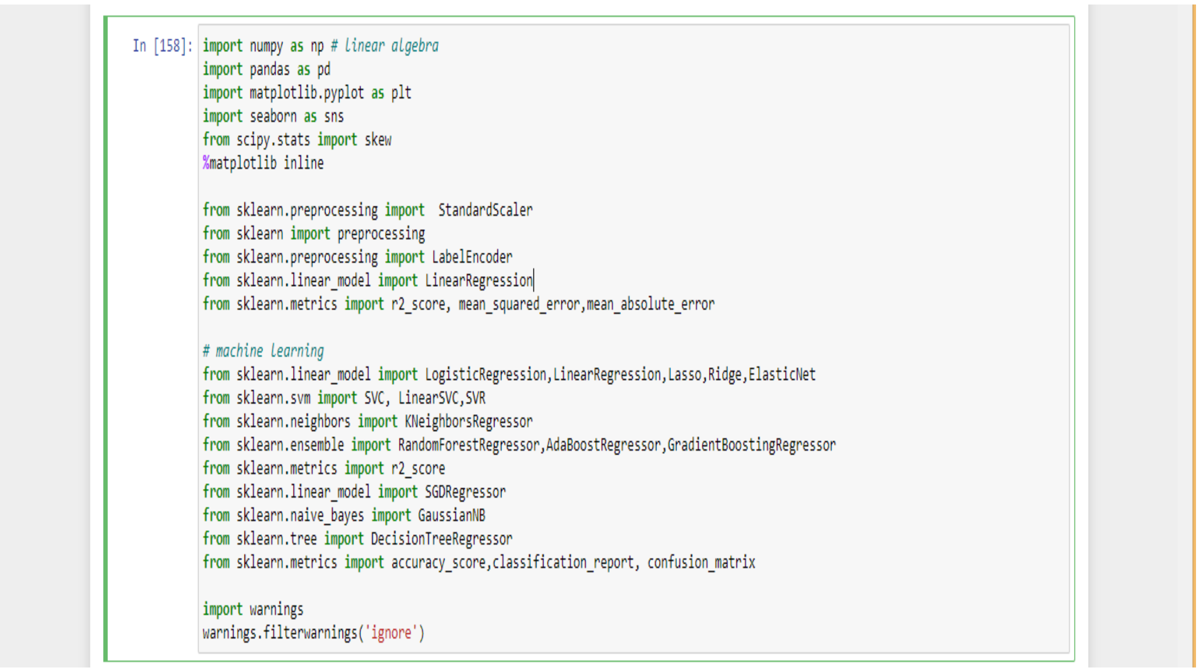
Naincy Joshi

# PROBLEM DEFINITION:

## 1.PROJECT OVERVIEW:

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 gonna prove that given the right data anything can be predicted.

## Importing Relevant Libraries:



In [158]:

