In [1]:

```
######### Table of content #############
#1. Importing flight fare data from kaggle
#2. EDA
#3. data visualization with univariate, bivariate analysis
#4. data pre processing
#5. changing string format into numeric
#6. data partitioning
#7. building model using RandomForestClassifier
#8. check accuracy
#9. hyperparameter tunning using randomizedsearchCV
#10. check final accuracy, RMSE value.






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

In [2]:

```
sns.set()
```

In [3]:

```
train_data = pd.read_excel("Data_Train.xlsx")
```

In [4]:

```
train_data
```

Out[4]:

	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	13882
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	13302
...	...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	9/04/2019	Kolkata	Banglore	CCU → BLR	19:55	22:25	2h 30m	non-stop	No info	4107
10679	Air India	27/04/2019	Kolkata	Banglore	CCU →	20:45	23:20	2h 35m	non-stop	No info	4145

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
10680	Jet Airways	27/04/2019	Banglore	Delhi	BLR → DEL	08:20	11:20	3h	non-stop	No info	7229
10681	Vistara	01/03/2019	Banglore	New Delhi	BLR → DEL	11:30	14:10	2h 40m	non-stop	No info	12648
10682	Air India	9/05/2019	Delhi	Cochin	DEL → GOI → BOM → COK	10:55	19:15	8h 20m	2 stops	No info	11753

10683 rows × 11 columns

In [5]:

```
pd.set_option('display.max_columns',None)
```

In [6]:

```
train_data.head()
```

Out[6]:

	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	13882
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	13302

In [7]:

```
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
```

In [8]:

```
########## data pre-processing ##########

train_data['Duration'].value_counts()     ####### how many times a duration has been used in the da
ta set #####
```

Out[8]:

```
2h 50m     550
1h 30m     386
2h 55m     337
2h 45m     337
2h 35m     329
          ...
31h 30m      1
30h 10m      1
47h          1
3h 25m       1
13h 35m      1
Name: Duration, Length: 368, dtype: int64
```

In [9]:

```
train_data.dropna(inplace=True) ######## dropping na values ########
```

In [10]:

```
train_data.isnull().sum()  ######### checking Null values ###########
```

Out[10]:

```
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 [11]:

```
######### Exploratory Data Analysis ############
###### date of journey, dep_time is in string format and arrival time is having character type dat
a like "mar","jun". etc.. we need to change these.

train_data["Journey_day"] = pd.to_datetime(train_data.Date_of_Journey,format = "%d/%m/%Y").dt.day
#### .dt.day will extract day from the date ###

train_data["Joourney_month"] = pd.to_datetime(train_data["Date_of_Journey"],format = "%d/%m/%Y").dt
.month ### .dt.month will extract month from the date ###
```

In [12]:

```
train_data.head()
```

Out[12]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Jou
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	

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Jou
2	Jet Airways	9/06/2019	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	
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	13302	

In [13]:

```
###### since we have converted "dat_of_journey" into integer , we can now drop it as it is of no u
se ######

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

In [14]:

```
train_data.head()
```

Out[14]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourne
0	IndiGo	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1	

In [15]:

```
######### similar to date_of_journey we can extract hour and minutes from dep_time also ##########
##

train_data["Dep_hour"] = pd.to_datetime(train_data["Dep_Time"]).dt.hour   #### .dt.hour will
extract hour from the time ###

train_data["Dep_min"] = pd.to_datetime(train_data["Dep_Time"]).dt.minute ### .dt.minute will
extract minute from the date ###
```

In [16]:

```
train_data.head()
```

Out[16]:

	Airline	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourne
0	IndiGo	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897	24	
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662	1	
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882	9	
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218	12	
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302	1	

In [17]:

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

In [18]:

```
train_data.head()
```

Out[18]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	D
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 [19]:

```
train_data["Arrival_hour"] = pd.to_datetime(train_data["Arrival_Time"]).dt.hour   #### .dt.hour
will extract hour from the time ###
```

```
train_data["Arrival_min"] = pd.to_datetime(train_data["Arrival_Time"]).dt.minute ### .dt.minute wil
l extract minute from the date ###
```

In [20]:

```
train_data.head()
```

Out[20]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	D
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 [21]:

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

In [22]:

```
train_data.head()
```

Out[22]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	Dep_hour	Dep
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	

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	Dep_hour	Dep

In [23]:

```
######## assigning and converting duration column into list ########

duration = list(train_data["Duration"])
```

In [24]:

```
for i in range(len(duration)):
    if len(duration[i].split()) != 2:            ######## check if duration contains only hour and
minute ########
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m "      ####### adding 0 minute #######
        else:
            duration[i] = " 0h "+duration[i]                ####### adding 0 hour ##########

duration_hours = []
duration_mins = []

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

In [25]:

```
########## adding "duration_hours" and "duration_mins" to train datasets ######

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

In [26]:

```
train_data.head()
```

Out[26]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	Dep_hour	Dep
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 [27]:

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

In [28]:

```
train_data.head()
```

Out[28]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Joourney_month	Dep_hour	Dep_min	Arri
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	

In [29]:

