FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

SHUBHAM MALIK

ACKNOWLEDGMENT

I would like to thank flip robo for providing me opportunity to work on this project. And would also thanks to my SME for helping me out in the completion of this project. Here I have taken help from the two research papers and from net surfing by help of these I got benefit to understand the problem and data we received from flip robo technology, and I have made my effort to solve this problem statement.

INTRODUCTION

- ## Business Problem Framing

As the demand of airline is increasing day by day because every person want to save its time because in any means of medium it take very long time to reach our destination and in some of the area flight plays a very important role in our daily life but there are some features which help us to make choose our flight easily before date and in low cost and we can identify which airline is giving low price

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 to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, you must work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

- ## Conceptual Background of the Domain Problem

In the past few years its seen that the demand related to flight have increased exponentially. Because due to covid all people want safety and train for every where is still not started and people are moving towards flight so one of our customer want to do business in this field so that we can identify how the price got increase as the date of flight come near and what is the difference between the charges of different airlines which among them is expensive.

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 to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, you must work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights.

- ## Review of Literature

Any individual who has booked a flight ticket previously knows how dynamically costs change. Aircraft uses advanced strategies called Revenue Management to execute a distinctive valuing strategy . The least expensive accessible ticket changes over a period the cost of a ticket might be high or low. This valuing method naturally modifies the toll as per the time like morning, afternoon, or night. Cost may likewise change with the seasons like winter, summer, and celebration seasons. The extreme goal of the carrier is to build its income yet on the opposite side purchaser is searching at the least expensive cost. Purchasers generally endeavor to purchase the ticket in advance to the takeoff day. Since they trust that airfare will be most likely high when the date of buying a ticket is closer to the takeoff date, yet it is not generally true. Purchaser may finish up with the paying more than they ought to for a similar seat.

A report says India's affable aeronautics industry is on a high- development movement. India is the third-biggest avionics showcase in 2020 and the biggest by 2030. Indian air traffic is normal to cross the quantity of 100 million travelers by 2017, whereas there were just 81 million passengers in 2015. Agreeing to Google, the expression Cheap Air Tickets is most sought in India. At the point when the white-collar class of India is presented to air travel, buyers searching at modest costs. The rate of flight tickets at the least cost is continuously expanding.

- ## Motivation for the Problem Undertaken
- This project was provided by FlipRobo as a part of my internship program the main objective behind solving this real time problem is to put forward my skill to solve these problem. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of flight price prediction model so that we can identify which airline is giving low price tickets and how it changes according to time.

Analytical Problem Framing

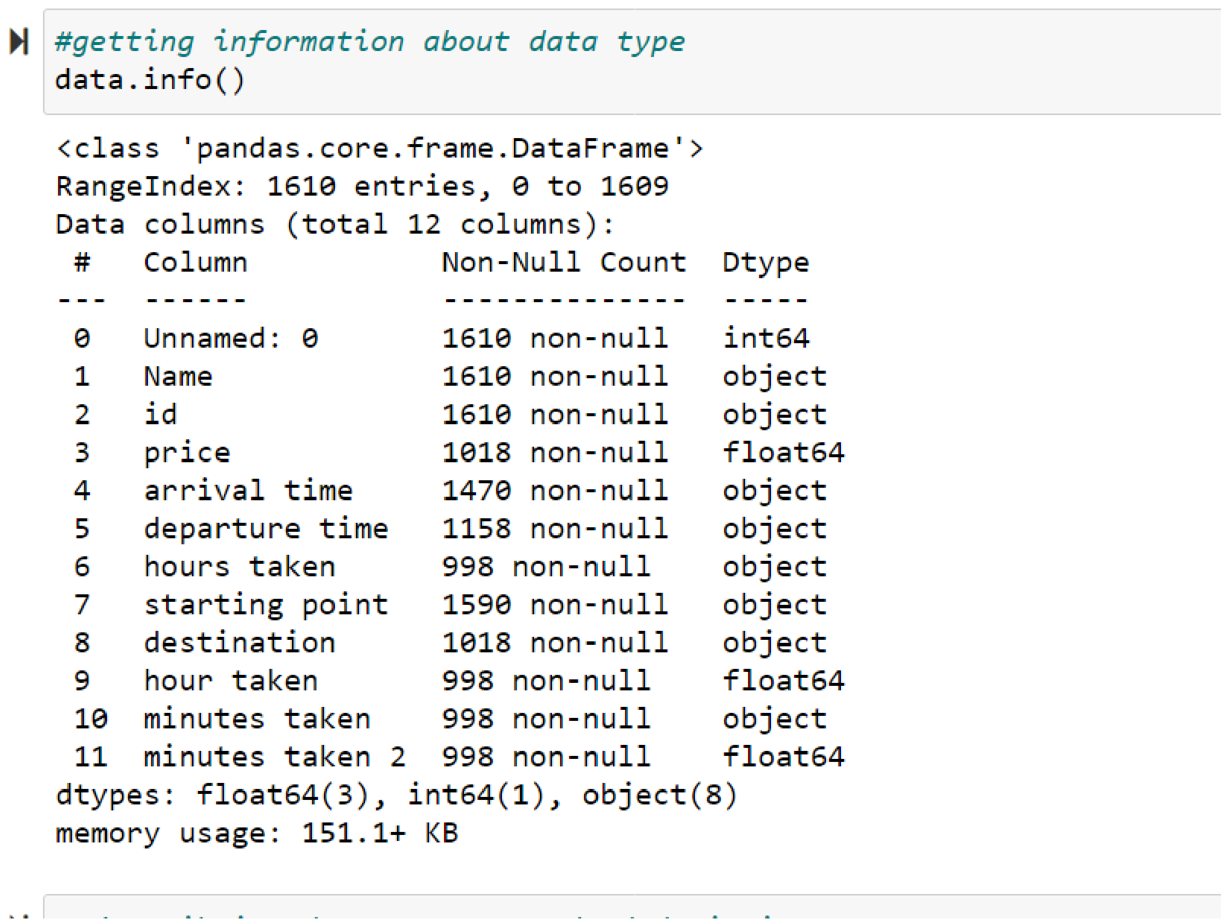
- Mathematical/ Analytical Modeling of the Problem

We have calculated the mean, standard deviation with the help of the describe function. The describe function applies basic statistical computations on the dataset like extreme values, count of data points standard deviation etc. we get details of all the columns by the help of this we can find the outlier present in the data by seeing the mean and 50% value of the given values. We can fill null values in the data set so that they do not make any problem in building the model. With the help of this we get proper justification of the data set we have on which we are working. Here we are dealing with one main text columns which held some importance of the data and others shows the multiple types of behavior inferred from the text. I prefer to select on focus more on the words which has great value of importance in the context. This converts the important words proper vectors with some weights.

- Data Sources and their formats

After loading the training dataset into Jupiter Notebook using Pandas and there are eight columns named as: "id, comment text, "malignant, highly malignant, rude, threat, abuse, loathe". There are 12columns in the dataset which we have collected using web scraping the different site provided:
The description of each of the column is given below:

Unnamed: 0
Name
id
price
arrival time
departure time
hours taken
starting point
destination
hour taken
minutes taken
minutes taken 2



```
#getting information about data type
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1610 entries, 0 to 1609
Data columns (total 12 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   Unnamed: 0        1610 non-null   int64  
 1   Name              1610 non-null   object 
 2   id                1610 non-null   object 
 3   price             1018 non-null   float64
 4   arrival time      1470 non-null   object 
 5   departure time    1158 non-null   object 
 6   hours taken       998 non-null    object 
 7   starting point    1590 non-null   object 
 8   destination       1018 non-null   object 
 9   hour taken        998 non-null    float64
 10  minutes taken     998 non-null    object 
 11  minutes taken 2   998 non-null    float64
dtypes: float64(3), int64(1), object(8)
memory usage: 151.1+ KB
```

a) Data Preprocessing Done

Data pre-processing means that making data correct by removing the unwanted data which help us to achieve highly accurate results. If the quality of data is good, then output of the result is also good. Hence if quality increase then the result of the model automatically increases. Some of the factors which can affect the data are :

i Incomplete

ii. Noisy

iii. Inconsistent data etc.

a) Incomplete data – It can occur due to many reasons. Due to some misunderstanding the data cannot be correct. And other way the machine may not work proper due to some error.

b) Noisy data – noisy data can be described as a faulty data which contain some error in it. It may be by human or by machine defects other way can be transferring of data.

c) Inconsistent data – inconsistent data can be defined as we take example of the bank; we made a transaction the amount of money is deducted from our account but not added to the other account that means there is some error in the database or the dataset.

There are many stages involved in data preprocessing:

1. Data cleaning
2. Data integration
3. Data transformation
4. Data reduction etc.

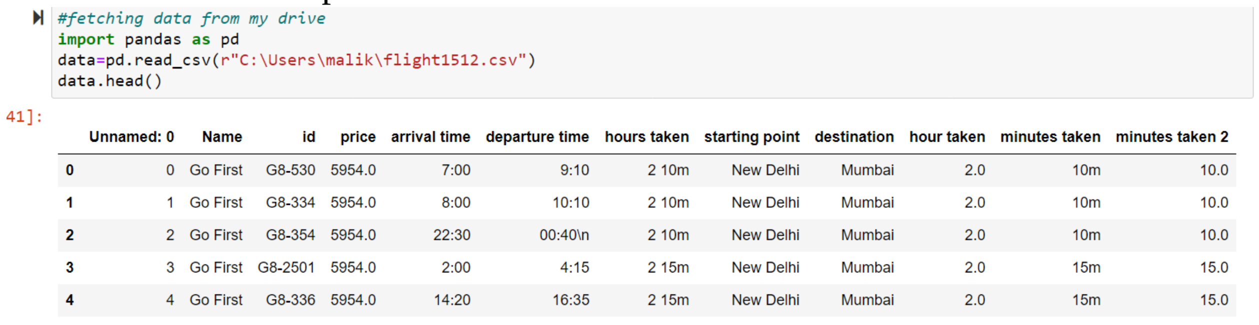
There are many steps which are performed for the data cleaning in this model building. They are as follows

Checking the null values in the data set.

Checking for the outliers we find outlier in 12 columns we applied z-score on these columns to remove the outliers because outlier harm the result of our model which we have built. Then we check for the skewness of the data some of the columns are skewed then we removed skewness with the help of power transform. Then moving towards the model building.

b) Data Inputs- Logic- Output Relationships

After loading all the required libraries, we loaded the data into our jupyter notebook. Then we uploaded our dataset

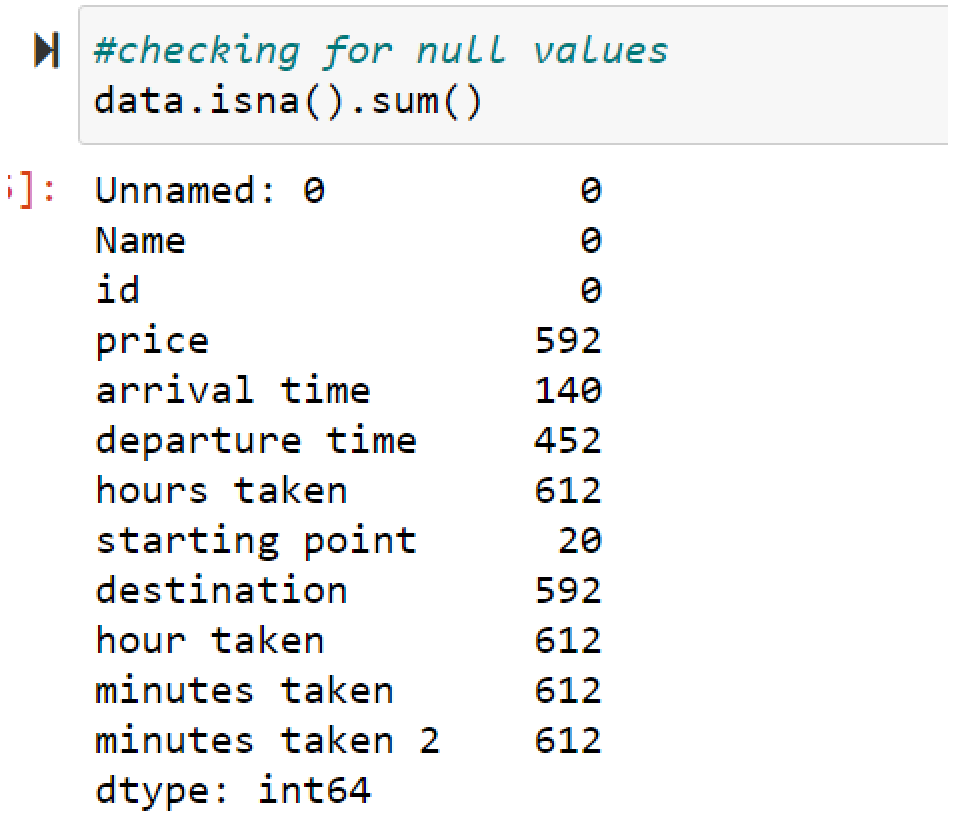


```
#fetching data from my drive
import pandas as pd
data=pd.read_csv(r"C:\Users\malik\flight1512.csv")
data.head()
```

41]:

	Unnamed: 0	Name	id	price	arrival time	departure time	hours taken	starting point	destination	hour taken	minutes taken	minutes taken 2
0	0	Go First	G8-530	5954.0	7:00	9:10	2 10m	New Delhi	Mumbai	2.0	10m	10.0
1	1	Go First	G8-334	5954.0	8:00	10:10	2 10m	New Delhi	Mumbai	2.0	10m	10.0
2	2	Go First	G8-354	5954.0	22:30	00:40\n	2 10m	New Delhi	Mumbai	2.0	10m	10.0
3	3	Go First	G8-2501	5954.0	2:00	4:15	2 15m	New Delhi	Mumbai	2.0	15m	15.0
4	4	Go First	G8-336	5954.0	14:20	16:35	2 15m	New Delhi	Mumbai	2.0	15m	15.0

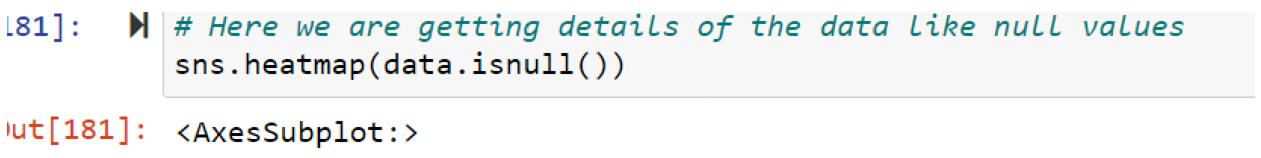
Then we have checked for the missing values in our data set which we have collected from the webscraping



```
#checking for null values
data.isna().sum()
```

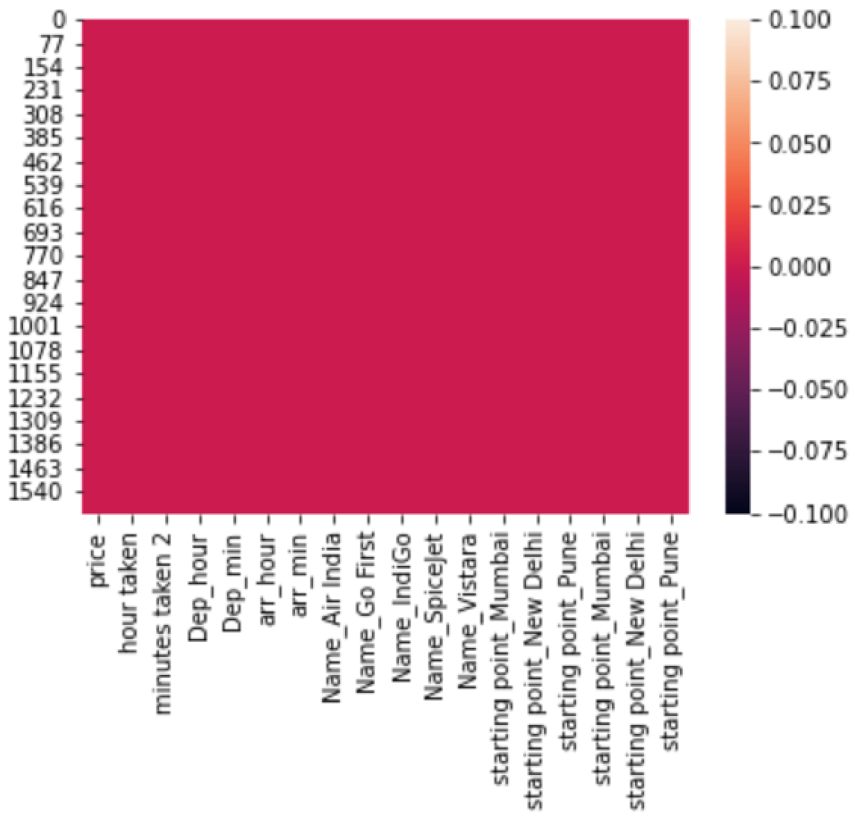
```
]: Unnamed: 0            0
   Name                  0
   id                    0
   price               592
   arrival time        140
   departure time      452
   hours taken         612
   starting point       20
   destination         592
   hour taken          612
   minutes taken       612
   minutes taken 2     612
   dtype: int64
```

Firstly we checked for the null values present in our data set and we find out that there were some null values in our data set and after that we filled those null values with the help of the different techniques like mean, median, mode, forward filling etc. then we proceed towards the model building method.

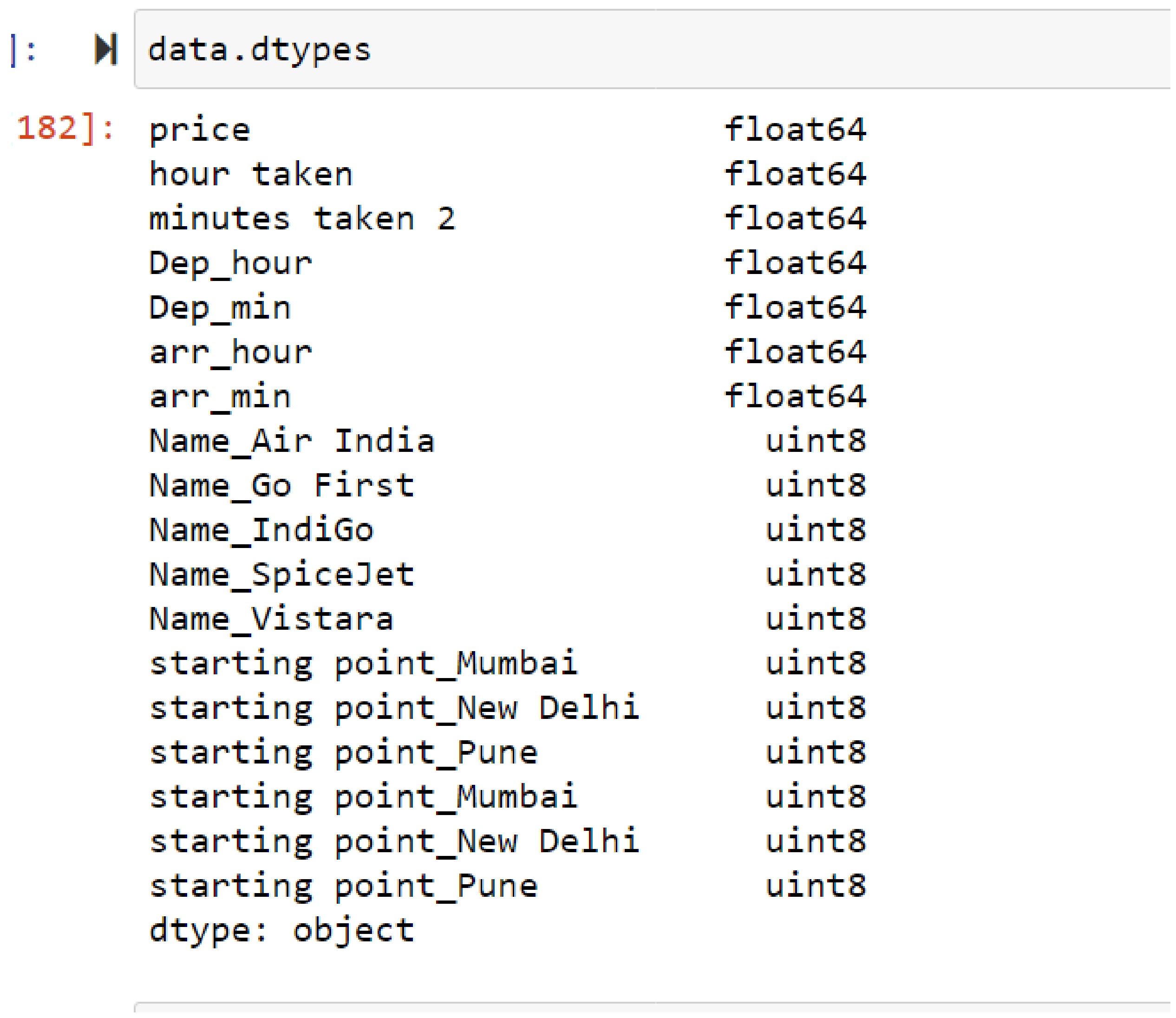


```
181]:  # Here we are getting details of the data like null values
       sns.heatmap(data.isnull())
```

```
ut[181]:  <AxesSubplot:>
```



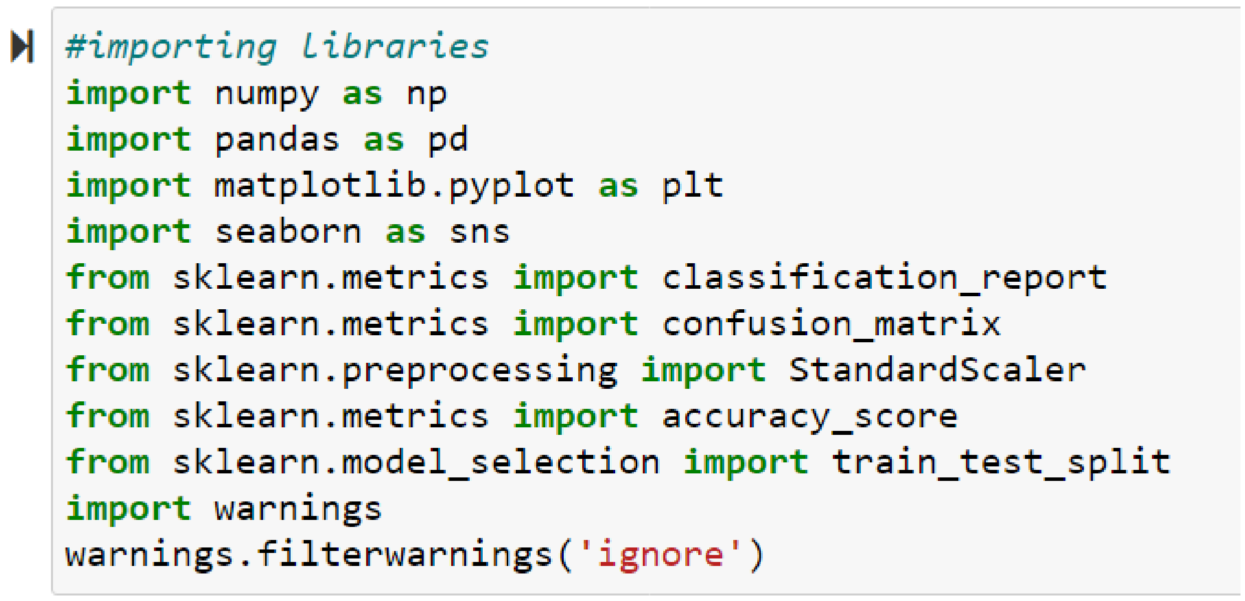
After that we checked for the datatype of the columns because if datatype is object, then our model building will not be proceeded



```
]:    data.dtypes

182]: price                              float64
      hour taken                         float64
      minutes taken 2                    float64
      Dep_hour                           float64
      Dep_min                            float64
      arr_hour                           float64
      arr_min                            float64
      Name_Air India                       uint8
      Name_Go First                        uint8
      Name_IndiGo                          uint8
      Name_SpiceJet                        uint8
      Name_Vistara                         uint8
      starting point_Mumbai                uint8
      starting point_New Delhi             uint8
      starting point_Pune                  uint8
      starting point_Mumbai                uint8
      starting point_New Delhi             uint8
      starting point_Pune                  uint8
      dtype: object
```

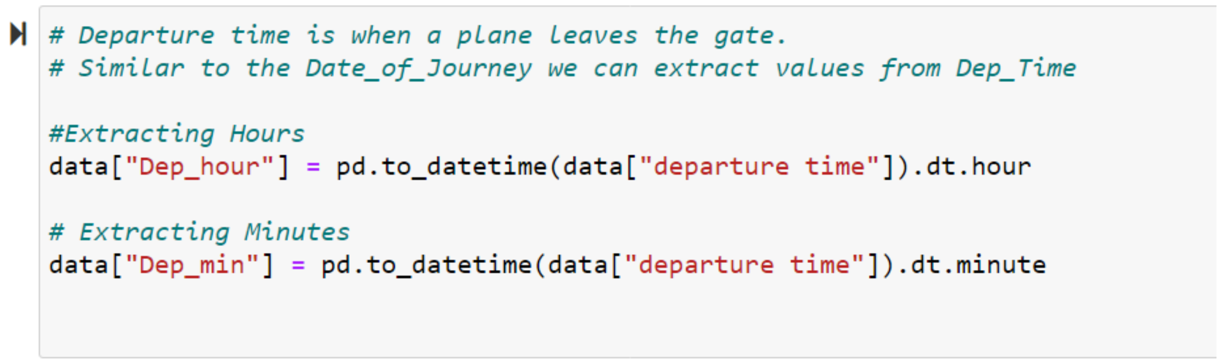
a) Data Preprocessing Done

After loading all the required libraries, we loaded the data into our Jupiter notebook.



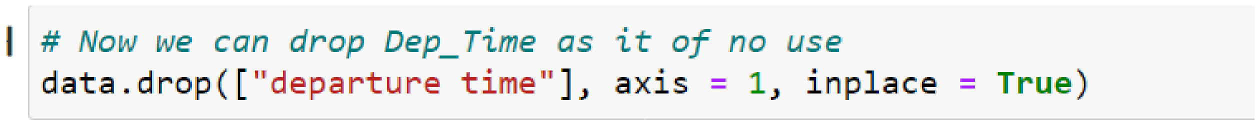
```python
#importing Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')
```

For Data pre-processing we did some data cleaning, where we used datetime library to convert time in hours and minutes columns because it was in different format that was not compatible with our model building so we have converted arrival time and departure time in separate columns as shown below in the figure



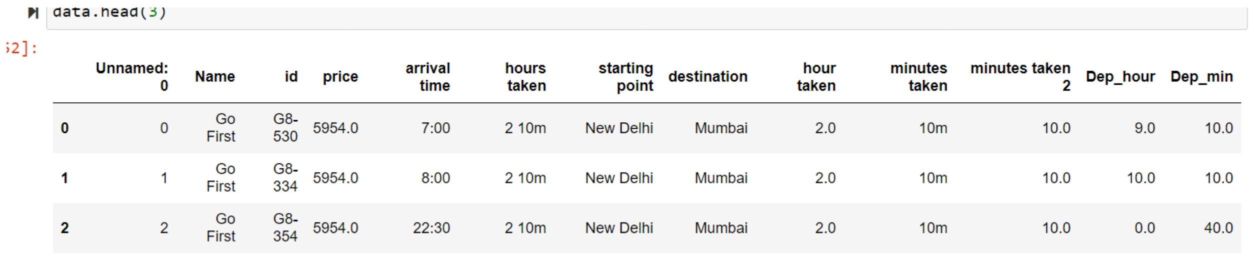
```python
# Departure time is when a plane leaves the gate.
# Similar to the Date_of_Journey we can extract values from Dep_Time

#Extracting Hours
data["Dep_hour"] = pd.to_datetime(data["departure time"]).dt.hour

# Extracting Minutes
data["Dep_min"] = pd.to_datetime(data["departure time"]).dt.minute
```

After converting it to other columns we have dropped the departure time column because that is no use for us now



```python
# Now we can drop Dep_Time as it of no use
data.drop(["departure time"], axis = 1, inplace = True)
```

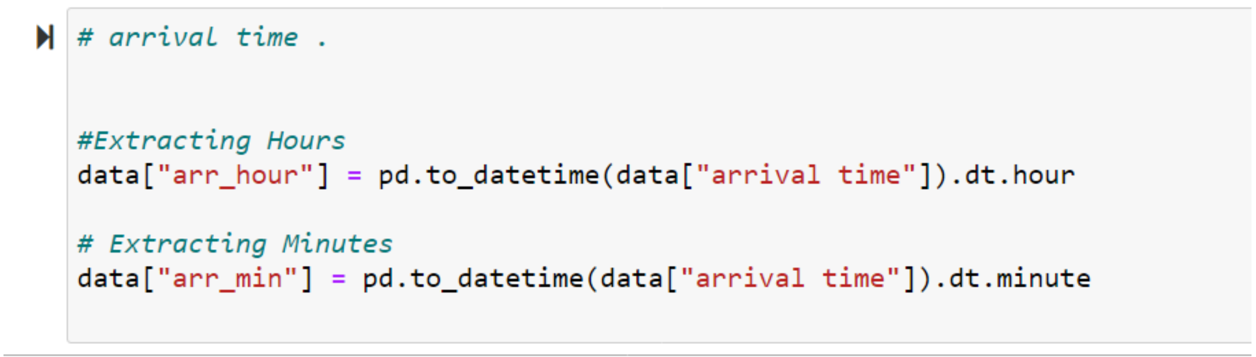
Hence, we got our dataset like this



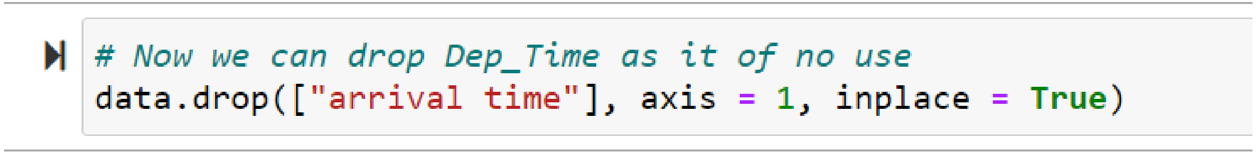
```
data.head(3)
```

	Unnamed: 0	Name	id	price	arrival time	hours taken	starting point	destination	hour taken	minutes taken	minutes taken 2	Dep_hour	Dep_min
0	0	Go First	G8-530	5954.0	7:00	2 10m	New Delhi	Mumbai	2.0	10m	10.0	9.0	10.0
1	1	Go First	G8-334	5954.0	8:00	2 10m	New Delhi	Mumbai	2.0	10m	10.0	10.0	10.0
2	2	Go First	G8-354	5954.0	22:30	2 10m	New Delhi	Mumbai	2.0	10m	10.0	0.0	40.0

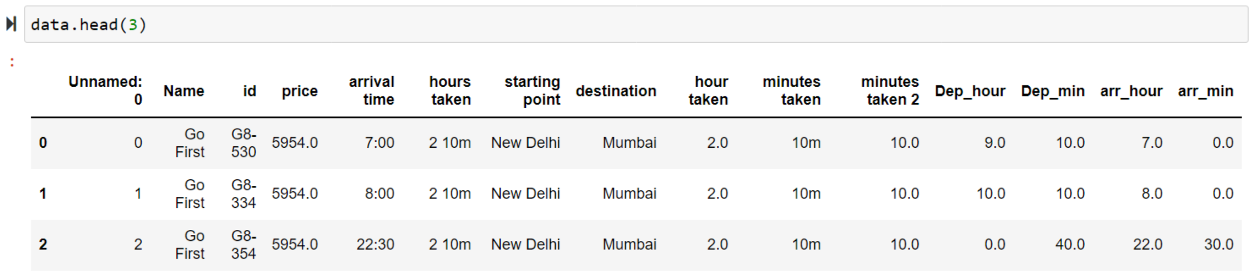
Same we have done for the arrival time as shown in the figure given below



```
# arrival time .


#Extracting Hours
data["arr_hour"] = pd.to_datetime(data["arrival time"]).dt.hour

# Extracting Minutes
data["arr_min"] = pd.to_datetime(data["arrival time"]).dt.minute
```



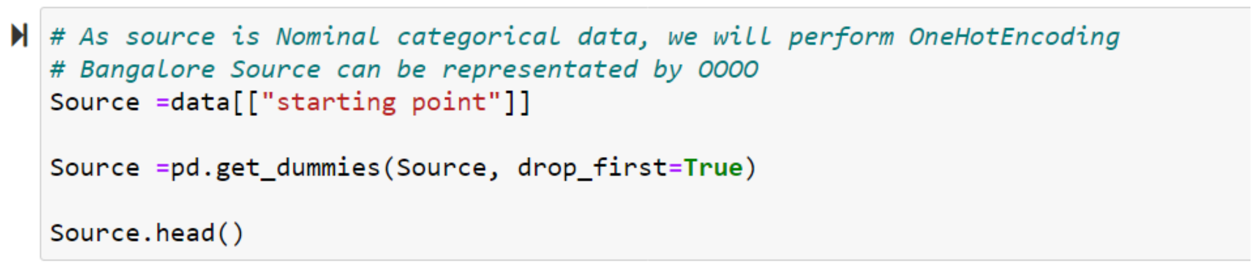
```
# Now we can drop Dep_Time as it of no use
data.drop(["arrival time"], axis = 1, inplace = True)
```



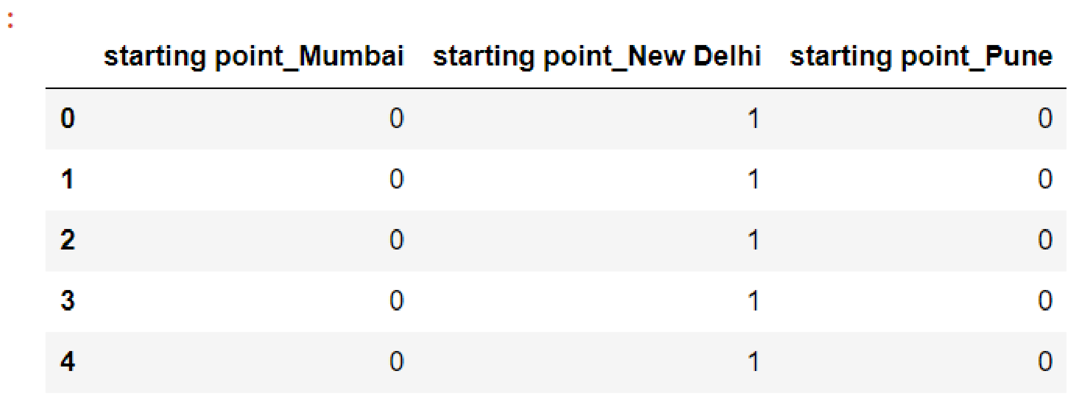
```
data.head(3)
```

	Unnamed: 0	Name	id	price	arrival time	hours taken	starting point	destination	hour taken	minutes taken	minutes taken 2	Dep_hour	Dep_min	arr_hour	arr_min
0	0	Go First	G8-530	5954.0	7:00	2 10m	New Delhi	Mumbai	2.0	10m	10.0	9.0	10.0	7.0	0.0
1	1	Go First	G8-334	5954.0	8:00	2 10m	New Delhi	Mumbai	2.0	10m	10.0	10.0	10.0	8.0	0.0
2	2	Go First	G8-354	5954.0	22:30	2 10m	New Delhi	Mumbai	2.0	10m	10.0	0.0	40.0	22.0	30.0

Then we have the starting point which are in the form of text which are not in the correct format then we created dummies for it like as shown in the figure

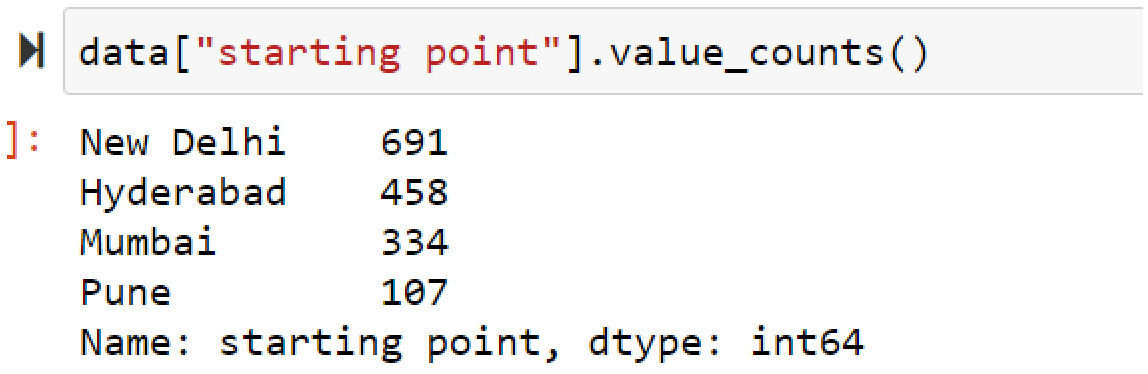


```
# As source is Nominal categorical data, we will perform OneHotEncoding
# Bangalore Source can be representated by 0000
Source =data[["starting point"]]

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

Source.head()
```



	starting point_Mumbai	starting point_New Delhi	starting point_Pune
0	0	1	0
1	0	1	0
2	0	1	0
3	0	1	0
4	0	1	0

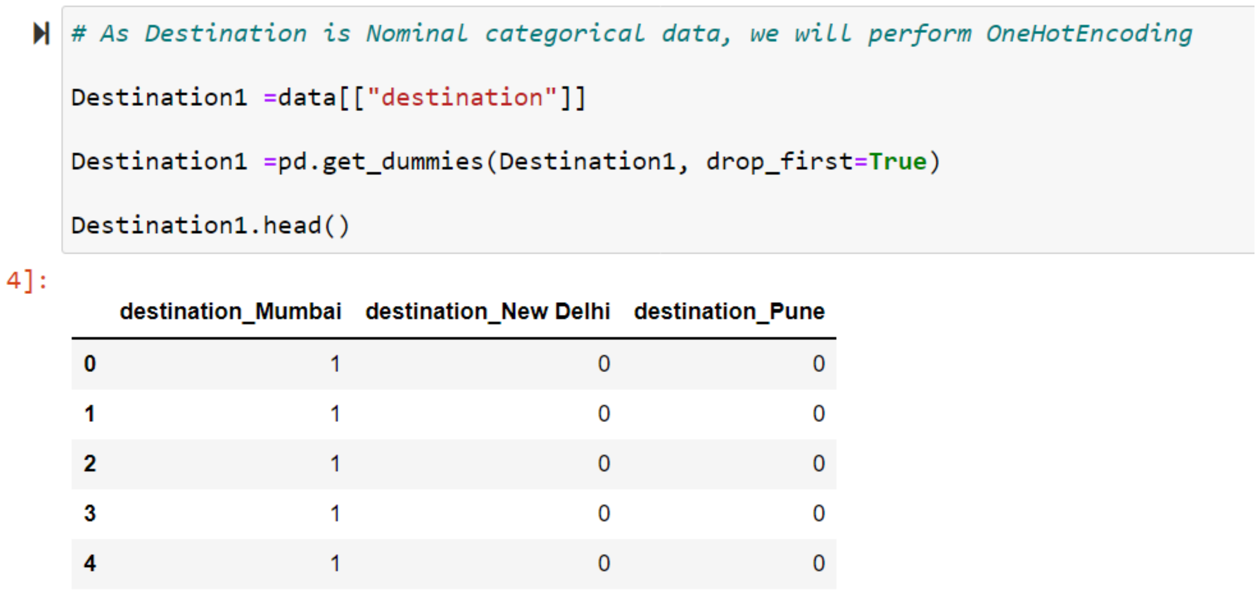
Afte that we have counted the number of stating point from where our flight will be starting into separate columns then we counted the no of flight from each starting point



```
data["starting point"].value_counts()
```

```
New Delhi     691
Hyderabad     458
Mumbai        334
Pune          107
Name: starting point, dtype: int64
```

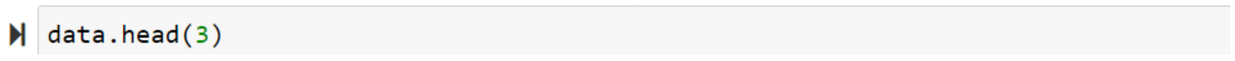
Then we have performed same steps for the destination where the flight is going as shown in the figure



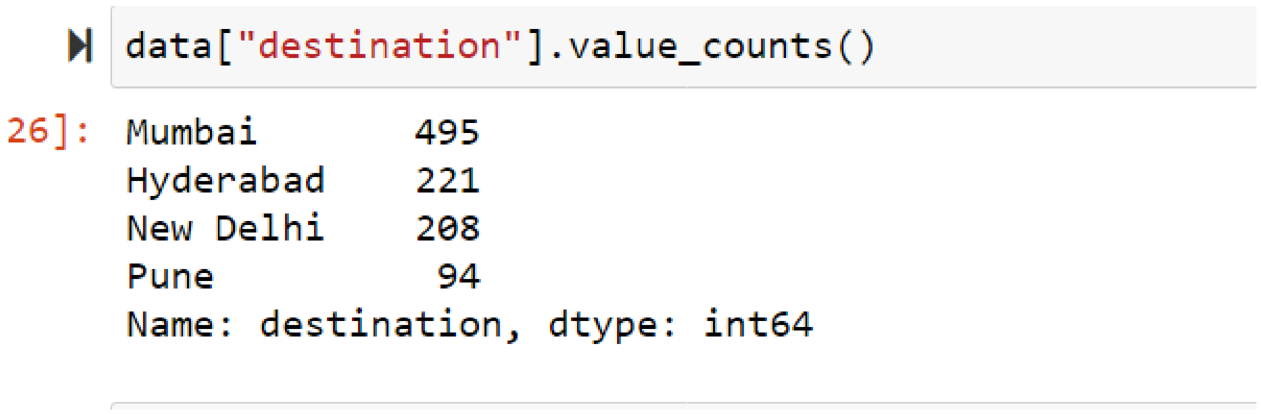
```
# As Destination is Nominal categorical data, we will perform OneHotEncoding

Destination1 =data[["destination"]]

Destination1 =pd.get_dummies(Destination1, drop_first=True)

Destination1.head()
```

4]:

	destination_Mumbai	destination_New Delhi	destination_Pune
0	1	0	0
1	1	0	0
2	1	0	0
3	1	0	0
4	1	0	0



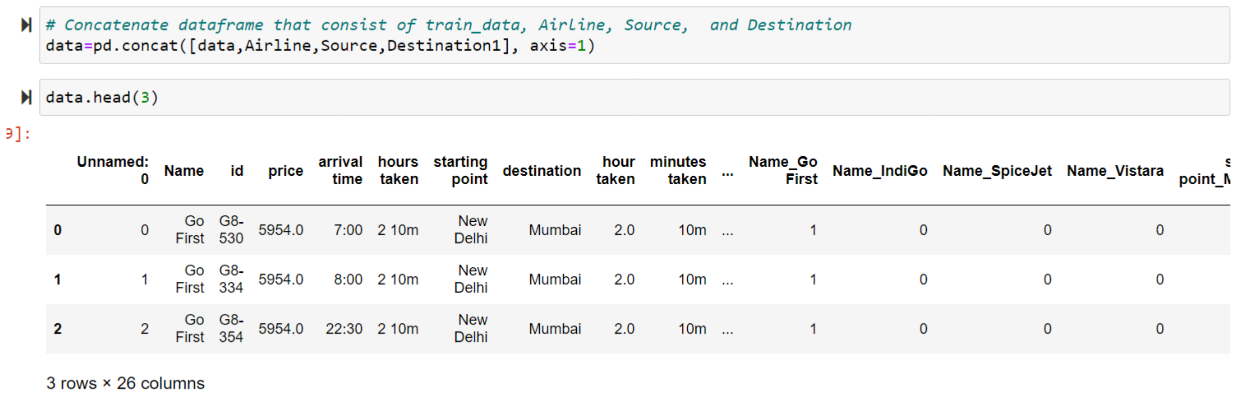
```
data.head(3)
```



```
data["destination"].value_counts()
```

```
26]:  Mumbai       495
      Hyderabad    221
      New Delhi    208
      Pune          94
      Name: destination, dtype: int64
```

These are the number of the destination points
After that we have combined all our data into our data format in which we have to work



```
# Concatenate dataframe that consist of train_data, Airline, Source,  and Destination
data=pd.concat([data,Airline,Source,Destination1], axis=1)
```

```
data.head(3)
```

9]:

	Unnamed: 0	Name	id	price	arrival time	hours taken	starting point	destination	hour taken	minutes taken	...	Name_Go First	Name_IndiGo	Name_SpiceJet	Name_Vistara	s point_N
0	0	Go First	G8-530	5954.0	7:00	2 10m	New Delhi	Mumbai	2.0	10m	...	1	0	0	0	
1	1	Go First	G8-334	5954.0	8:00	2 10m	New Delhi	Mumbai	2.0	10m	...	1	0	0	0	
2	2	Go First	G8-354	5954.0	22:30	2 10m	New Delhi	Mumbai	2.0	10m	...	1	0	0	0	

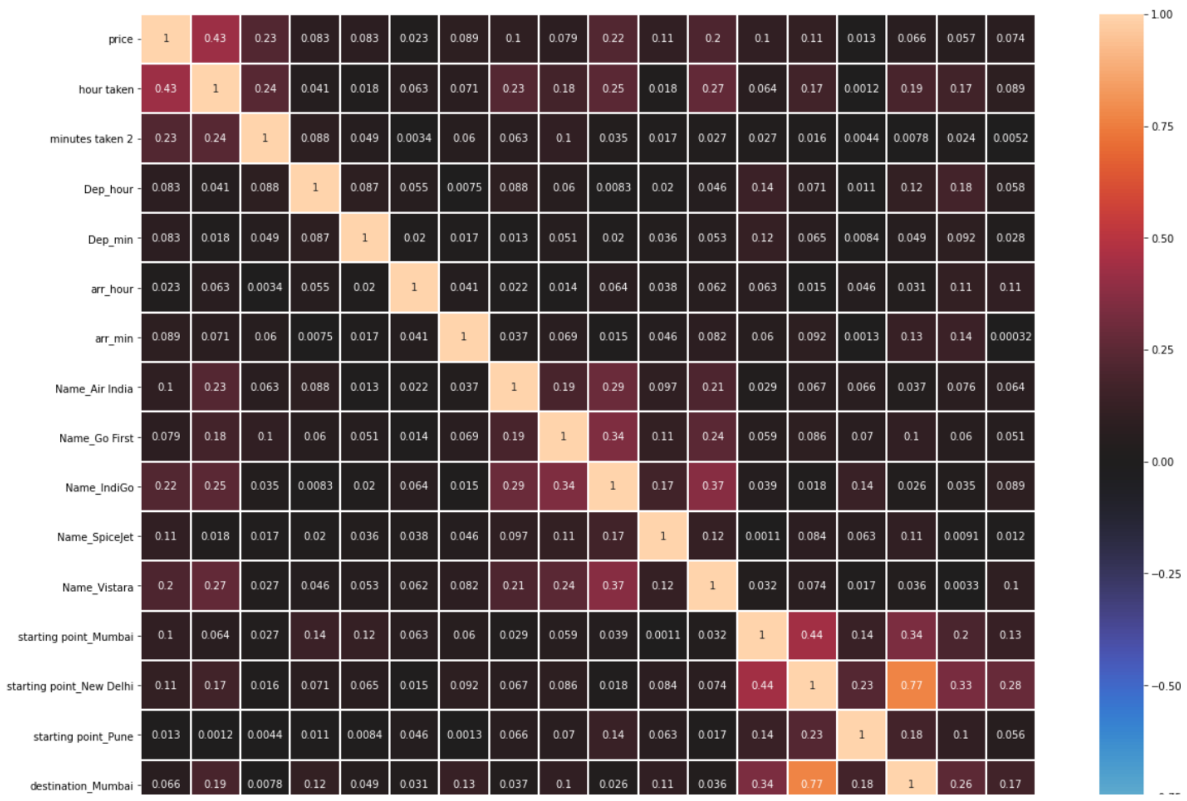
3 rows × 26 columns

Visualization performed

We have plotted heat map for correlation checking

```
#heat map for corelation checking
df_corr =data.corr().abs()
plt.figure(figsize=(22,16))
sns.heatmap(df_corr,  vmin=-1,annot=True,
          square=True,center=0,fmt='.2g',linewidths=1,)
plt.tight_layout
```

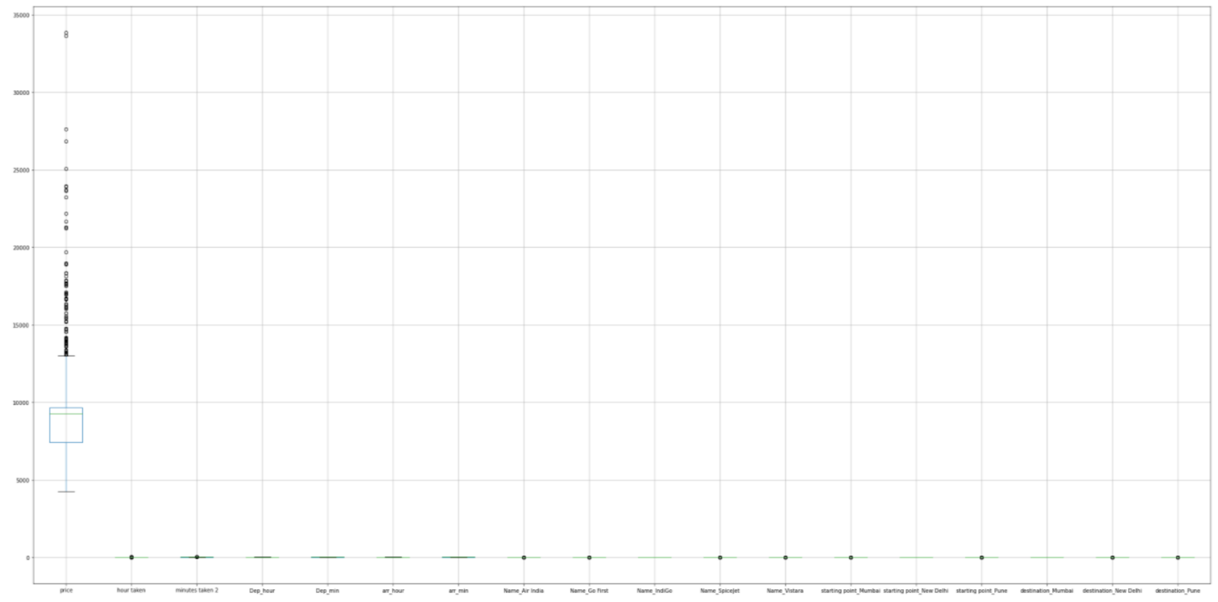
```
]: <function matplotlib.pyplot.tight_layout(*, pad=1.08, h_pad=None, w_pad=None, rect=None)>
```



price	1	0.43	0.23	0.083	0.083	0.023	0.089	0.1	0.079	0.22	0.11	0.2	0.1	0.11	0.013	0.066	0.057	0.074
hour taken	0.43	1	0.24	0.041	0.018	0.063	0.071	0.23	0.18	0.25	0.018	0.27	0.064	0.17	0.0012	0.19	0.17	0.089
minutes taken 2	0.23	0.24	1	0.088	0.049	0.0034	0.06	0.063	0.1	0.035	0.017	0.027	0.027	0.016	0.0044	0.0078	0.024	0.0052
Dep_hour	0.083	0.041	0.088	1	0.087	0.055	0.0075	0.088	0.06	0.0083	0.02	0.046	0.14	0.071	0.011	0.12	0.18	0.058
Dep_min	0.083	0.018	0.049	0.087	1	0.02	0.017	0.013	0.051	0.02	0.036	0.053	0.12	0.065	0.0084	0.049	0.092	0.028
arr_hour	0.023	0.063	0.0034	0.055	0.02	1	0.041	0.022	0.014	0.064	0.038	0.062	0.063	0.015	0.046	0.031	0.11	0.11
arr_min	0.089	0.071	0.06	0.0075	0.017	0.041	1	0.037	0.069	0.015	0.046	0.082	0.06	0.092	0.0013	0.13	0.14	0.00032
Name_Air India	0.1	0.23	0.063	0.088	0.013	0.022	0.037	1	0.19	0.29	0.097	0.21	0.029	0.067	0.066	0.037	0.076	0.064
Name_Go First	0.079	0.18	0.1	0.06	0.051	0.014	0.069	0.19	1	0.34	0.11	0.24	0.059	0.086	0.07	0.1	0.06	0.051
Name_IndiGo	0.22	0.25	0.035	0.0083	0.02	0.064	0.015	0.29	0.34	1	0.17	0.37	0.039	0.018	0.14	0.026	0.035	0.089
Name_SpiceJet	0.11	0.018	0.017	0.02	0.036	0.038	0.046	0.097	0.11	0.17	1	0.12	0.0011	0.084	0.063	0.11	0.0091	0.012
Name_Vistara	0.2	0.27	0.027	0.046	0.053	0.062	0.082	0.21	0.24	0.37	0.12	1	0.032	0.074	0.017	0.036	0.0033	0.1
starting point_Mumbai	0.1	0.064	0.027	0.14	0.12	0.063	0.06	0.029	0.059	0.039	0.0011	0.032	1	0.44	0.14	0.34	0.2	0.13
starting point_New Delhi	0.11	0.17	0.016	0.071	0.065	0.015	0.092	0.067	0.086	0.018	0.084	0.074	0.44	1	0.23	0.77	0.33	0.28
starting point_Pune	0.013	0.0012	0.0044	0.011	0.0084	0.046	0.0013	0.066	0.07	0.14	0.063	0.017	0.14	0.23	1	0.18	0.1	0.056
destination_Mumbai	0.066	0.19	0.0078	0.12	0.049	0.031	0.13	0.037	0.1	0.026	0.11	0.036	0.34	0.77	0.18	1	0.26	0.17

Checking for outliers

```
# plotting boxplot to identify outliers
data.boxplot(figsize=[40,40])
plt.subplots_adjust(bottom=0.5)
plt.show()
```

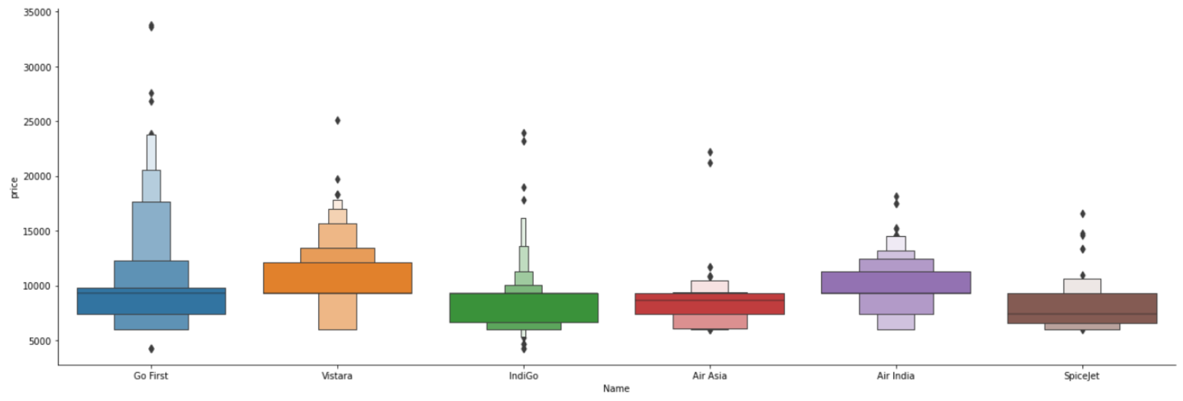


Then we removed outliers using quantile method

```
#removing data of outliers
q=data['price'].quantile(0.60)
data_cleaned =data[data['price']<q]
```

With the help of the visualization we identified that go first airline has the highest price among all the other airlines in our data apart from that all other airlines have similar price index very little changes in the price as shown in the given graph we can depict by seeing it.

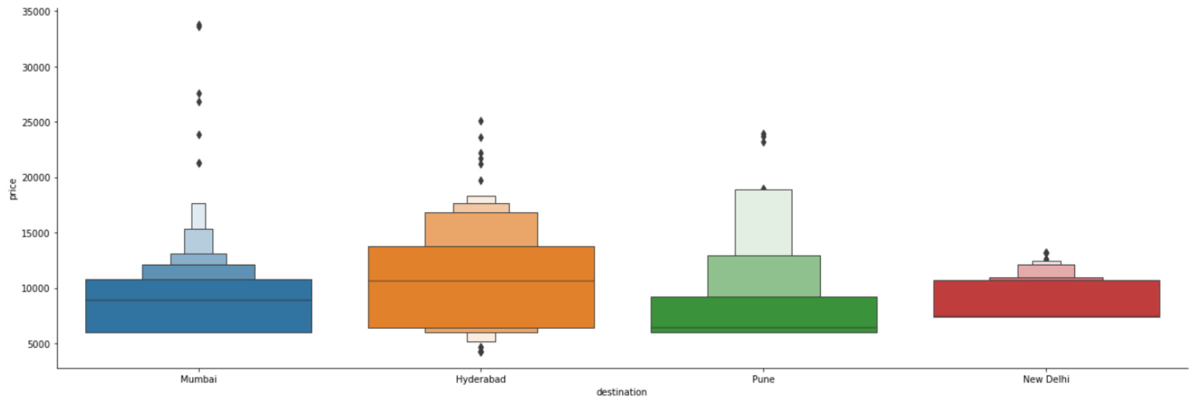
```
# From grap we can see that go first Airways Business have the highest Price
# Apart from the first Airlines almost all are having similar median

# Airline vs price
sns.catplot(y= "price", x = "Name", data = data.sort_values("price", ascending = False), kind="boxen", height = 6, aspect = 3
plt.show()
```



Here we done comparison between the price and the destination where the flight is going

```
# Compare Source and Price
# We can see some outliers in Bangalore while the others place doe not too different

sns.catplot(y = "price", x= "destination", data = data.sort_values("price", ascending = False), kind="boxen", height = 6, asp
```

```
<seaborn.axisgrid.FacetGrid at 0x1c51633ff10>
```



- Hardware and Software Requirements and Tools Used

Here software used is Jupiter notebook in which we build the model in python. This is a built-in platform we must start by importing some of the library which are used in the model building. Some are as follows:

i. Importing NumPy-this is a numerical python in which numeric calculation are done.

ii. Importing pandas- pandas are used to perform different operation from uploading or retrieving the file to make changes in file.

iii. Matplotlib- these are used for the visualization technique to plot the histogram, boxplot etc.
iv. Seaborn – these are used for the visualization technique by the help of which we plot heat map.

There are more libraries like

i. Classification report
ii. Confusion matrix
iii. Standard scalar
iv. Accuracy score

Model/s Development and Evaluation

- For this we have used different method for solving this problem for the analysis purpose we use visualization that is boxplot, seaborn and matplotlib to identify the pattern of the data. We find that data is starting from zero which is good then we applied boxplot for outliers' detection we find that 3 columns are having the outliers. Outliers are not good for the model building. Then with the help of quantile we remove the outliers. By the help of the visualization, we find the features related to price. Then after we find the correlation of the data each column correlation with itself. we find that some of them are highly corelated. After that we started building the model, we split data set in two form label and features using train test split.

-

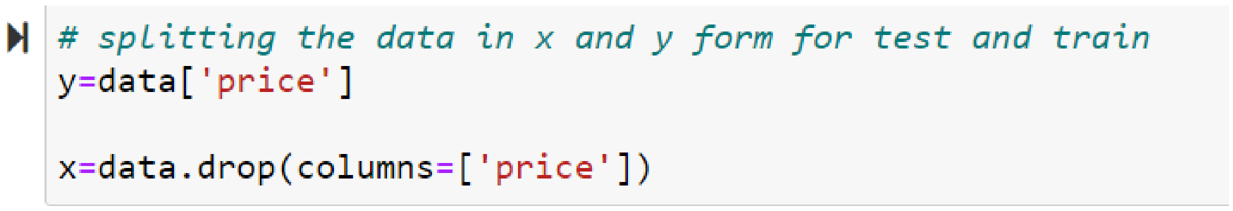
- ## Testing of Identified Approaches (Algorithms)

List of algorithms used are:

I. Linear Regression
II. Decision Tree regressor
III. Random Forest regressor

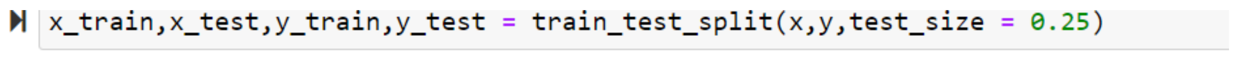
- ## Run and evaluate selected models

Now we need to separate the train dataset into X and y values. We have converted the train data set in the form of array and split it in the form of x and y



```
# splitting the data in x and y form for test and train
y=data['price']

x=data.drop(columns=['price'])
```

Let’s split the master dataset into training and test set with an 75%-25% ratio. it is a function in sklearn model selection for splitting data arrays into two subsets for training data and testing data. With this function, you don’t need to divide the dataset manually. By default, sklearn train_test_split will make random partitions for the two subsets. However, you can also specify a random state for the operation. It gives four outputs x train, x test, y train and y test. The x train and x test contain the training and testing predictor variables while y train and y test contains the training and testing target variable



```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.25)
```

a) Linear regression

Linear regression may be defined as the statistical model that analyzes the linear relationship between a dependent variable with given set of independent variables. Linear relationship between variables means that when the value of one or more independent variables will change (increase or decrease), the value of dependent variable will also change accordingly (increase or decrease).

Mathematically the relationship can be represented with the help of following equation –

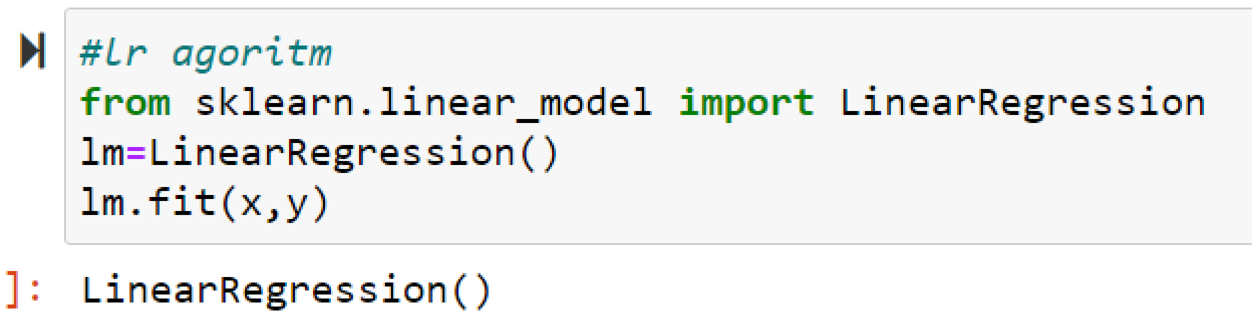
$Y = mX + b$

Here, Y is the dependent variable we are trying to predict

X is the dependent variable we are using to make predictions.

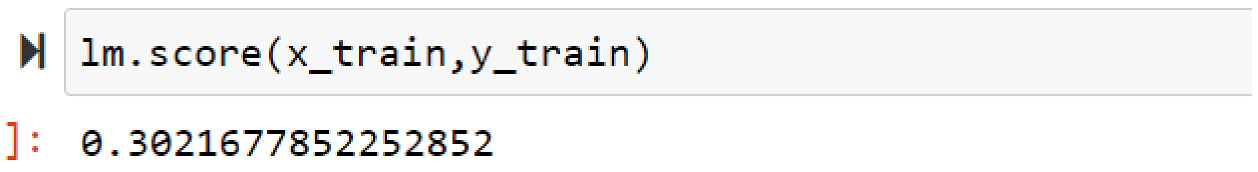
m is the slop of the regression line which represents the effect X has on Y

b is a constant, known as the Y-intercept. If X = 0,Y would be equal to b.



```
#Lr agoritm
from sklearn.linear_model import LinearRegression
lm=LinearRegression()
lm.fit(x,y)
```

```
]: LinearRegression()
```

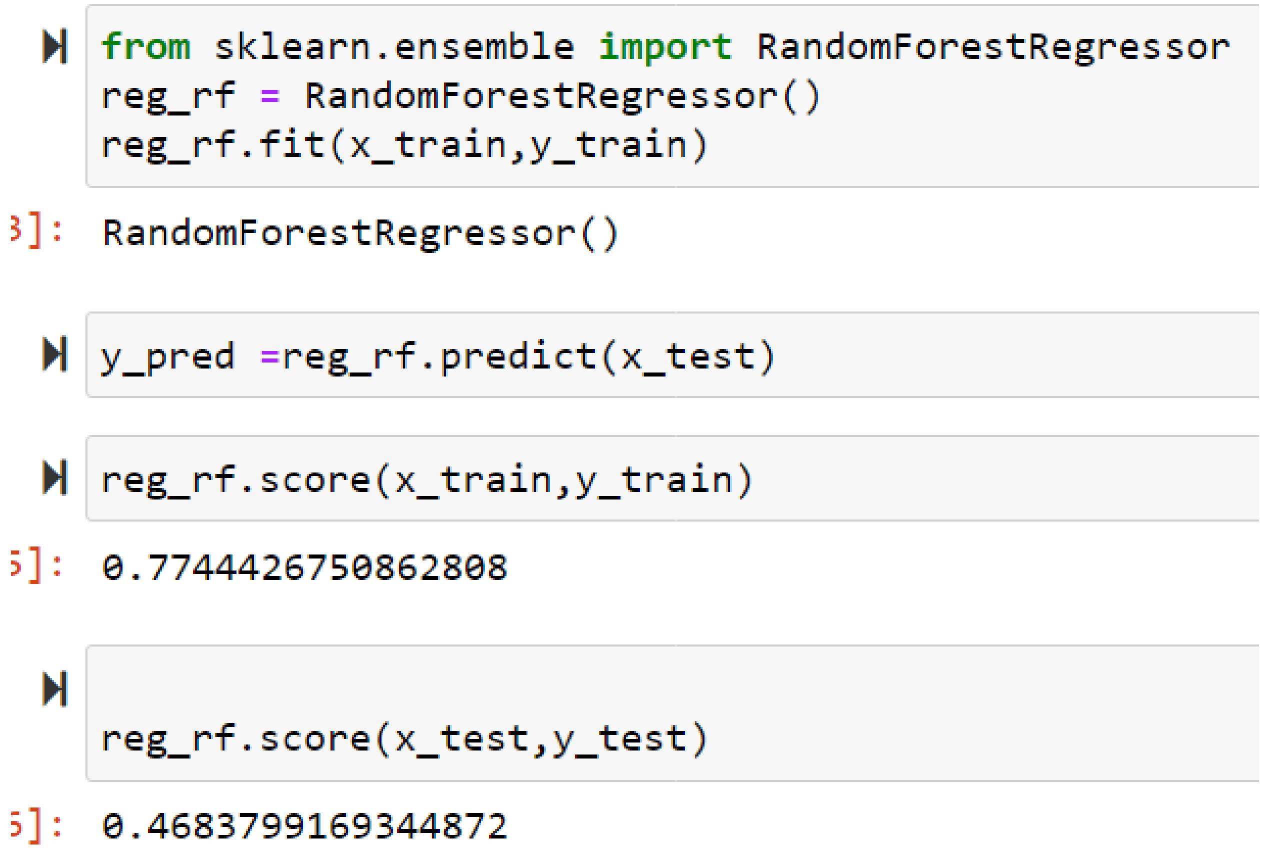


```
lm.score(x_train,y_train)
```

```
]: 0.3021677852252852
```

b) Random forest Regressor

As we know that a forest is made up of trees and more trees means more robust forest. Similarly, a random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.



```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(x_train,y_train)
```

```
3]:  RandomForestRegressor()
```

```
y_pred =reg_rf.predict(x_test)
```

```
reg_rf.score(x_train,y_train)
```

```
5]:  0.7744426750862808
```

```

reg_rf.score(x_test,y_test)
```

```
5]:  0.4683799169344872
```

c) Decision tree Regressor algorithm
Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. The two main entities of a tree are decision nodes, as parent node where the data is split and leaves or child node, where we get the outcome.

```
#using dtc algoritm
from sklearn.tree import DecisionTreeRegressor
clf=DecisionTreeRegressor()
clf.fit(x_train,y_train)
```

```
]: DecisionTreeRegressor()
```

```
clf.score(x_train,y_train)
```

```
]: 0.8016758909145979
```

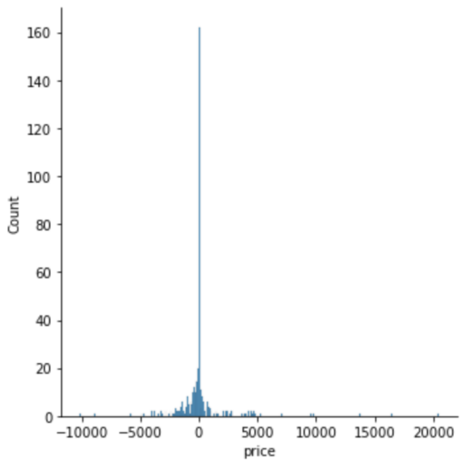
```
y_pred=clf.predict(x_test)
```

```
clf.score(x_test,y_test)
```

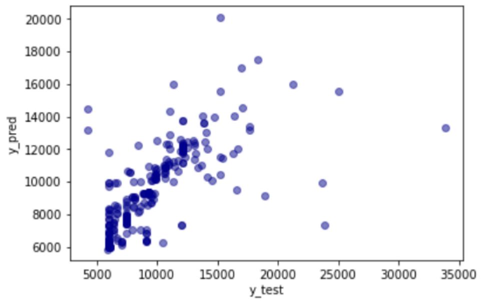
```
]: 0.2571241146246377
```

For prediction we plotted graph

```
sns.displot(y_test-y_pred)
plt.show()
```



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



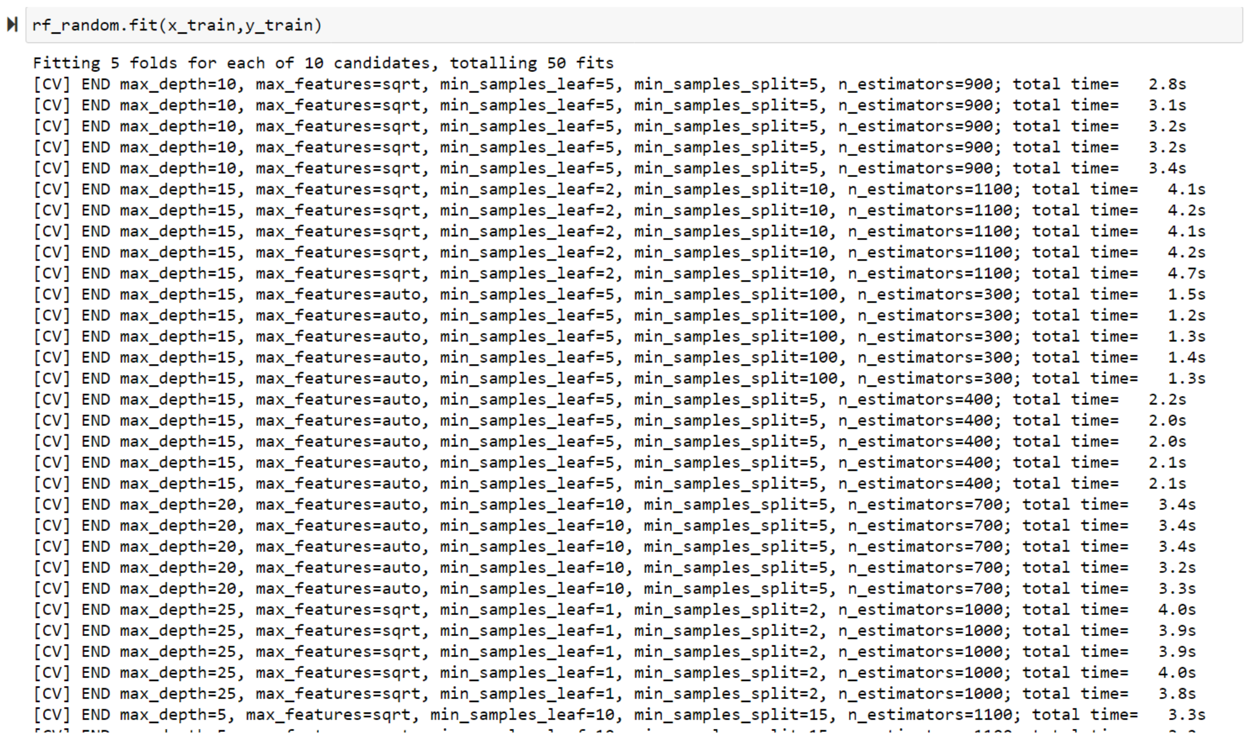
d) Hyperparametric tunning

Model improvement can be done by choosing the different methods of machine learning that we come up with. There are two methods that we have used in this model -first type is the model learning by the help of training and after that predicting the result and the second method is hyperparameters it a random model that cannot be determined by the data we tried to optimize tuning techniques using randomized search CV using lasso model. Hyperparameter tuning might not improve the model every time. For instance, when we tried to tune the model further which is shown below in the figure.

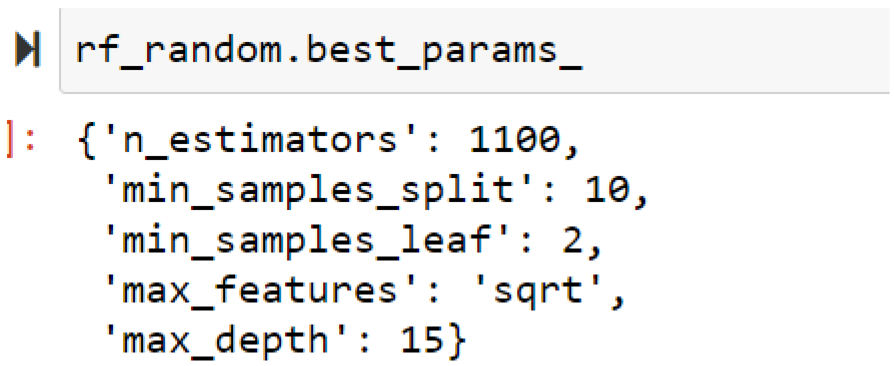
```
#Randomized Search CV

# 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]
```

```
# 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}
```

```
# 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_squared_error', n_iter
```



```
rf_random.fit(x_train,y_train)

Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.2s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.1s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.7s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   1.3s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   2.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   2.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   2.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   2.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   2.1s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   3.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   3.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   3.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   3.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total time=   3.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   4.0s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.9s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.9s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   4.0s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total time=   3.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total time=   3.3s
```

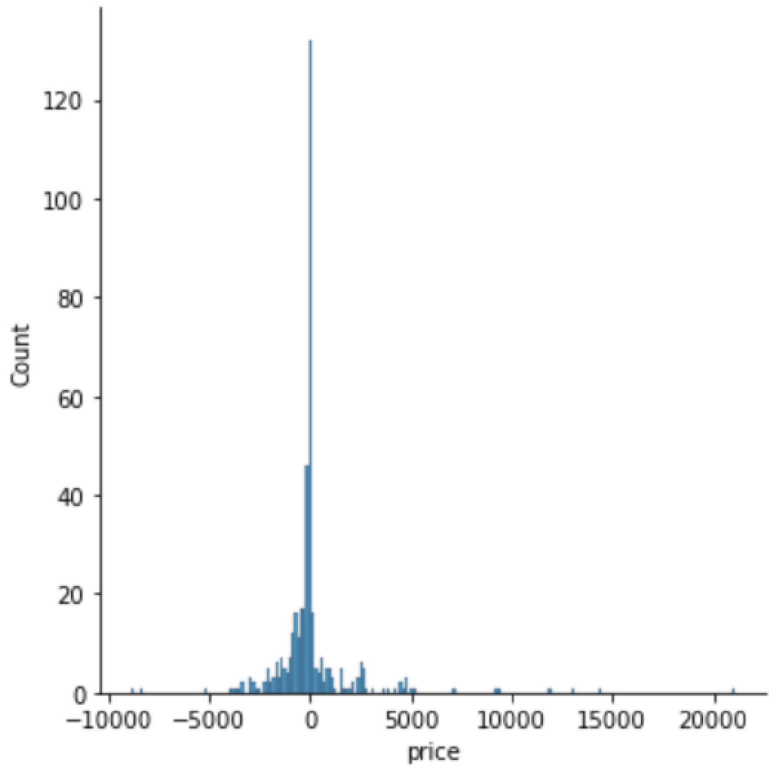


```
rf_random.best_params_

]: {'n_estimators': 1100,
     'min_samples_split': 10,
     'min_samples_leaf': 2,
     'max_features': 'sqrt',
     'max_depth': 15}
```

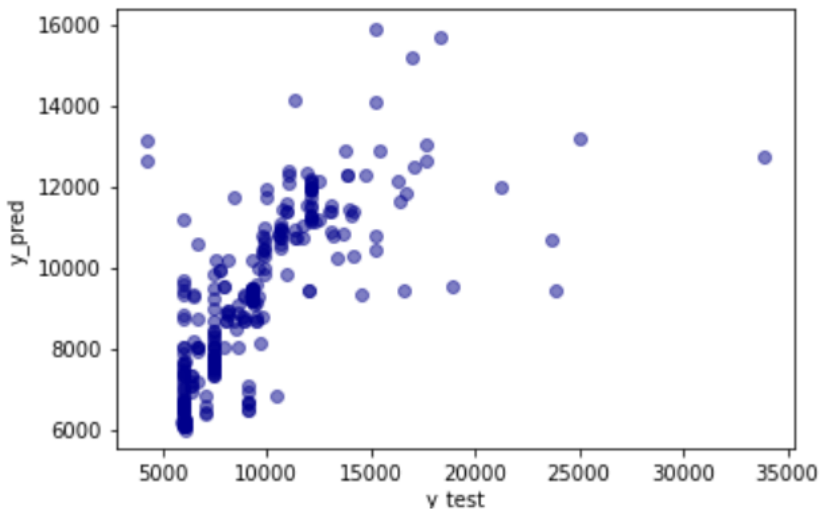
```
prediction = rf_random.predict(x_test)
```

```
plt.figure(figsize =(8,8))
sns.displot(y_test-prediction)
plt.show()
```

```
<Figure size 576x576 with 0 Axes>
```



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



Metrics evaluation

```python
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: 984.4738096490603
MSE: 4978916.473970062
RMSE: 2231.3485774235414
```

Finally we saved

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

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

CONCLUDING REMARKS:

After applying the machine learning algorithms, we concluded that it provides the accuracy of the model is 81% without losing any data. The best accuracy is given by the random forest regressor algorithm in this algorithm it identifies the random forest contain n number of trees according to it check for each iteration and provide the best result to us. The main aim of this project was to predict the price of the flight. As the demand of airline is increasing day by day because every person want to save its time because in any means of medium it take very long time to reach our destination and in some of the area flight plays a very important role in our daily life but there are some features which help us to make choose our flight easily before date and in low cost and we can identify which airline is giving low price. So that why one of our client want prediction about the flight price.