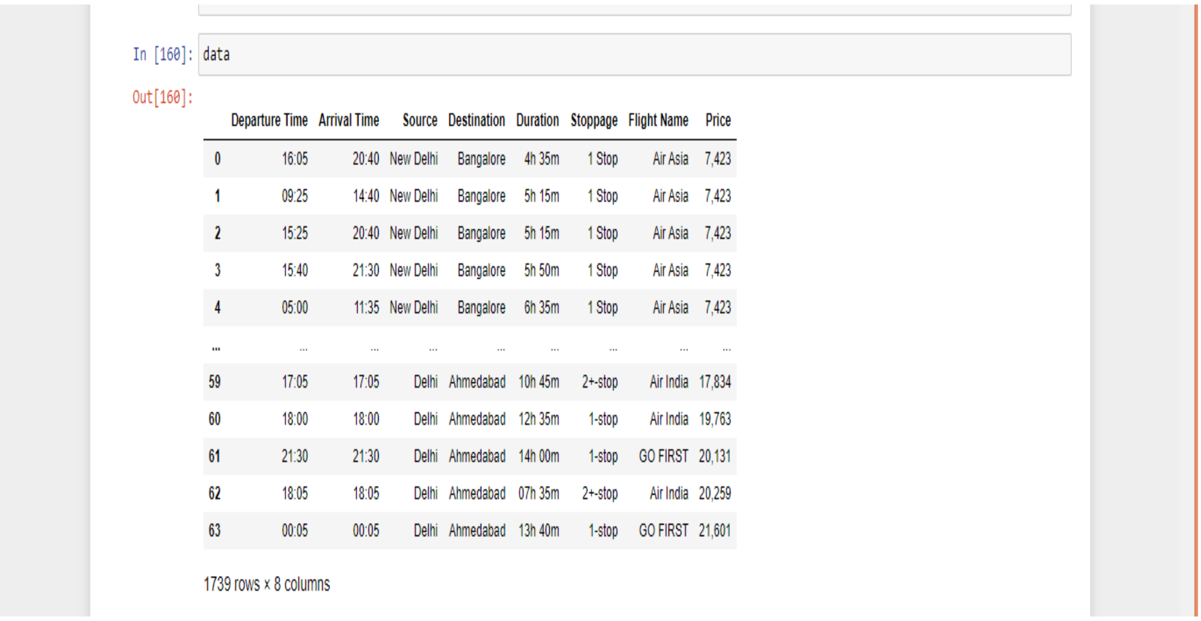
```
import numpy as np # linear algebra
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import skew
%matplotlib inline

from sklearn.preprocessing import  StandardScaler
from sklearn import preprocessing
from sklearn.preprocessing import LabelEncoder
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error,mean_absolute_error

# machine learning
from sklearn.linear_model import LogisticRegression,LinearRegression,Lasso,Ridge,ElasticNet
from sklearn.svm import SVC, LinearSVC,SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.ensemble import RandomForestRegressor,AdaBoostRegressor,GradientBoostingRegressor
from sklearn.metrics import r2_score
from sklearn.linear_model import SGDRegressor
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import accuracy_score,classification_report, confusion_matrix

import warnings
warnings.filterwarnings('ignore')
```

# Dataset details:



```
In [160]: data
Out[160]:
```

| | Departure Time | Arrival Time | Source | Destination | Duration | Stoppage | Flight Name | Price |
|---|---|---|---|---|---|---|---|---|
| 0 | 16:05 | 20:40 | New Delhi | Bangalore | 4h 35m | 1 Stop | Air Asia | 7,423 |
| 1 | 09:25 | 14:40 | New Delhi | Bangalore | 5h 15m | 1 Stop | Air Asia | 7,423 |
| 2 | 15:25 | 20:40 | New Delhi | Bangalore | 5h 15m | 1 Stop | Air Asia | 7,423 |
| 3 | 15:40 | 21:30 | New Delhi | Bangalore | 5h 50m | 1 Stop | Air Asia | 7,423 |
| 4 | 05:00 | 11:35 | New Delhi | Bangalore | 6h 35m | 1 Stop | Air Asia | 7,423 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 59 | 17:05 | 17:05 | Delhi | Ahmedabad | 10h 45m | 2+-stop | Air India | 17,834 |
| 60 | 18:00 | 18:00 | Delhi | Ahmedabad | 12h 35m | 1-stop | Air India | 19,763 |
| 61 | 21:30 | 21:30 | Delhi | Ahmedabad | 14h 00m | 1-stop | GO FIRST | 20,131 |
| 62 | 18:05 | 18:05 | Delhi | Ahmedabad | 07h 35m | 2+-stop | Air India | 20,259 |
| 63 | 00:05 | 00:05 | Delhi | Ahmedabad | 13h 40m | 1-stop | GO FIRST | 21,601 |

1739 rows × 8 columns

# Data-Preprocessing:

In the pre-processing  we have found that all the datatype are of “object”,so we need to convert into the machine learning language.

Then we check whether data contains the missing value or not through the data. isna().sum() which give us missing value. In this dataset there is no missing value.