```
########## handling categorical data #############
########## nominal data : data are not in any order : one-hot encoder
########## ordinal data : data are in order : label encoder

train_data["Airline"].value_counts()
```

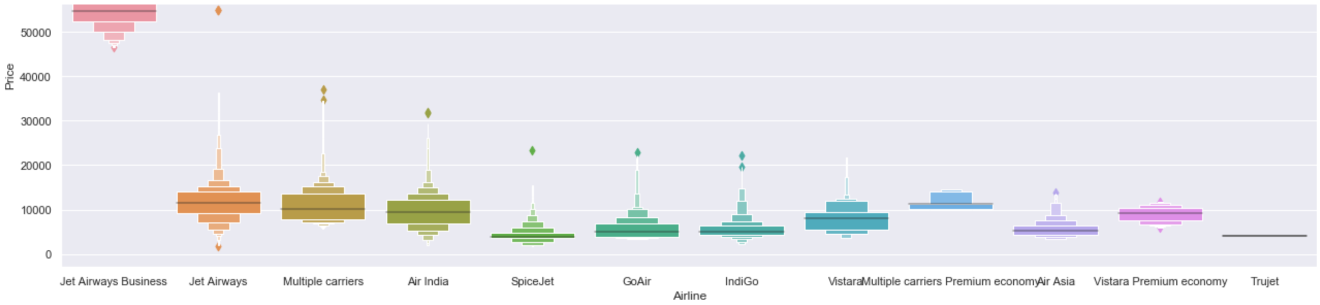
Out[29]:

```
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 [30]:

```
###### from graph we can see that jet airways business have the highest price ######
##### apart from 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),kin
d = "boxen", height=6, aspect=3)
plt.show()
```



In [32]:

```
###### as Airline is nominal categorical data , hence we will use one hot encoding ##########

Airline   = train_data[["Airline"]]
Airline   = pd.get_dummies(Airline, drop_first=True)
Airline.head()
```

Out[32]:

	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	Ai
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	

In [34]:

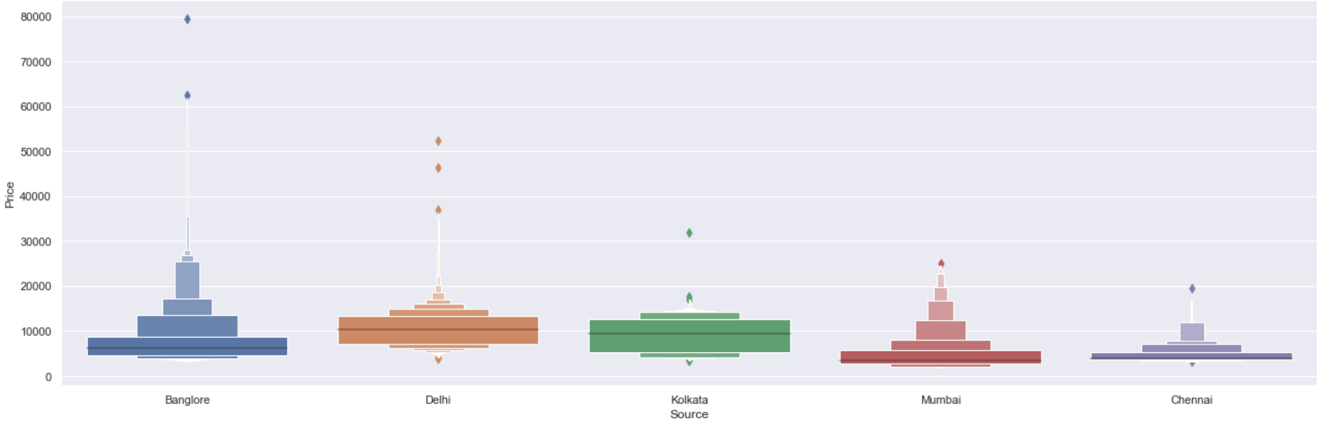
```
train_data["Source"].value_counts()
```

Out[34]:

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

In [35]:

```
sns.catplot(y = "Price",x = "Source",data = train_data.sort_values("Price", ascending = False),kind
= "boxen", height=6, aspect=3)
plt.show()
```



In [40]:

```
##### as source is also a nominal categorical data hence we can apply one-hot encoding
#############

Source = train_data[["Source"]]
Source  = pd.get_dummies(Source,drop_first = True)
Source.head()
```

Out[40]:

	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 [41]:

```
train_data["Destination"].value_counts()
Destination = train_data[["Destination"]]
Destination  = pd.get_dummies(Destination,drop_first = True)
Destination.head()
```

Out[41]:

	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 [42]:

```
train_data["Route"]
```

Out[42]:

```
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 [43]:

```
###### Additional_Info contains almost 80% no_info#######
###### Route and Total_Stop are related to each other #########

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

In [45]:

```
train_data["Total_Stops"].value_counts()
```

Out[45]:

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

In [46]:

```
######### as Total_Stops is a ordinal categorical data hence we can use label encoder here #######
#
#########  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 [47]:

```
train_data.head()
```

Out[47]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Joourney_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 [49]:

```
########## concatenate data frame ###########

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

In [50]:

```
data_train
```

Out[50]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_m
0	IndiGo	Banglore	New Delhi	0	3897	24	3	22	20	1	1
1	Air India	Kolkata	Banglore	2	7662	1	5	5	50	13	1
2	Jet Airways	Delhi	Cochin	2	13882	9	6	9	25	4	2
3	IndiGo	Kolkata	Banglore	1	6218	12	5	18	5	23	3
4	IndiGo	Banglore	New Delhi	1	13302	1	3	16	50	21	3
...	...	...	...	...	...	...	...	...	...	...	
10678	Air Asia	Kolkata	Banglore	0	4107	9	4	19	55	22	2
10679	Air India	Kolkata	Banglore	0	4145	27	4	20	45	23	2
10680	Jet Airways	Banglore	Delhi	0	7229	27	4	8	20	11	2
10681	Vistara	Banglore	New Delhi	0	12648	1	3	11	30	14	1
10682	Air India	Delhi	Cochin	2	11753	9	5	10	55	19	1

10682 rows × 33 columns

Airline Source Destination Total_Stops Price Journey_day Joourney_month Dep_hour Dep_min Arrival_hour Arrival_m

In [51]:

```
####### now we can drop first three column as we have already applied onehot encoding of all the p
arameters ##########

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

In [52]:

```
data_train.head()
```

Out[52]:

	Total_Stops	Price	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins
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

In [53]:

```
data_train.shape
```

Out[53]:

```
(10682, 30)
```

In [54]:

```
############### test data ##################

test_data = pd.read_excel("Test_set.xlsx")
```

In [55]:

```
test_data
```

Out[55]:

	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
...	...	...	...	...	...	...	...	...	...	...
2666	Air India	6/06/2019	Kolkata	Banglore	CCU → DEL → BLR	20:30	20:25 07 Jun	23h 55m	1 stop	No info
2667	IndiGo	27/03/2019	Kolkata	Banglore	CCU →	14:20	16:55	2h 35m	non-stop	No info

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info
2667	IndiGo	27/03/2019	Kolkata	Banglore	BLR	14:20	16:55	2h 35m	non-stop	No info
2668	Jet Airways	6/03/2019	Delhi	Cochin	DEL → BOM → COK	21:50	04:25 07 Mar	6h 35m	1 stop	No info
2669	Air India	6/03/2019	Delhi	Cochin	DEL → BOM → COK	04:00	19:15	15h 15m	1 stop	No info
2670	Multiple carriers	15/06/2019	Delhi	Cochin	DEL → BOM → COK	04:55	19:15	14h 20m	1 stop	No info

2671 rows × 10 columns

In [56]:

```
test_data.head()
```

Out[56]:

	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]:

```
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
```

In [58]:

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

Out[58]:

```
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
```

In [59]:

```
######### Exploratory Data Analysis ############
###### date of journey, dep_time is in string format and arrival time is having character type dat
a like "mar","jun". etc.. we need to change these.

test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey,format = "%d/%m/%Y").dt.day  ##
## .dt.day will extract day from the date ###

test_data["Joourney_month"] = pd.to_datetime(test_data["Date_of_Journey"],format = "%d/%m/%Y").dt.m
onth ### .dt.month will extract month from the date ###
```

In [60]:

```
###### since we have converted "dat_of_journey" into integer , we can now drop it as it is of no u
se ######

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

In [61]:

```
test_data.head()
```

Out[61]:

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

In [62]:

```
######### similar to date_of_journey we can extract hour and minutes from dep_time also ##########
##

test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour   #### .dt.hour will extract
hour from the time ###

test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute ### .dt.minute will extract
minute from the date ###
```

In [63]:

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

In [64]:

```
test_data.head()
```

Out[64]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Journey_day	Joourney_month	Dep_hou
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	04:25 07 Jun	10h 55m	1 stop	No info	6	6	1
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	10:20	4h	1 stop	No info	12	5	6
2	Jet Airways	Delhi	Cochin	DEL → BOM → COK	19:00 22 May	23h 45m	1 stop	In-flight meal not included	21	5	1
3	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	21:00	13h	1 stop	No info	21	5	8
4	Air Asia	Banglore	Delhi	BLR → DEL	02:45 25 Jun	2h 50m	non-stop	No info	24	6	2

In [65]:

```
test_data["Arrival_hour"] = pd.to_datetime(test_data["Arrival_Time"]).dt.hour   #### .dt.hour will
extract hour from the time ###

test_data["Arrival_min"] = pd.to_datetime(test_data["Arrival_Time"]).dt.minute ### .dt.minute will
extract minute from the date ###
```

In [66]:

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

In [67]:

```
test_data.head()
```

Out[67]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Journey_day	Joourney_month	Dep_hour	Dep_min	A
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	10h 55m	1 stop	No info	6	6	17	30	
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	4h	1 stop	No info	12	5	6	20	
2	Jet Airways	Delhi	Cochin	DEL → BOM → COK	23h 45m	1 stop	In-flight meal not included	21	5	19	15	
3	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	13h	1 stop	No info	21	5	8	0	
4	Air Asia	Banglore	Delhi	BLR → DEL	2h 50m	non-stop	No info	24	6	23	55	

In [68]:

```
######## assigning and converting duration column into list ########

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

In [69]:

```
for i in range(len(duration)):
    if len(duration[i].split()) != 2:             ######## check if duration contains only hour and minute ########
        if "h" in duration[i]:
            duration[i] = duration[i].strip() + " 0m "       ####### adding 0 minute #######
        else:
            duration[i] = " 0h "+duration[i]                 ####### adding 0 hour ##########

duration_hours = []
duration_mins = []

