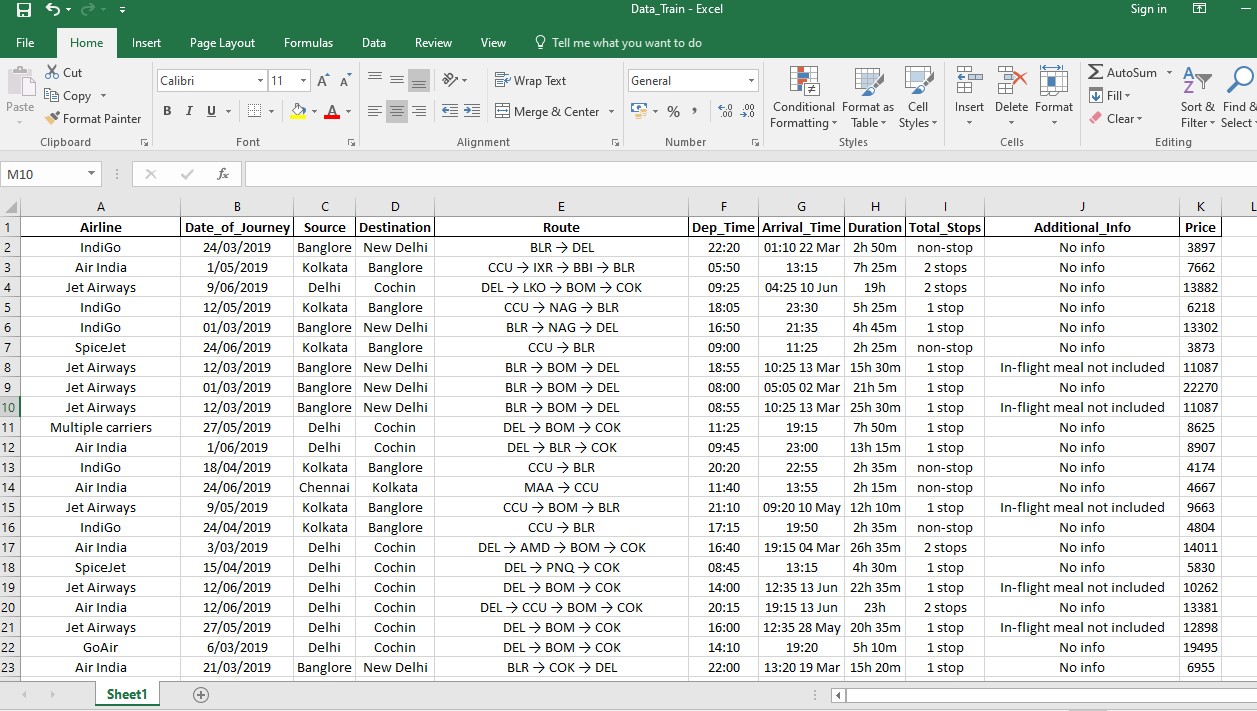
**Flight Price Prediction**

## **Introduction:** This project is all about Flight Price Prediction. 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 like: Mumbai, Bangalore, Chennai, Cochin, New Delhi, Hyderabad, Kolkata. In this project I will take some inputs from users like: Departure Date & Time, Arrival Date & Time, Source, Destination, Stoppage and Which Airline you want to travel? Etc. This project is a Supervised Machine Learning project. In this project I have used **Multiple Linear Regression** because here is Price is a dependent feature and others are independent features like: Departure Date & Time, Arrival Date & Time, Source, Destination, Stoppage. In this project I have train my model using **RandomForestRegressor**, this ML algorithm predict better accuracy. Using RandomForestRegressor model I got **95% Accuracy** on my **train dataset** and **79% Accuracy** on my **test dataset**.

**Datasets:** For this project I used **Data\_Train.xlsx** Flight Price Prediction from kaggle.com. I have uploaded my dataset on GitHub.

[**https://github.com/SMIIT-Projects/Flight-Price-Prediction**](https://github.com/SMIIT-Projects/Flight-Price-Prediction)

****

**Python Coding**

**The Code is written in Python 3.8.5.**

**Libraries used:**

**certifi==2020.12.5, click==7.1.2, cycler==0.10.0, Flask==1.1.2, Flask-Cors==3.0.10, gunicorn==20.0.4, itsdangerous==1.1.0, Jinja2==2.11.3, joblib==1.0.1, kiwisolver==1.3.1, MarkupSafe==1.1.1, matplotlib==3.3.4, numpy==1.20.1, pandas==1.2.3, Pillow==8.1.2, pyparsing==2.4.7, python-dateutil==2.8.1, pytz==2021.1, scikit-learn==0.24.1, scipy==1.6.1, seaborn==0.11.1, six==1.15.0, threadpoolctl==2.1.0, Werkzeug==1.0.1, wincertstore==0.2.**

## **Step1: Import all important libraries**

Import all Important libraries in python Jupyter notebook. Then load the dataset form Kaggle.com/above mention GitHub link. After loading the dataset check shape of the data means how many rows & columns in our dataset. Here I have loaded my dataset and got a 10683 rows and 11 columns. After that check the information of all Dependent and Independent variables like how many Integer, float and object datatypes are available in dataset and count these all.

**Code:**

import numpy as np

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import os

import warnings

warnings.filterwarnings('ignore')

from sklearn.ensemble import ExtraTreesRegressor

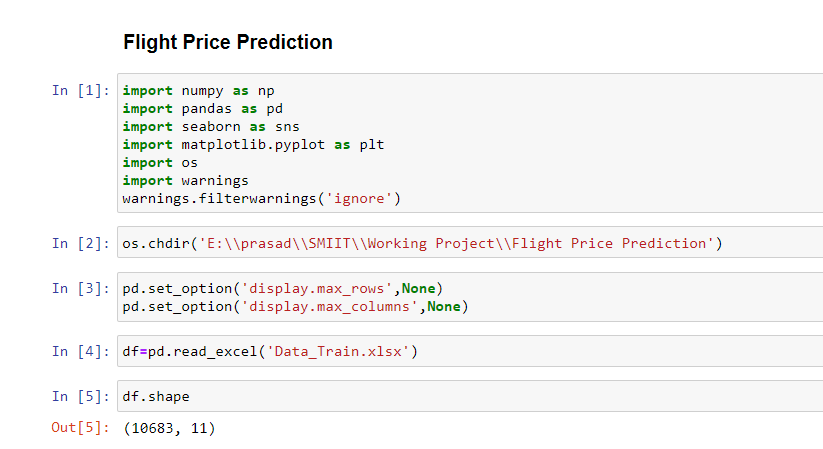
from sklearn.model\_selection import train\_test\_split

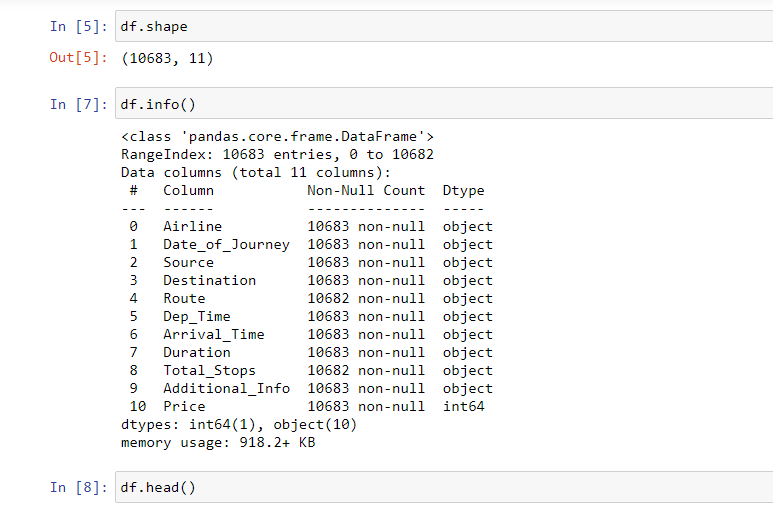
from sklearn.ensemble import RandomForestRegressor

from sklearn.model\_selection import RandomizedSearchCV

import pickle,joblib

from sklearn.model\_selection import cross\_val\_score





## **Step2: Data Cleaning of Numeric Value**

Second Step is Data Cleaning check the how many number of null values are available in dataset using df.isnull().sum() command. If null value is available at that time clean all null value using some different strategies like using mean, median and most frequent value. Now here I have clean first Numeric Values in dataset.