# Feature Engineering:

- Removing the symbol and comma in the price column

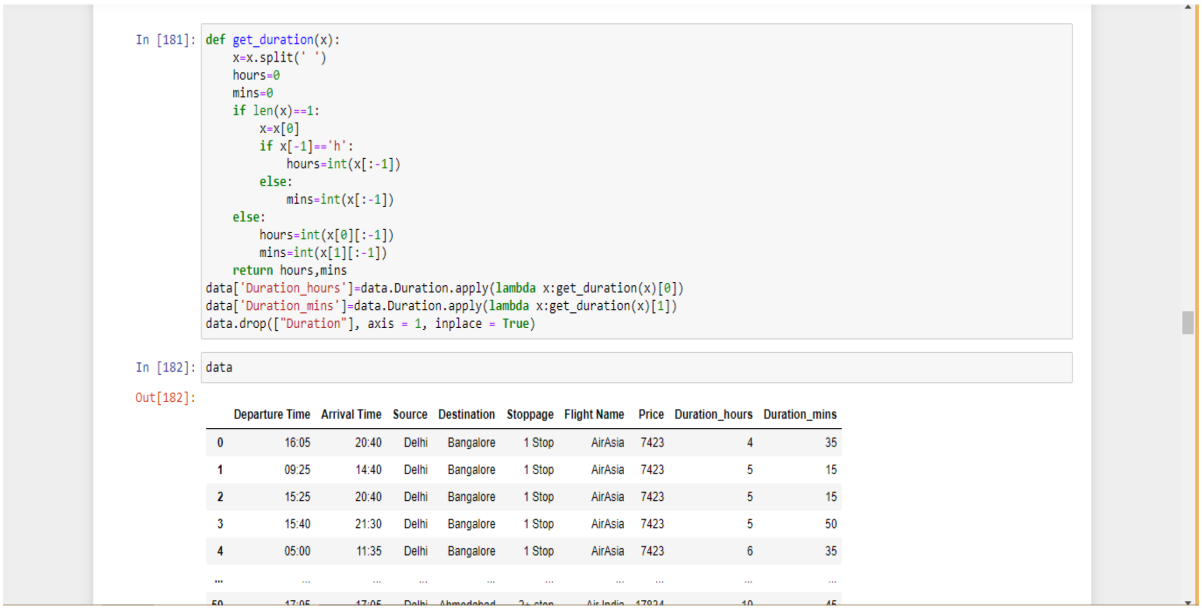
```
data['Price'] = data['Price'].str.replace(',', '')
```

```
data['Price'] = data['Price'].str.replace('₹', '')
```

- New Delhi and the Delhi are the same, so we will be replacing New Delhi with  Delhi

```
data['Source']=data['Source'].str.replace('New Delhi',' Delhi')
```

We will be adding the new column by splitting duration into duration hour and duration minutes.



```
In [181]: def get_duration(x):
              x=x.split(' ')
              hours=0
              mins=0
              if len(x)==1:
                  x=x[0]
                  if x[-1]=='h':
                      hours=int(x[:-1])
                  else:
                      mins=int(x[:-1])
              else:
                  hours=int(x[0][:-1])
                  mins=int(x[1][:-1])
              return hours,mins
          data['Duration_hours']=data.Duration.apply(lambda x:get_duration(x)[0])
          data['Duration_mins']=data.Duration.apply(lambda x:get_duration(x)[1])
          data.drop(["Duration"], axis = 1, inplace = True)
```