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

In [70]:

```
########## adding "duration_hours" and "duration_mins" to test datasets ######

test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
```

In [71]:

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

In [72]:

```
test_data.head()
```

Out[72]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hou
0	Jet Airways	Delhi	Cochin	DEL → BOM → COK	1 stop	No info	6	6	17	30	4
1	IndiGo	Kolkata	Banglore	CCU → MAA → BLR	1 stop	No info	12	5	6	20	1
2	Jet Airways	Delhi	Cochin	DEL → BOM → COK	1 stop	In-flight meal not included	21	5	19	15	1
3	Multiple carriers	Delhi	Cochin	DEL → BOM → COK	1 stop	No info	21	5	8	0	2
4	Air Asia	Banglore	Delhi	BLR → DEL	non-stop	No info	24	6	23	55	2

In [73]:

```
########## handling categorical data #############
########## nominal data : data are not in any order : one-hot encoder
########## ordinal data : data are in order : label encoder
```

```
########## ordinal data : data are in order : label encoder

test_data["Airline"].value_counts()
```

Out[73]:

```
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
Jet Airways Business                   2
Vistara Premium economy                2
Name: Airline, dtype: int64
```

In [75]:

```
###### from graph we can see that jet airways business have the highest price ######
##### apart from first airline almost all are having similar median ######
##### Airline vs price #####

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

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
<ipython-input-75-bf3c31c0e468> in <module>
      3 ##### Airline vs price #####
      4 
----> 5 sns.catplot(y = "Price",x = "Airline",data = test_data.sort_values("Price", ascending = Fal
se),kind = "boxen", height=6, aspect=3)
      6 plt.show()

~\anaconda3\lib\site-packages\pandas\core\frame.py in sort_values(self, by, axis, ascending,
inplace, kind, na_position, ignore_index)
   4925 
   4926             by = by[0]
-> 4927             k = self._get_label_or_level_values(by, axis=axis)
   4928 
   4929             if isinstance(ascending, (tuple, list)):

~\anaconda3\lib\site-packages\pandas\core\generic.py in _get_label_or_level_values(self, key,
axis)
   1690             values = self.axes[axis].get_level_values(key)._values
   1691         else:
-> 1692             raise KeyError(key)
   1693 
   1694         # Check for duplicates

KeyError: 'Price'
```

In [76]:

```
###### as Airline is nominal categorical data , hence we will use one hot encoding ##########

Airline   = test_data[["Airline"]]
Airline   = pd.get_dummies(Airline, drop_first=True)
Airline.head()
```

Out[76]:

	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	0	1	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0

	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
2	0	0	0	1	0	0	0	0	0
3	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	0

In [78]:

```
##### as source is also a nominal categorical data hence we can apply one-hot encoding
#############

Source = test_data[["Source"]]
Source  = pd.get_dummies(Source,drop_first = True)
Source.head()
```

Out[78]:

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

In [79]:

```
test_data["Destination"].value_counts()
Destination = test_data[["Destination"]]
Destination  = pd.get_dummies(Destination,drop_first = True)
Destination.head()
```

Out[79]:

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

In [80]:

```
###### Additional_Info contains almost 80% no_info########
###### Route and Total_Stop are related to each other ##########

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

In [81]:

```
######### as Total_Stops is a ordinal categorical data hence we can use label encoder here #######
#
#########  here values are assigned with corresponding keys ############

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

In [82]:

```
test_data.head()
```

Out[82]:

	Airline	Source	Destination	Total_Stops	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration

	Airline	Source	Destination	Total_Stops	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration
0	Jet Airways	Delhi	Cochin	1	6	6	17	30	4	25	
1	IndiGo	Kolkata	Banglore	1	12	5	6	20	10	20	
2	Jet Airways	Delhi	Cochin	1	21	5	19	15	19	0	
3	Multiple carriers	Delhi	Cochin	1	21	5	8	0	21	0	
4	Air Asia	Banglore	Delhi	0	24	6	23	55	2	45	

In [83]:

```
########## concatenate data frame ###########

data_test = pd.concat([test_data,Airline,Source,Destination], axis = 1)
```

In [84]:

```
####### now we can drop first three column as we have already applied onehot encoding of all the p
arameters ##########

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

In [85]:

```
data_test.head()
```

Out[85]:

	Total_Stops	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_ In
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 [86]:

```
data_test.shape
```

Out[86]:

```
(2671, 28)
```

In [88]:

```
data_train.columns
```

Out[88]:

```
Index(['Total_Stops', 'Price', 'Journey_day', 'Joourney_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 [90]:

```
x = data_train.loc[:, ['Total_Stops', 'Journey_day', 'Joourney_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']]                        ##### we have
ll our independent variable here except dependent variable ###########
x.head()
```

Out[90]:

	Total_Stops	Journey_day	Joourney_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_In
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 [91]:

```
y = data_train.iloc[:,1]
```