**Code:**

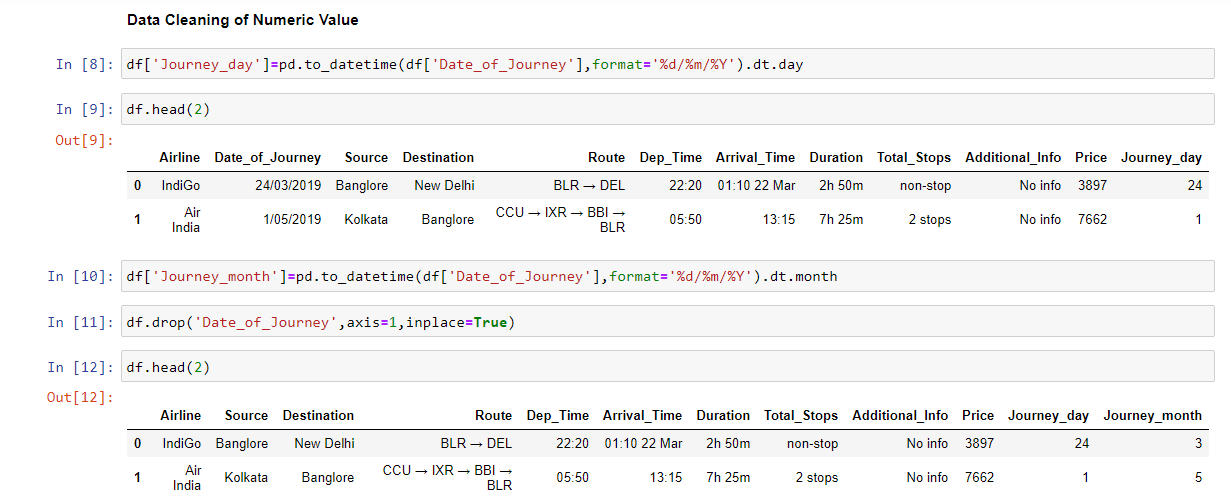
df['Journey\_day']=pd.to\_datetime(df['Date\_of\_Journey'],format='%d/%m/%Y').dt.day

df.head(2)

df['Journey\_month']=pd.to\_datetime(df['Date\_of\_Journey'],format='%d/%m/%Y').dt.month

df.drop('Date\_of\_Journey',axis=1,inplace=True)

df.head(2)



df.isnull().sum()

df.dropna(inplace=True)

df['Dep\_hour']=pd.to\_datetime(df.Dep\_Time).dt.hour

df['Dep\_min']=pd.to\_datetime(df.Dep\_Time).dt.minute

df.drop('Dep\_Time',axis=1,inplace=True)

df.head(2)

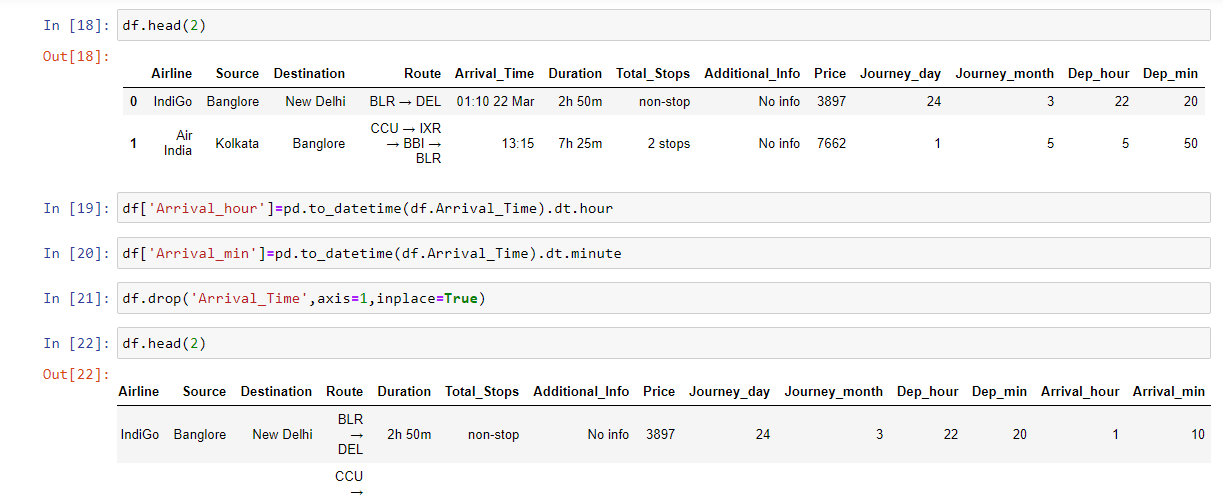


df['Arrival\_hour']=pd.to\_datetime(df.Arrival\_Time).dt.hour

df['Arrival\_min']=pd.to\_datetime(df.Arrival\_Time).dt.minute

df.drop('Arrival\_Time',axis=1,inplace=True)

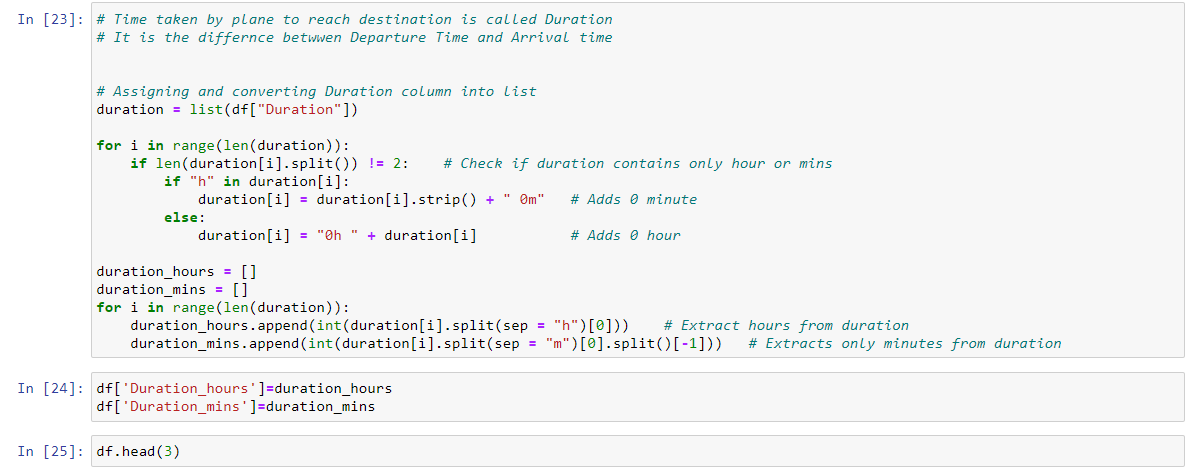
df.head(2)



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(df["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

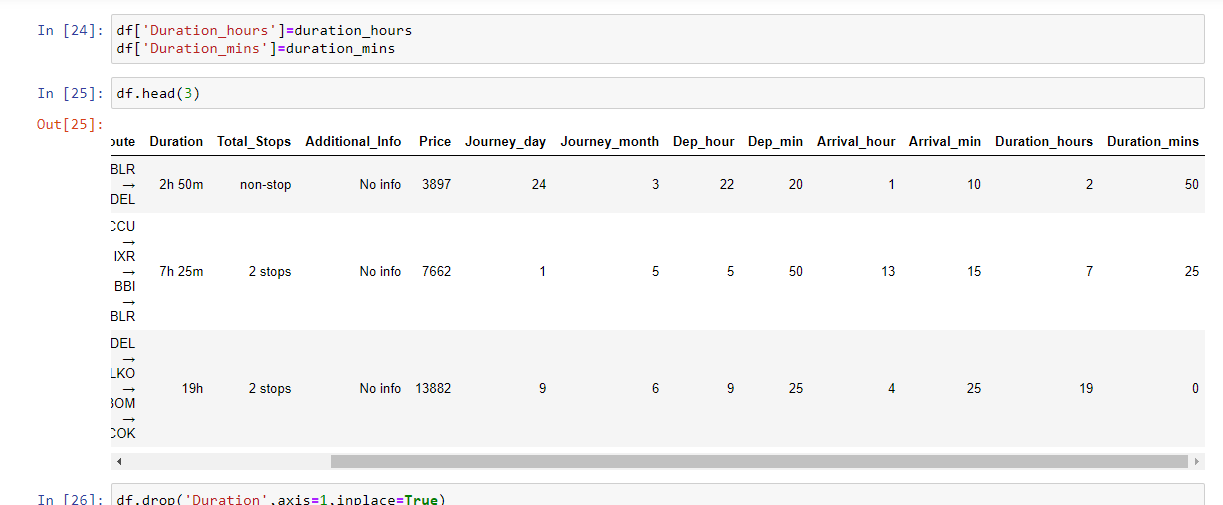
df['Duration\_hours']=duration\_hours

df['Duration\_mins']=duration\_mins

df.head(3)

df.drop('Duration',axis=1,inplace=True)

df.head(2)



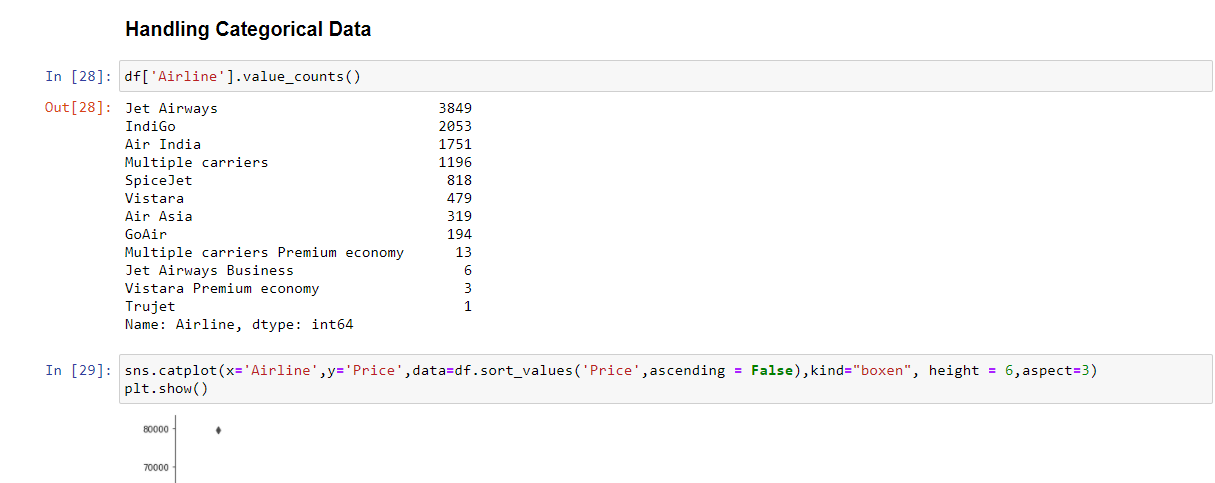
### **Step3: Handling Categorical Data**

Here We have a Categorical Variable in our dataset like: Airline, Source, Destination, Route, and Additional\_nfo. I have to clean this Categorical variable because of better prediction accuracy. If our data has properly clean, then we got a pretty much Accuracy and our model will be train properly. We have to understand this line “Garbage in garbage out”. If we give garbage data in our model, then our model will predict some garbage output and if we give clean and perfect data to out model then model will predict better Accuracy and output.

**Clean Airline Column**

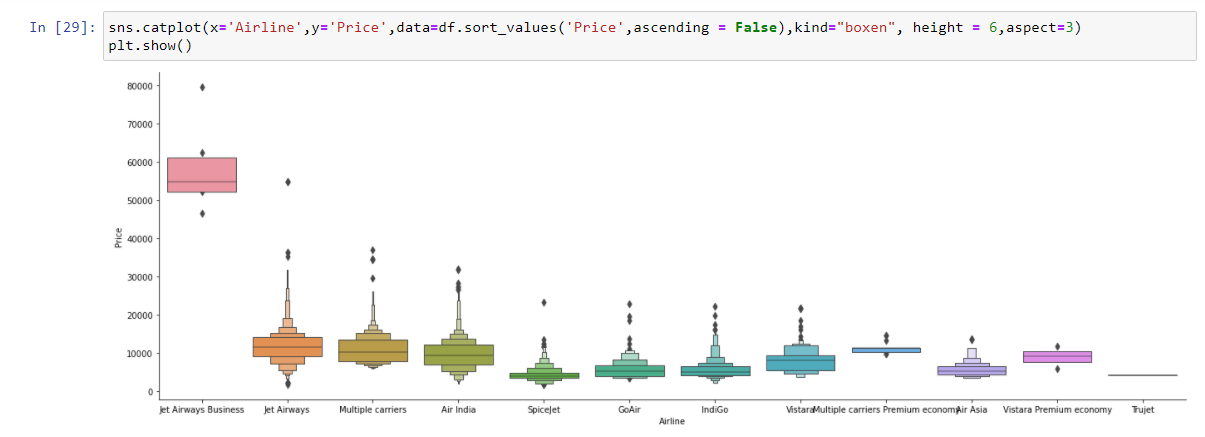
**Code:**

df['Airline'].value\_counts()



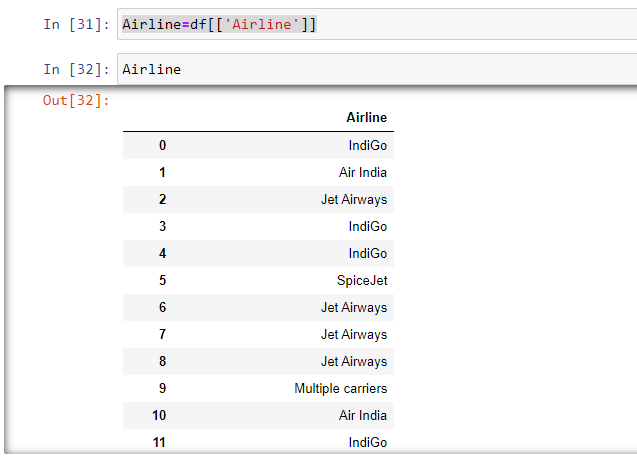
sns.catplot(x='Airline',y='Price',data=df.sort\_values('Price',ascending = False),kind="boxen", height = 6,aspect=3)

plt.show()



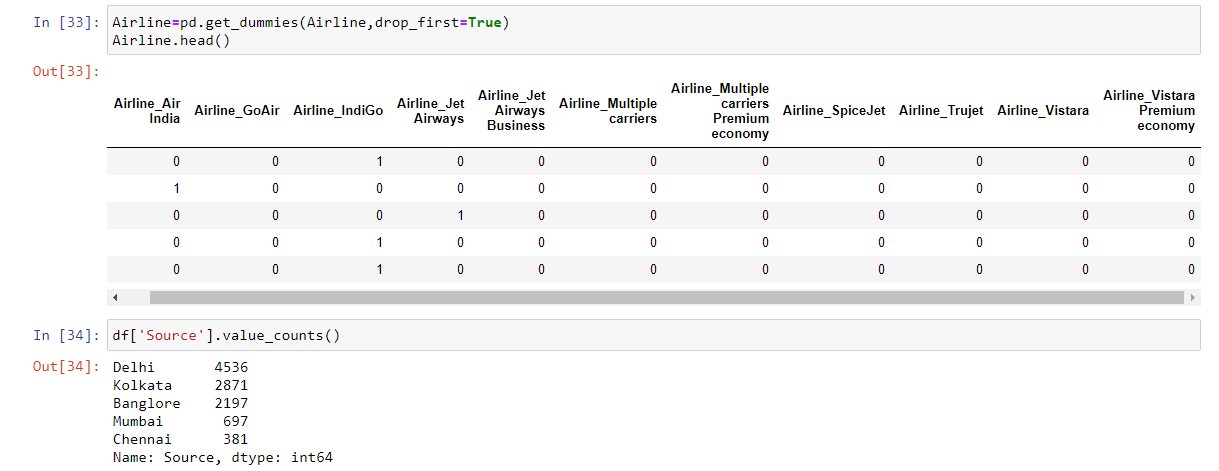
df.head(2)

Airline=df[['Airline']]



Airline=pd.get\_dummies(Airline,drop\_first=True)

Airline.head()



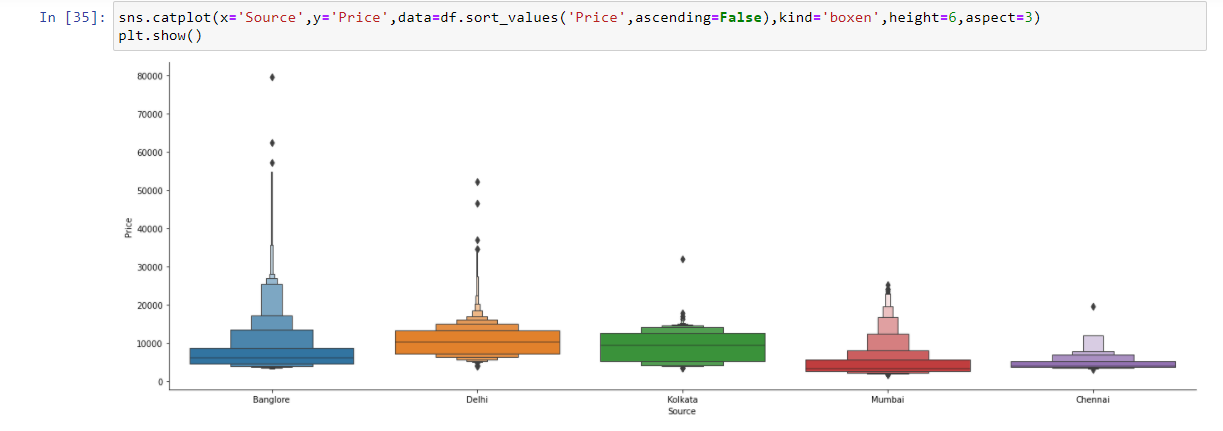
**Clean Source Column**

**Code:**

df['Source'].value\_counts()

sns.catplot(x='Source',y='Price',data=df.sort\_values('Price',ascending=False),kind='boxen',height=6,aspect=3)

plt.show()

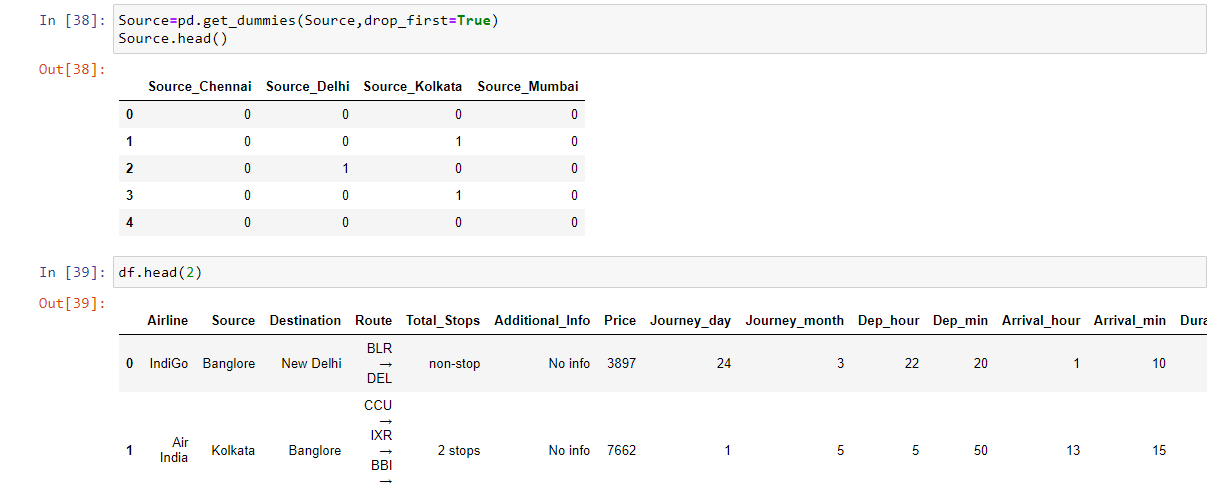


Source=df[['Source']]

Source=pd.get\_dummies(Source,drop\_first=True)

Source.head()

df.head(2)

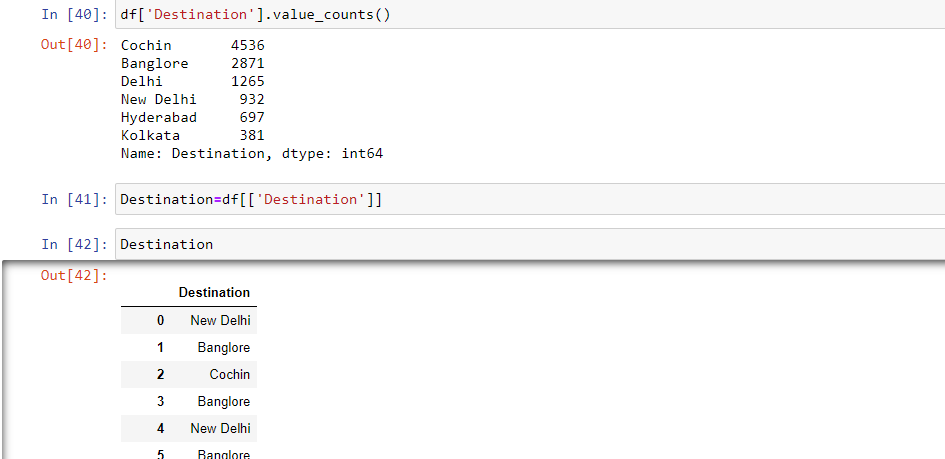


**Clean Destination Column**

**Code:**

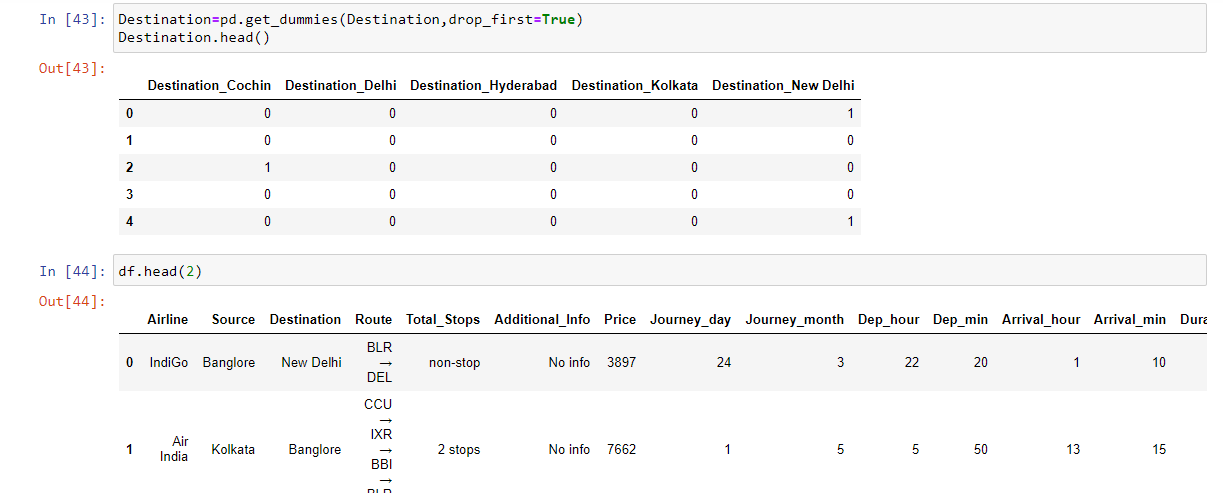
df['Destination'].value\_counts()

Destination=df[['Destination']]



Destination=pd.get\_dummies(Destination,drop\_first=True)

Destination.head()



df.head(2)

df.drop(['Route','Additional\_Info'],axis=1,inplace=True)

df.head(1)

df['Total\_Stops'].value\_counts()

df['Total\_Stops'].unique()



stops\_map={'non-stop':0,'1 stop':1,'2 stops':2,'3 stops':3,'4 stops':4}

df['Total\_Stops']=df['Total\_Stops'].map(stops\_map)

df['Total\_Stops'].unique()

df.head()



### **Step4: Concatenate Data**

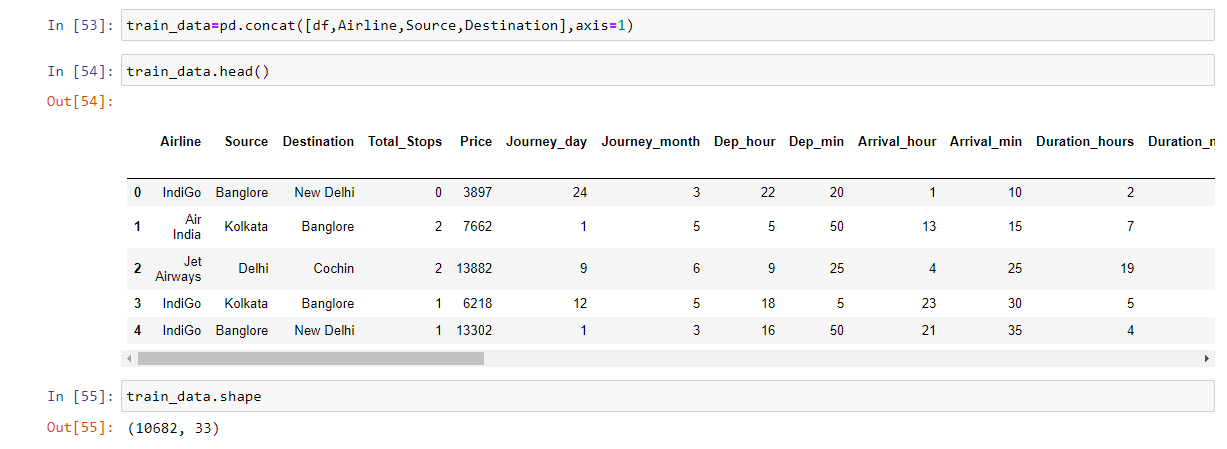
Now we have three clean new variables like Airline, Source and Destination. These variables we have to concatenate with our data frame. By using following code.

**Code:**

train\_data=pd.concat([df,Airline,Source,Destination],axis=1)

train\_data.head()

train\_data.shape



train\_data.drop(['Airline','Source','Destination'],axis=1,inplace=True)

train\_data.shape

train\_data.head(1)

train\_data.columns

****

### **Step5: Create Vectors**

Now we need to create vectors like X and y. Use this 2 vectors we can easily Split data into train & test and building a model.

**Code:**

X=train\_data.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',

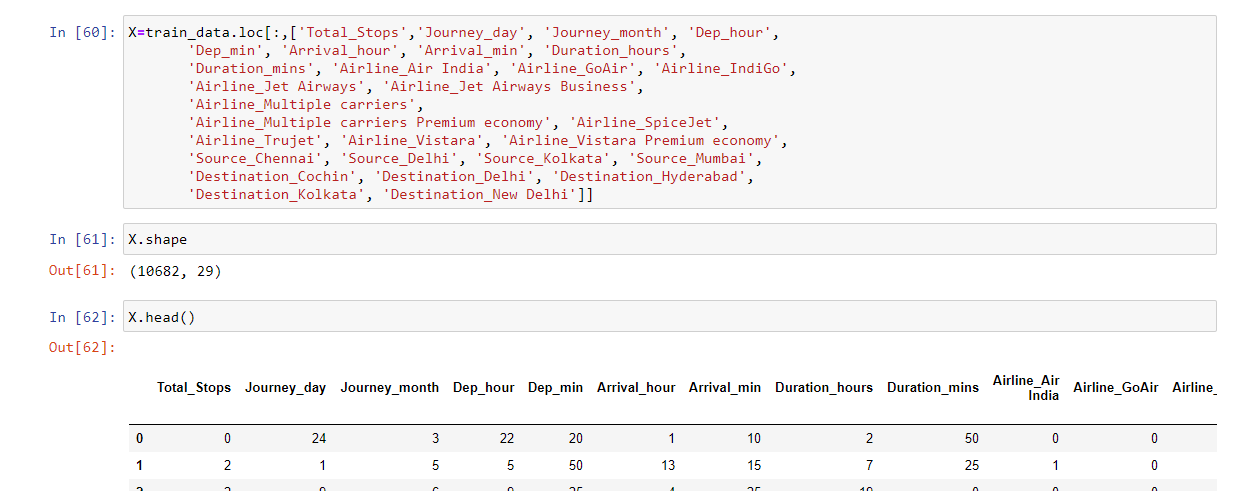
'Source\_Chennai', 'Source\_Delhi', 'Source\_Kolkata', 'Source\_Mumbai',

'Destination\_Cochin', 'Destination\_Delhi', 'Destination\_Hyderabad',

'Destination\_Kolkata', 'Destination\_New Delhi']]

X.shape

X.head()



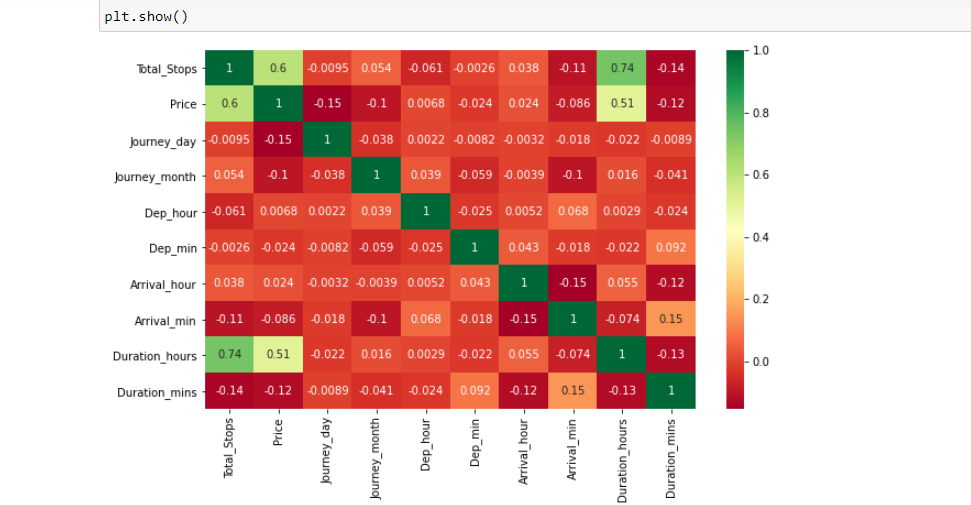
y=train\_data.iloc[:,1]

**Finding correlation between Independent and dependent attributes**

plt.figure(figsize = (18,18))

sns.heatmap(df.corr(), annot = True, cmap = "RdYlGn")

plt.show()



### **Step6: Features Selection**

Find the important features using ExtraTreesRegressor. This is a very good algorithm for selecting and finding the important features dependent and independent variables. Which feature is important for predicting the price of flight so that’s way we need to try this technique.

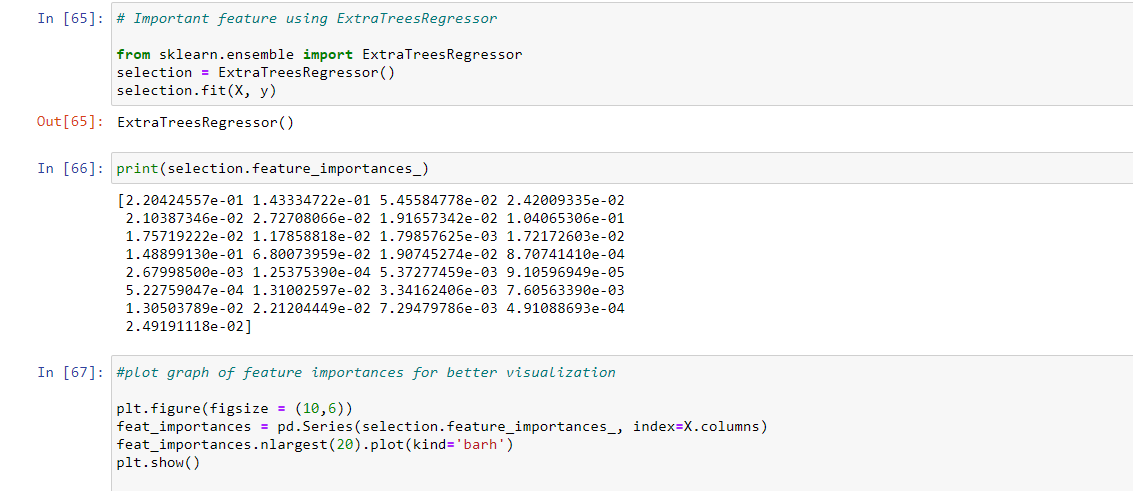
**Code:**

from sklearn.ensemble import ExtraTreesRegressor

selection = ExtraTreesRegressor()

selection.fit(X, y)

print(selection.feature\_importances\_)



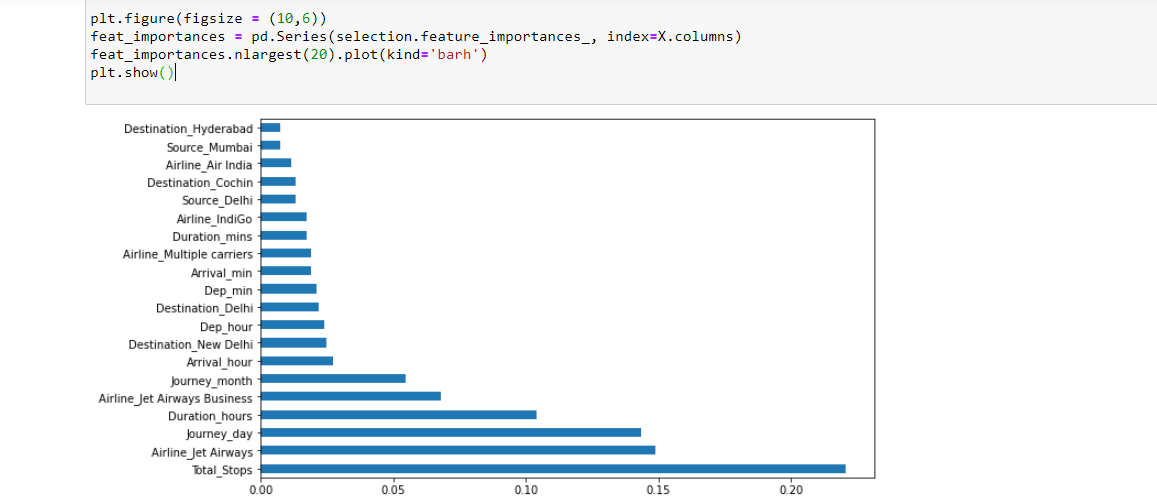
**Plot graph of feature importances for better visualization**

plt.figure(figsize = (10,6))

feat\_importances = pd.Series(selection.feature\_importances\_, index=X.columns)

feat\_importances.nlargest(20).plot(kind='barh')

plt.show()



### **Step7: Split Data into Train and Test**

Building a Machine Learning Model, we need to data divided in to train test and Split by using Scikit-learn library. Use 80% data for training and 20% data for testing.

**from sklearn.model\_selection import train\_test\_split**

**Code:**

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)

### **Step8: Model Building**

Build a Machine Learning Model using Scikit-learn library. For Multiple Linear Regression we need to use RandomForestRegressor. A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is controlled with the max\_samples parameter if bootstrap=True (default), otherwise the whole dataset is used to build each tree.

**Code:**

from sklearn.ensemble import RandomForestRegressor

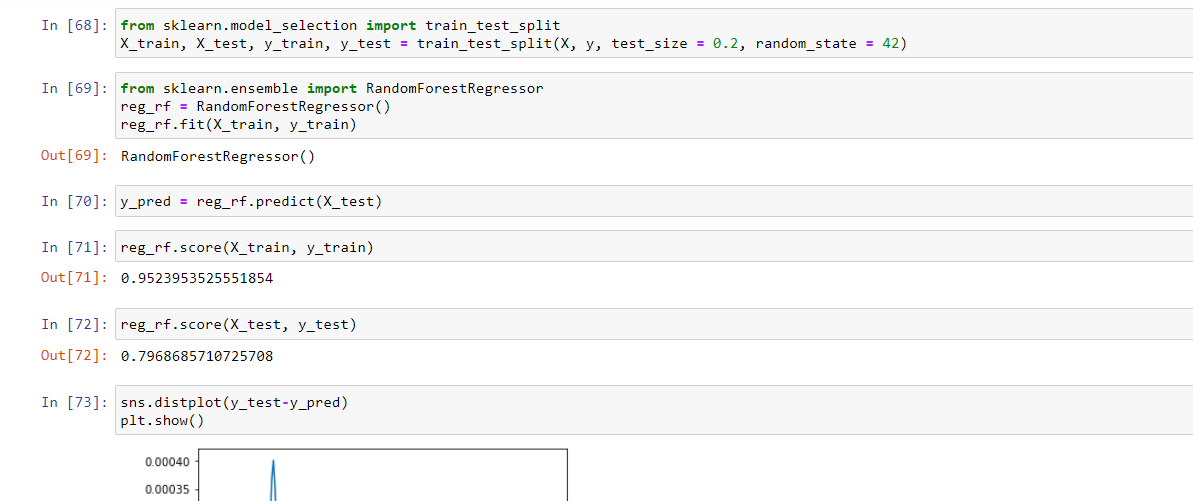
reg\_rf = RandomForestRegressor()

reg\_rf.fit(X\_train, y\_train)

y\_pred = reg\_rf.predict(X\_test)

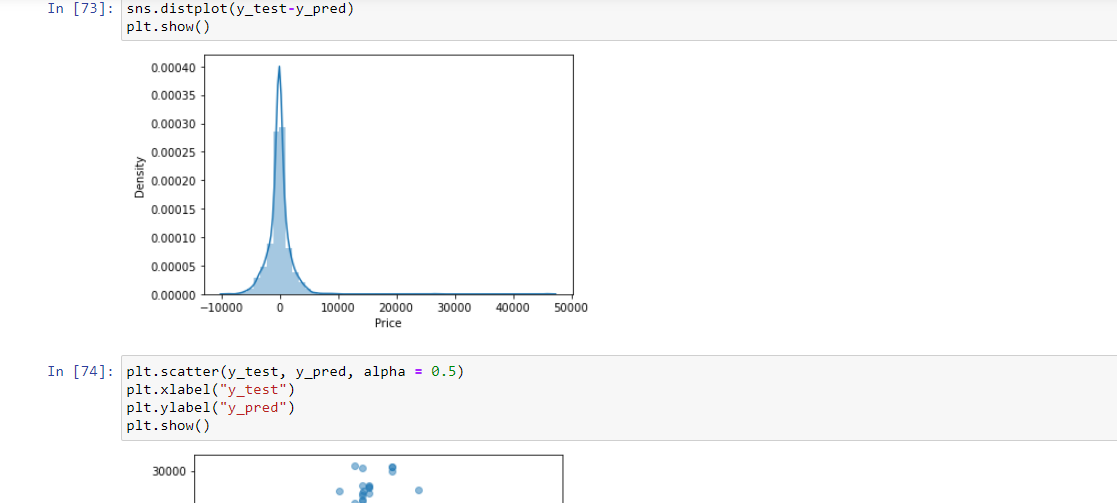
reg\_rf.score(X\_train, y\_train)

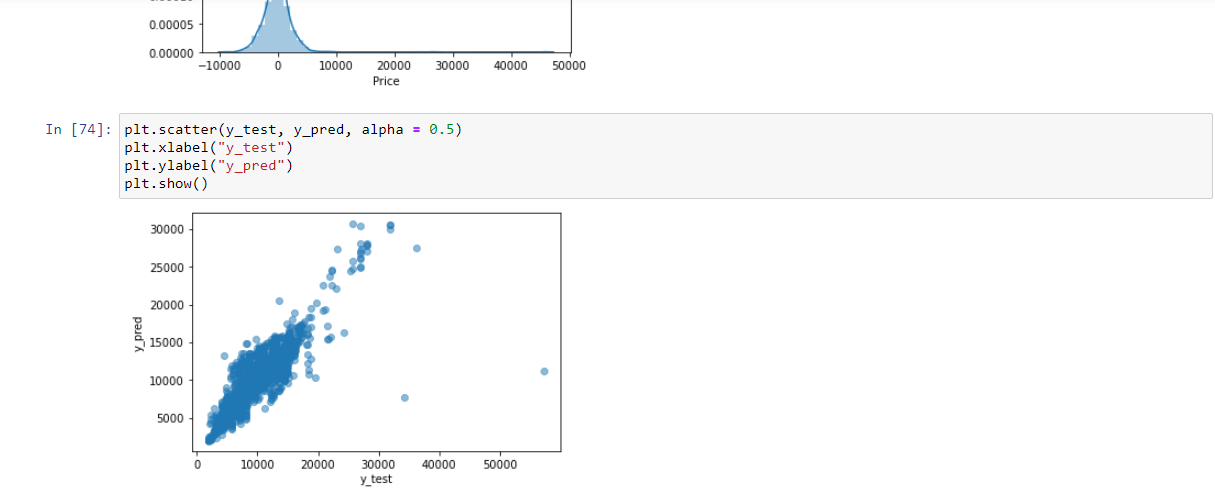
reg\_rf.score(X\_test, y\_test)



sns.distplot(y\_test-y\_pred)

plt.show()





## **Step9: Hyperparameter Tuning**

## 

When creating a machine learning model, you'll be presented with design choices as to how to define your model architecture. Often times, we don't immediately know what the optimal model architecture should be for a given model, and thus we'd like to be able to explore a range of possibilities. In true machine learning fashion, we'll ideally ask the machine to perform this exploration and select the optimal model architecture automatically. Parameters which define the model architecture are referred to as **hyperparameters** and thus this process of searching for the ideal model architecture is referred to as hyperparameter tuning.

In hyperparameter tuning there are two techniques first one is a RandomizedSearchCV and another one is a GreedSearchCV.

In this Project we need to perform on RandomizedSearchCV.

**Code:**

from sklearn.model\_selection import RandomizedSearchCV

#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

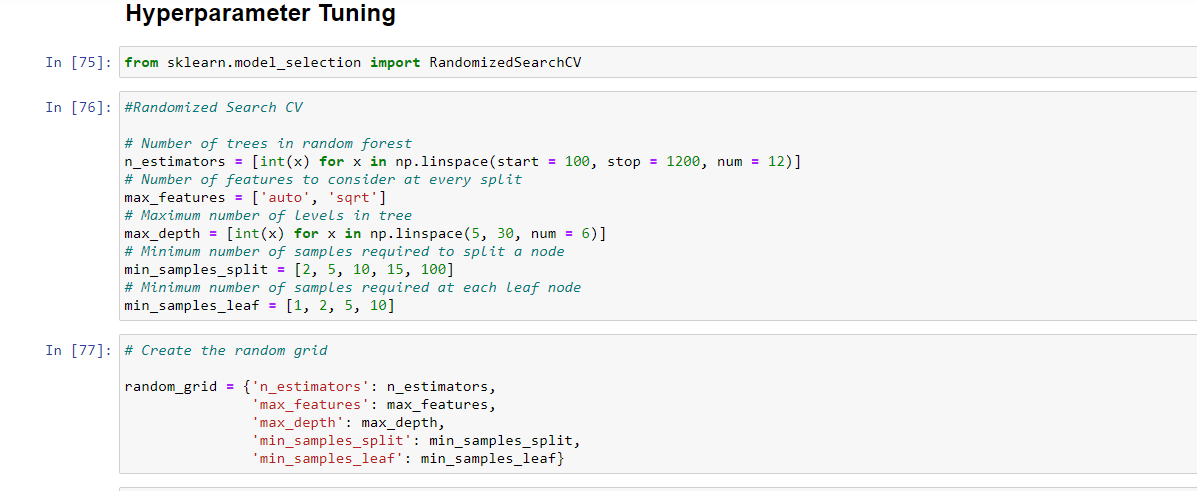
random\_grid = {'n\_estimators': n\_estimators,

'max\_features': max\_features,

'max\_depth': max\_depth,

'min\_samples\_split': min\_samples\_split,

'min\_samples\_leaf': min\_samples\_leaf}

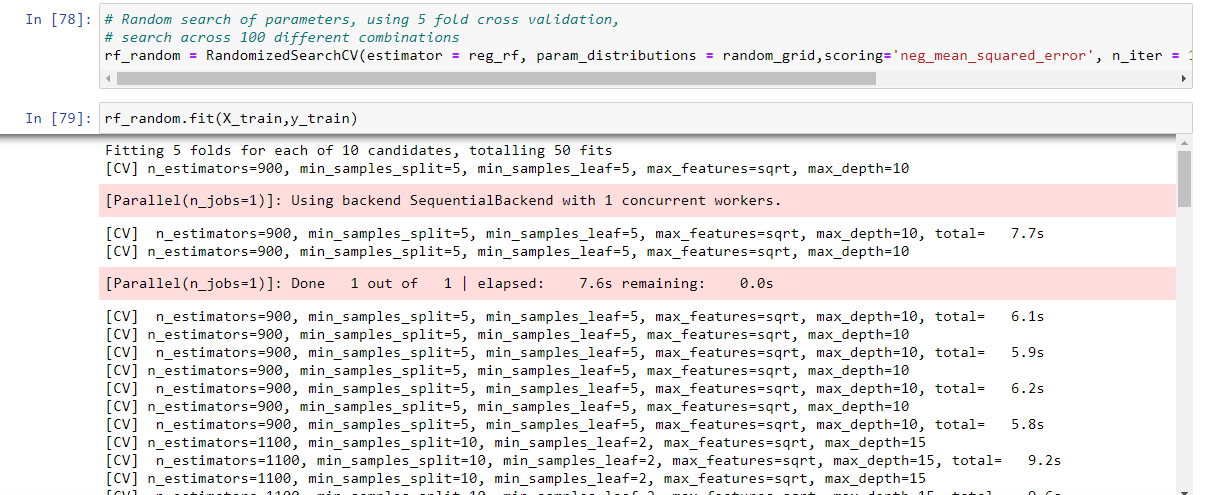


# Random search of parameters, using 5 fold cross validation,

# search across 100 different combinations

rf\_random = RandomizedSearchCV(estimator = reg\_rf, param\_distributions = random\_grid,scoring='neg\_mean\_squared\_error', n\_iter = 10, cv = 5, verbose=2, random\_state=42, n\_jobs = 1)

rf\_random.fit(X\_train,y\_train)



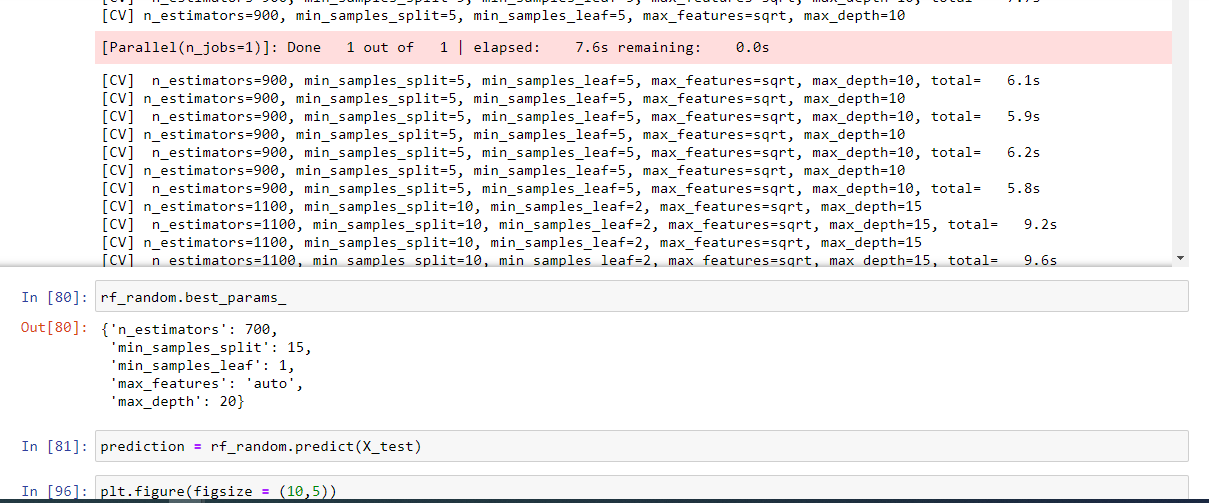
rf\_random.best\_params\_

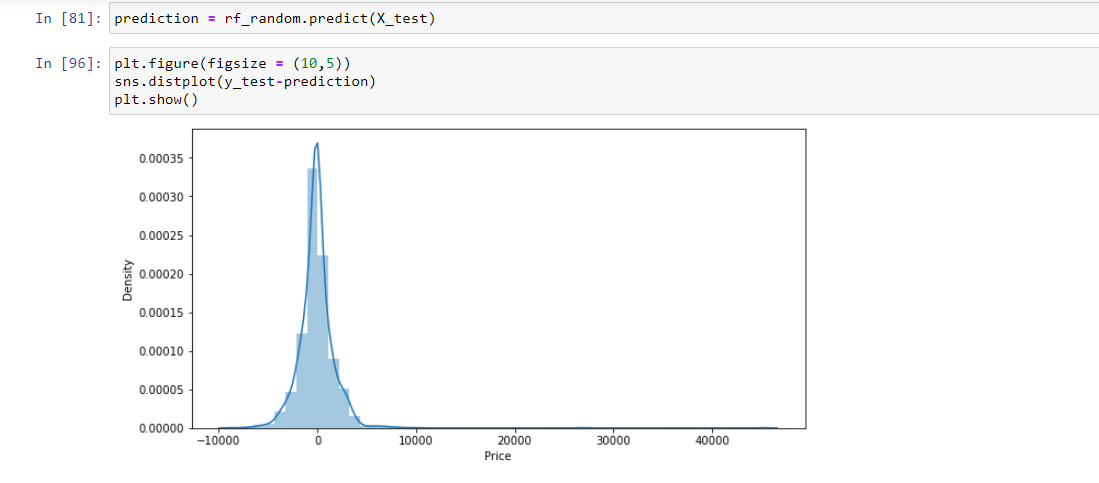
prediction = rf\_random.predict(X\_test)

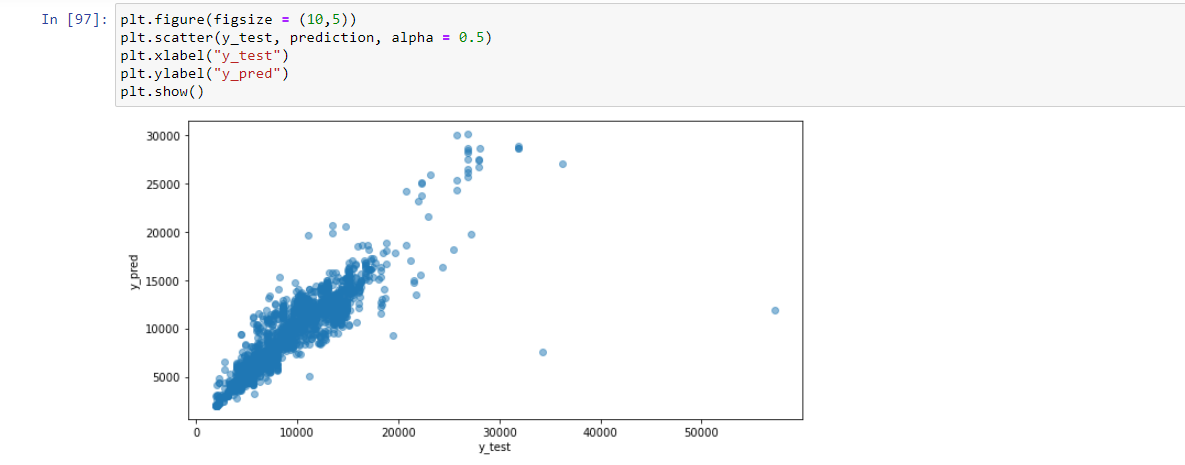
plt.figure(figsize = (8,8))

sns.distplot(y\_test-prediction)

plt.show()







## **Step10: Save Model to reuse it again**

**Code:**

import pickle,joblib

pickle.dump(reg\_rf,open('flight.pkl','wb'))

joblib.dump(reg\_rf,'flight.jbl')

## **Step11: Load Pickel Model**

**Code:**

model\_pkl=pickle.load(open('flight.pkl','rb'))

model\_pkl.score(X\_test,y\_test)

model\_pkl.score(X\_train,y\_train)

from sklearn.model\_selection import cross\_val\_score

cv=cross\_val\_score(RandomForestRegressor(),X\_train,y\_train)

print('n\_split',cv)

print('Average',np.average(cv))



## **Step12: Deploy Machine Learning Model**

## **Model Deployment Using Flask, AWS EC instance.**

# Flight-Price-Prediction Model Link: <http://ec2-52-91-122-24.compute-1.amazonaws.com:8080/>

# E:\prasad\SMIIT\Working Project\Flight Price Prediction\Blog img\10.png

# E:\prasad\SMIIT\Working Project\Flight Price Prediction\Blog img\12.png

# Created by: Balaprasad Garudkar

# \*\*\*\*\*\*