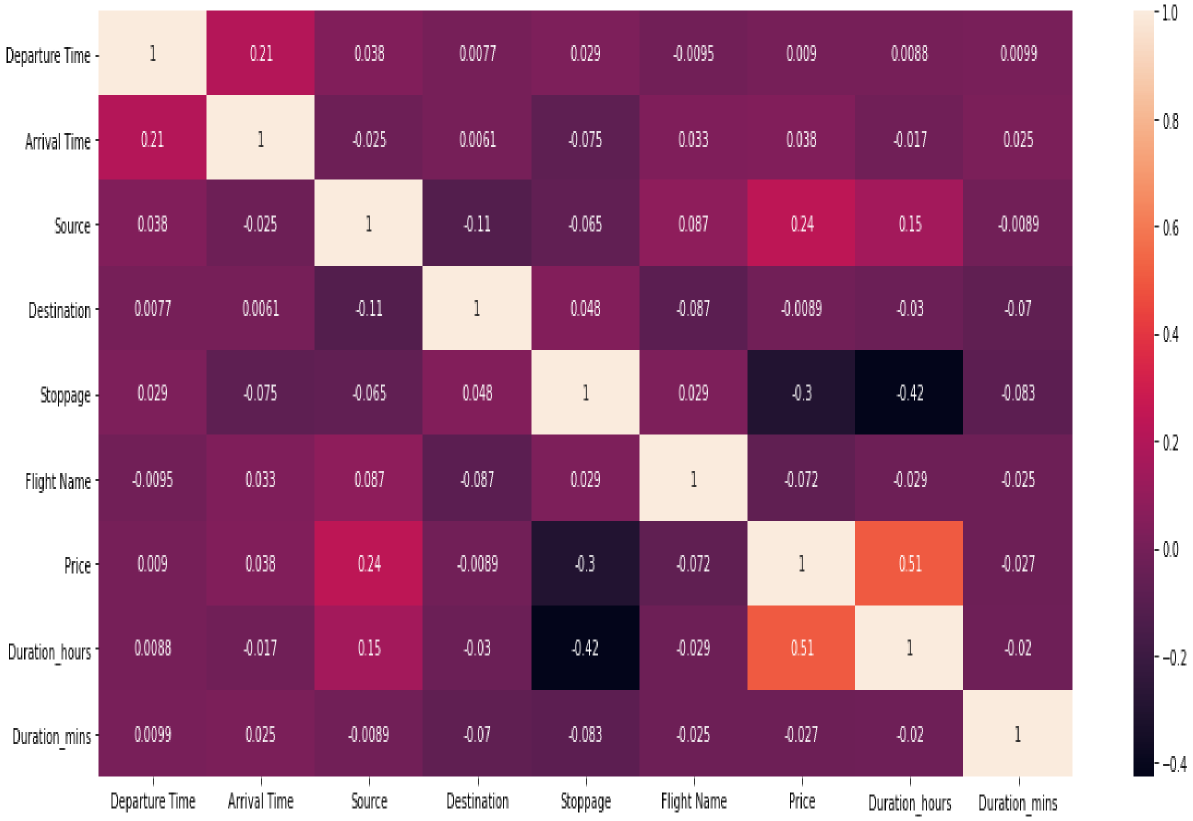
```
In [182]: data
```

Out[182]:

| | Departure Time | Arrival Time | Source | Destination | Stoppage | Flight Name | Price | Duration_hours | Duration_mins |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 16:05 | 20:40 | Delhi | Bangalore | 1 Stop | AirAsia | 7423 | 4 | 35 |
| 1 | 09:25 | 14:40 | Delhi | Bangalore | 1 Stop | AirAsia | 7423 | 5 | 15 |
| 2 | 15:25 | 20:40 | Delhi | Bangalore | 1 Stop | AirAsia | 7423 | 5 | 15 |
| 3 | 15:40 | 21:30 | Delhi | Bangalore | 1 Stop | AirAsia | 7423 | 5 | 50 |
| 4 | 05:00 | 11:35 | Delhi | Bangalore | 1 Stop | AirAsia | 7423 | 6 | 35 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 59 | 17:05 | 17:05 | Delhi | Ahmedabad | 2+ stop | Air India | 17924 | 10 | 45 |

- Converting the data object type into binary encoding using Label Encoder()

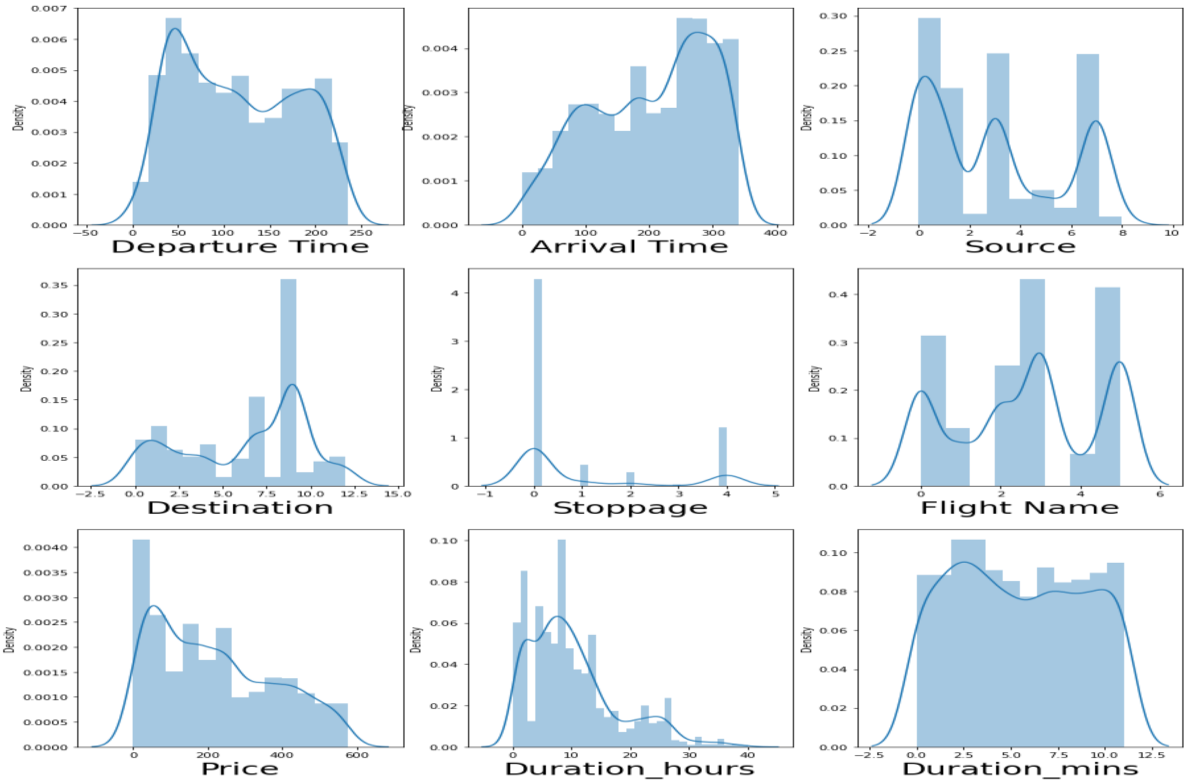
# Visualization:



From the above heat map, we can conclude the relationship between the feature columns with the target column such as “Price”.

As we can see that the highest relationship with the “price” is the “duration of the hour” that takes to travel from one place to another.

The label “price” also had good relationship with the “source” from the journey start.



As there is no outlier in the dataset so we will move to our step.

## Splitting the data into “x” and “y” column:

- We will be using Z score to remove the skewness in the dataset.
- Then we divide the column into “x” as the feature column and “y” as

the label column or the vector column.

- The label column or the vector in this dataset is the “Price” column.

## Building the Model:

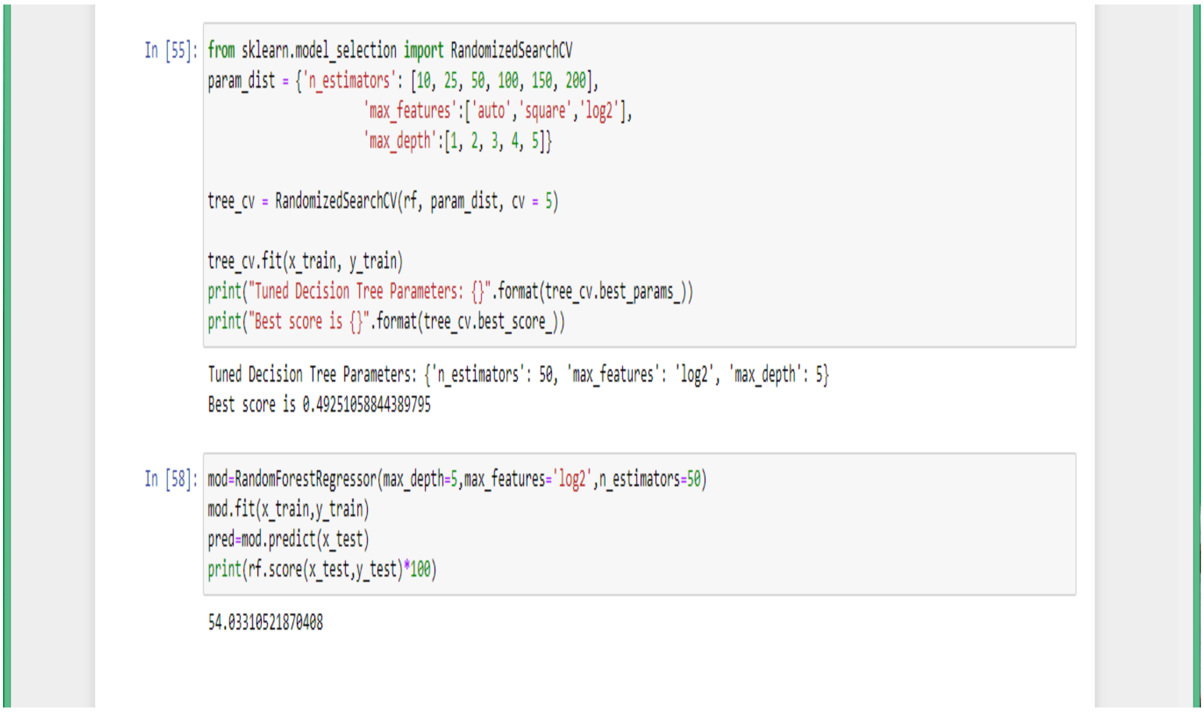
- We will split the data into training and for testing purpose.
- We will divide the data into train as the 0.8 i.e 80%  is the training dataset and testing purpose as   0.2 means 20%   for the testing data.
- Regression predictive modeling is the task of approximating a mapping function (f) from input variables (X) to a continuous output variable (y).
- A continuous output variable is a real-value, such as an integer or floating point

value. These are often quantities, such as amounts and sizes

- Some algorithms have the word "regression" in their name, such as linear regression and logistic regression, which can make things confusing because linear regression is a regression algorithm whereas logistic regression is a classification algorithm.

## Hyper-Parameter Tuning:

Hyper-parameter tuning relies more on experimental results than theory, and thus the best method to determine the optimal settings is to try many different combinations evaluate the performance of each model. However, evaluating each model only on the training set can lead to one of the most fundamental problems in machine learning: overfitting or underfitting. Hyper-parameter tuning is the process of tuning the parameters present as the tuples while we build machine learning models. These parameters are defined by us which can be manipulated according to programmer wish. Machine learning algorithms never learn these parameters. These are tuned so that we could get good performance by the model. Hyper-parameter tuning aims to find such parameters where the performance of the model is highest or where the model performance is best and the error rate is least. We define the hyper-parameter as shown below for the random forest regressor model. These parameters are tuned randomly and results are checked.



```
In [55]: from sklearn.model_selection import RandomizedSearchCV
         param_dist = {'n_estimators': [10, 25, 50, 100, 150, 200],
                       'max_features':['auto','square','log2'],
                       'max_depth':[1, 2, 3, 4, 5]}

         tree_cv = RandomizedSearchCV(rf, param_dist, cv = 5)

         tree_cv.fit(x_train, y_train)
         print("Tuned Decision Tree Parameters: {}".format(tree_cv.best_params_))
         print("Best score is {}".format(tree_cv.best_score_))

         Tuned Decision Tree Parameters: {'n_estimators': 50, 'max_features': 'log2', 'max_depth': 5}
         Best score is 0.49251058844389795

In [58]: mod=RandomForestRegressor(max_depth=5,max_features='log2',n_estimators=50)
         mod.fit(x_train,y_train)
         pred=mod.predict(x_test)
         print(rf.score(x_test,y_test)*100)

         54.03310521870408
```

# Conclusion:

- As we have achieved the model accuracy as 49% but when we tunned the model, we got the accuracy as 54%.
- As we know this is very less accuracy due to many other features that are missing such as date of journey or any other additional feature such as meal is included or not
- We have saved the model using joblib and there is always a scope to improve the accuracy.