In [92]:

```
y.head()
```

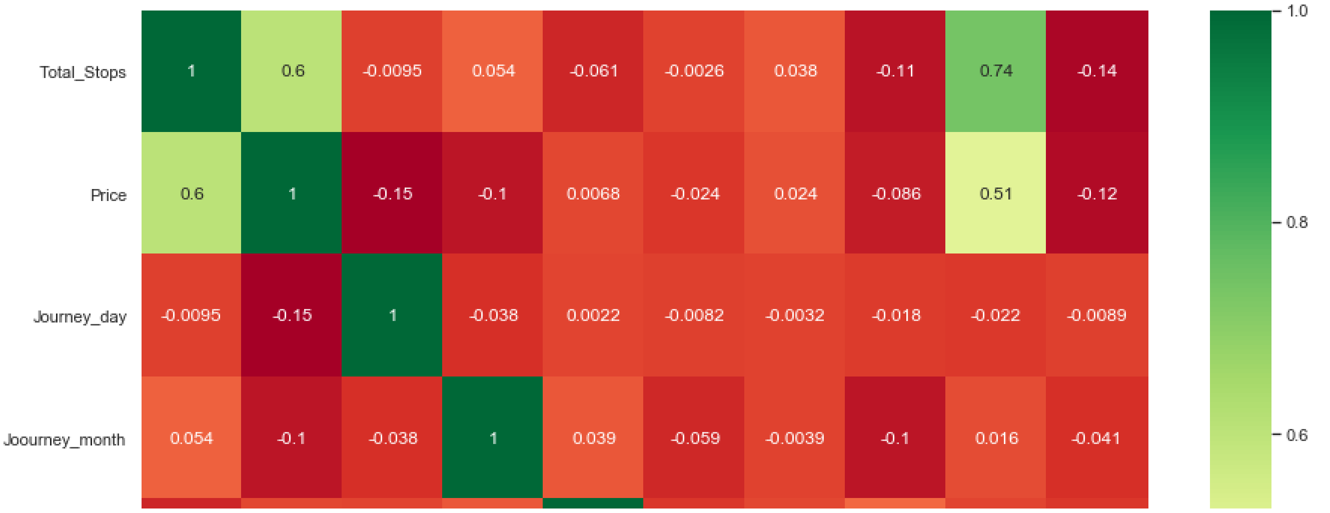
Out[92]:

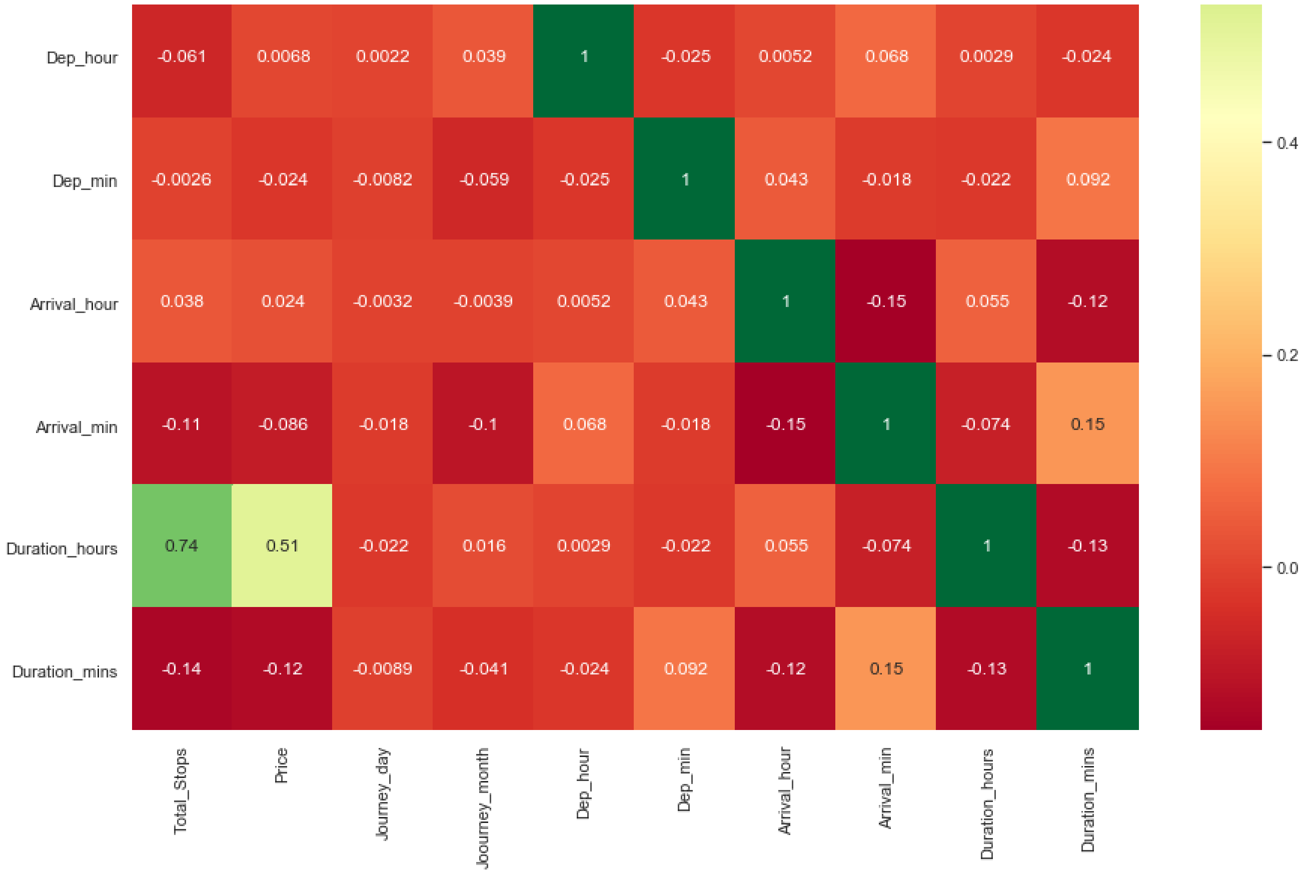
```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

In [94]:

```
########## find correaltion between independent and dependent variable #############

plt.figure(figsize=(15,15))
sns.heatmap(train_data.corr(),annot=True,cmap = "RdYlGn")
plt.show()
```





In [95]:

```
####### Important feature using extraTreeRegressor ########

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

Out[95]:

```
ExtraTreesRegressor(bootstrap=False, ccp_alpha=0.0, criterion='mse',
                    max_depth=None, max_features='auto', max_leaf_nodes=None,
                    max_samples=None, min_impurity_decrease=0.0,
                    min_impurity_split=None, min_samples_leaf=1,
                    min_samples_split=2, min_weight_fraction_leaf=0.0,
                    n_estimators=100, n_jobs=None, oob_score=False,
                    random_state=None, verbose=0, warm_start=False)
```

In [96]:

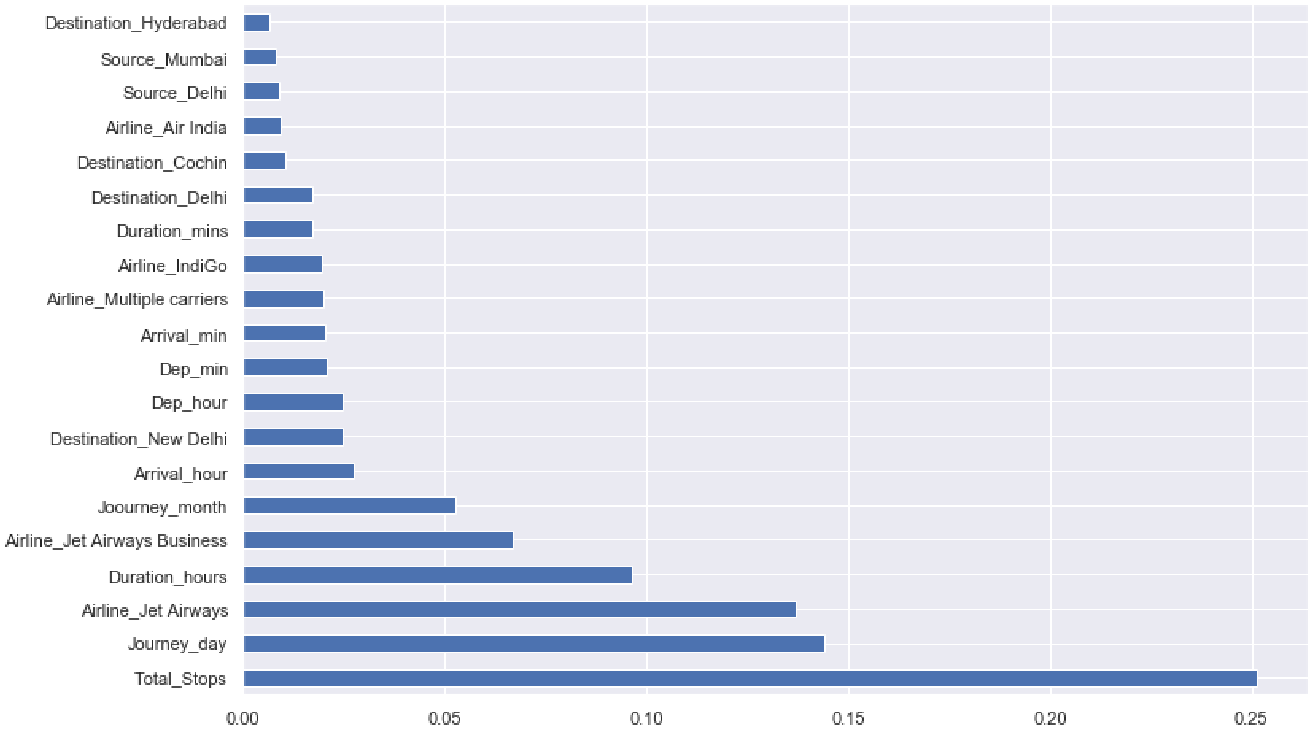
```
print(selection.feature_importances_)
```

```
[2.51385867e-01 1.44243495e-01 5.26502805e-02 2.46295344e-02
 2.09025618e-02 2.75839981e-02 2.02208164e-02 9.63074123e-02
 1.73388461e-02 9.43568839e-03 1.94717422e-03 1.93925626e-02
 1.37071913e-01 6.69296734e-02 1.98750338e-02 8.51248521e-04
 3.84927812e-03 1.16774184e-04 5.00140040e-03 7.83913107e-05
 4.70179598e-04 9.05530587e-03 3.37652805e-03 8.06784688e-03
 1.05625968e-02 1.70362823e-02 6.36691548e-03 4.88064001e-04
 2.47643322e-02]
```

In [98]:

```
###### plot graph for feature importance 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()                                              ########## total stops has more
importance ###########
```



In [99]:

```
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 = True)
```

In [100]:

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

Out[100]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [101]:

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

In [102]:

```
reg_rf.score(x_train,y_train)     ##### R^2 score ##########
```

Out[102]:

```
0.9544700547426853
```

In [103]:

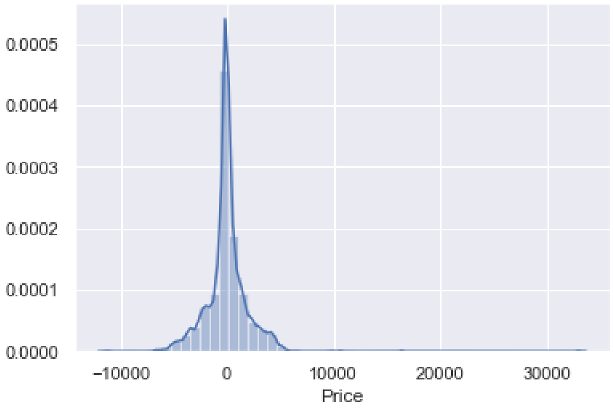
```
reg_rf.score(x_test,y_test)
```

Out[103]:

```
0.8083117905428086
```

In [104]:

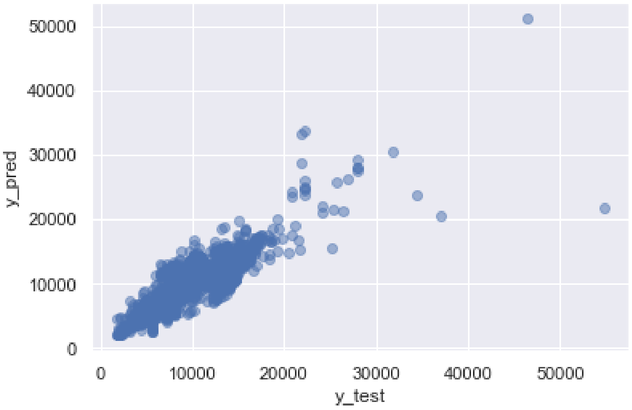
```
sns.distplot(y_test-y_pred)
plt.show()
```



In [105]:

```
############# scatter plot ############

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



In [107]:

```
from sklearn import metrics
```

In [108]:

```
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: 1210.66953190278
MSE: 3931657.519265643
RMsE: 1982.8407700230603
```

In [110]:

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

Out[110]:

```
0.8083117905428086
```

In [112]:

```
############# Hyper parameter tunning ##############

from sklearn.model_selection import RandomizedSearchCV

##### no. of trees in the random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]

####### no. of features to consider in every split #########

max_features = ['auto','sqrt']

###### max no. of levels in tree###########

max_depth = [int(x) for x in np.linspace(5,30, num = 6)]

##### min. no of samples required to split a node ########

min_samples_split = [2,5,10,15,100]

####### min no. of samples required at each leaf node #######

min_samples_leaf = [1,2,5,10]
```

In [117]:

```
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 [120]:

```
rf_random = RandomizedSearchCV(estimator = reg_rf,param_distributions= random_grid,scoring='neg_mea
n_squared_error',n_iter = 10, cv =5,verbose=2,random_state=42,n_jobs=1)
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=   7.1s
[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:    7.0s remaining:    0.0s
```

```
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=sqrt, max_depth=10,
total=   7.9s
[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=   8.5s
[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=   9.1s
[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=   8.8s
[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=  16.2s
[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=  15.0s
[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=  10.6s
[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=  14.2s
[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=  13.1s
[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=   8.6s
[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=   8.2s
[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=   6.2s
[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=   5.8s
[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=   5.5s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=  10.8s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=  11.1s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=  12.6s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=  14.7s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=auto, max_depth=15,
total=  13.8s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=  22.0s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=  21.2s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=  21.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=  21.1s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features=auto, max_depth=20,
total=  19.2s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=  20.8s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=  20.9s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=  19.9s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=  20.2s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features=sqrt, max_depth=25,
total=  16.6s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   6.6s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   7.6s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   6.8s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt,
max_depth=5, total=   7.3s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, max_depth=5
```

```
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_features=sqrt, 
max_depth=5, total=   7.0s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15 
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, 
total=   3.4s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15 
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, 
total=   3.9s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15 
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, 
total=   3.3s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15 
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, 
total=   3.2s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15 
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features=sqrt, max_depth=15, 
total=   3.2s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5 
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, 
total=   4.7s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5 
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, 
total=   4.5s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5 
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, 
total=   3.3s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5 
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, 
total=   3.2s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5 
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features=sqrt, max_depth=5, 
total=   3.4s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20 
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, 
total=  20.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20 
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, 
total=  18.6s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20 
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, 
total=  19.2s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20 
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, 
total=  18.9s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20 
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features=auto, max_depth=20, 
total=  18.7s
```

```
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  9.7min finished
```

Out[120]:

```
RandomizedSearchCV(cv=5, error_score=nan,
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   ccp_alpha=0.0,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   max_samples=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0.0,
                                                   n_estimators=100,
                                                   n_jobs=None, oob_score=Fals...
                   iid='deprecated', n_iter=10, n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                              1200]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [122]:

```
rf_random.best_params_
```

Out[122]:

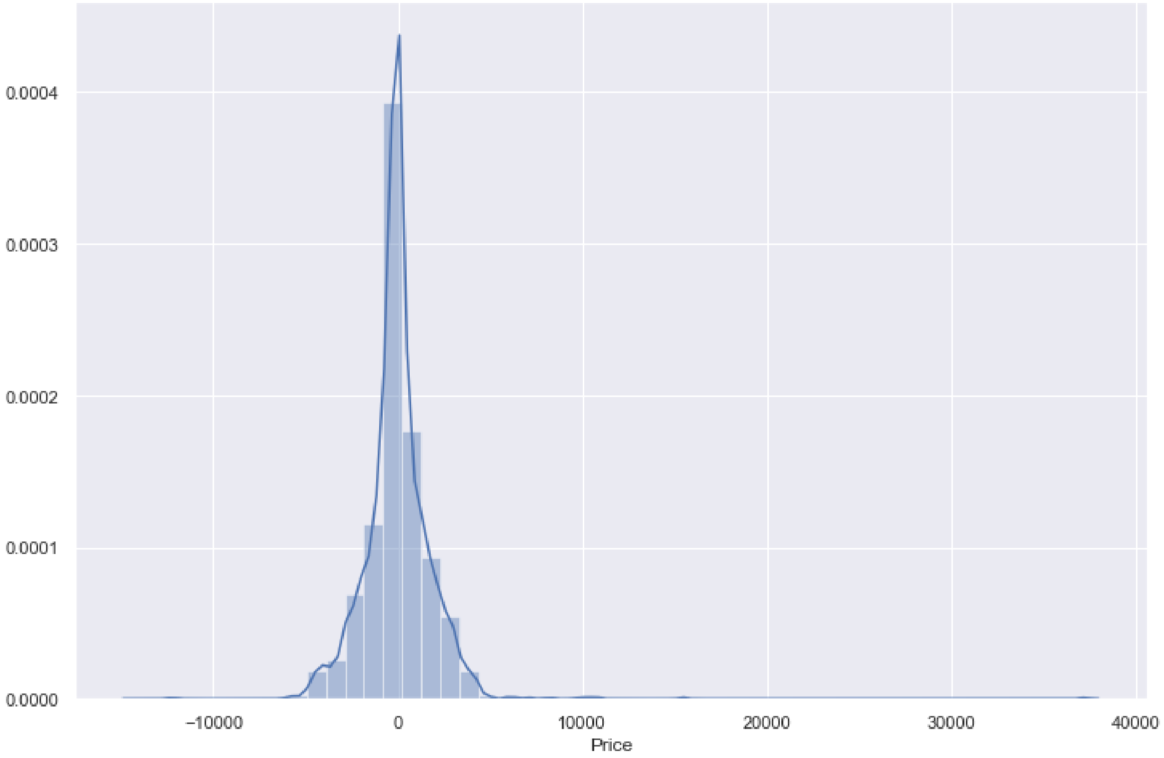
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

In [123]:

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

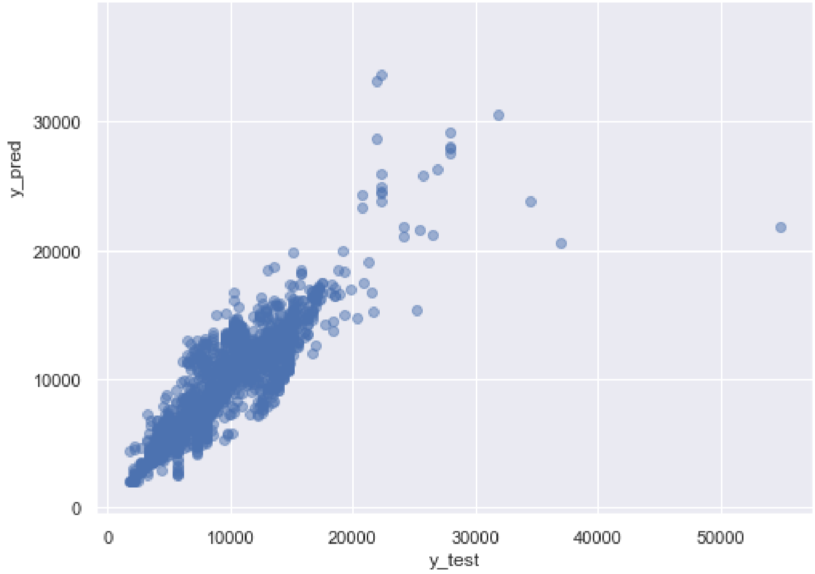
In [124]:

```
plt.figure(figsize=(12,8))
sns.distplot(y_test-prediction)
plt.show()
```



In [125]:

```
plt.figure(figsize=(8,8))
plt.scatter(y_test,y_pred,alpha=0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



In [127]:

```
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: 1175.3759459098128
MSE: 3635126.711854808
RMsE: 1906.6008265640735
```

In [133]:

```
import pickle

file = open('flight_rf.pkl','wb')

####### dump info to that file #########

pickle.dump(rf_random,file)
```

In [134]:

```
model = open('flight_rf.pkl','rb')
forest = pickle.load(model)
```

In [135]:

```
y_prediction = forest.predict(x_test)
```

In [136]:

```
metrics.r2_score(y_test,y_prediction)     ########## increasing R^2 value by using rf_random
```

Out[136]:

```
0.8227691686951393
```

In [ ]: