Flight Price Prediction

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

In [2]:
```
train_data = pd.read_excel(r'C:\Users\Arvind Kumar Yadav\Downloads\Data_Train.xlsx')
```

To display all the columns present in the dataset

In [3]:
```
pd.set_option('display.max_columns',None)
```

In [4]:
```
train_data.head()
```

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

In [5]:
```
1 train_data.shape
```

Out[5]:
```
(10683, 11)
```

In [6]:
```
1 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 [7]:
```
1 train_data['Duration'].value_counts()
```

Out[7]:
```
2h 50m     550
1h 30m     386
2h 45m     337
2h 55m     337
2h 35m     329
          ...
19h 50m      1
35h 35m      1
35h 20m      1
36h 25m      1
4h 10m       1
Name: Duration, Length: 368, dtype: int64
```

EDA

In [8]:
```
1 train_data.isnull().sum()
```

Out[8]:
```
Airline           0
Date_of_Journey   0
Source            0
Destination       0
Route             1
Dep_Time          0
Arrival_Time      0
Duration          0
Total_Stops       1
Additional_Info   0
Price             0
dtype: int64
```

In [9]:
```
1 train_data.dropna(inplace=True)
```

In [10]:
```
1 train_data.isnull().sum()
```

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]:
```
1 train_data['Journey_day'] = pd.to_datetime(train_data['Date_of_Journey'],format='%d/%m/%Y').dt.day
```

In [12]:
```
1 train_data['Journey_month'] = pd.to_datetime(train_data['Date_of_Journey'],format='%d/%m/%Y').dt.month
```

In [13]:
```
train_data.head()
```

Out[13]:

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journe
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 [14]:
```
# Since we have converted Date_of_Journey column into integers,Now we can drop as it is of no use.

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

In [15]:

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

# Extracting hours
train_data['Dep_hour'] = pd.to_datetime(train_data['Dep_Time']).dt.hour

# Extracting minutes
train_data['Dep_min'] = pd.to_datetime(train_data['Dep_Time']).dt.minute

# Now we can drop Dep_Time as it is of no use
train_data.drop(['Dep_Time'],axis=1,inplace=True)
```

In [16]:
```
1 train_data.head()
```

Out[16]:

	Airline	Source	Destination	Route	Arrival_Time	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_
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 [17]:

```
# Arrival time is when the plane pulls up to the gate.
# Similar to Date_of_journey we can extact values from Arrival_Time

# extracting hours
train_data['Arrival_hour'] = pd.to_datetime(train_data['Arrival_Time']).dt.hour

# extracting minutes
train_data['Arrival_min'] = pd.to_datetime(train_data['Arrival_Time']).dt.minute

# Now we can drop Arrival_Time as it is of no use
train_data.drop(['Arrival_Time'],axis=1,inplace=True)
```

In [18]:
```
1 train_data.head()
```

Out[18]:

	Airline	Source	Destination	Route	Duration	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_mir
0	IndiGo	Banglore	New Delhi	BLR → DEL	2h 50m	non-stop	No info	3897	24	3	22	20
1	Air India	Kolkata	Banglore	CCU → IXR → BBI → BLR	7h 25m	2 stops	No info	7662	1	5	5	50
2	Jet Airways	Delhi	Cochin	DEL → LKO → BOM → COK	19h	2 stops	No info	13882	9	6	9	25
3	IndiGo	Kolkata	Banglore	CCU → NAG → BLR	5h 25m	1 stop	No info	6218	12	5	18	5
4	IndiGo	Banglore	New Delhi	BLR → NAG → DEL	4h 45m	1 stop	No info	13302	1	3	16	50

In [19]:
```
# 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(train_data['Duration'])

for i in range(len(duration)):
    if len(duration[i].split()) != 2:
        if 'h' in duration[i]:
            duration[i] = duration[i].strip() + " 0m"
        else:
            duration[i] = "0h " + duration[i]

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 du
```

In [20]:
```
# Adding duration_hours and  duration_mins list to train_data dataframe

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

In [21]:
```
train_data.drop(['Duration'],axis=1,inplace=True)
```

In [22]:
```
1 train_data.head()
```

Out[22]:

	Airline	Source	Destination	Route	Total_Stops	Additional_Info	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_
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	

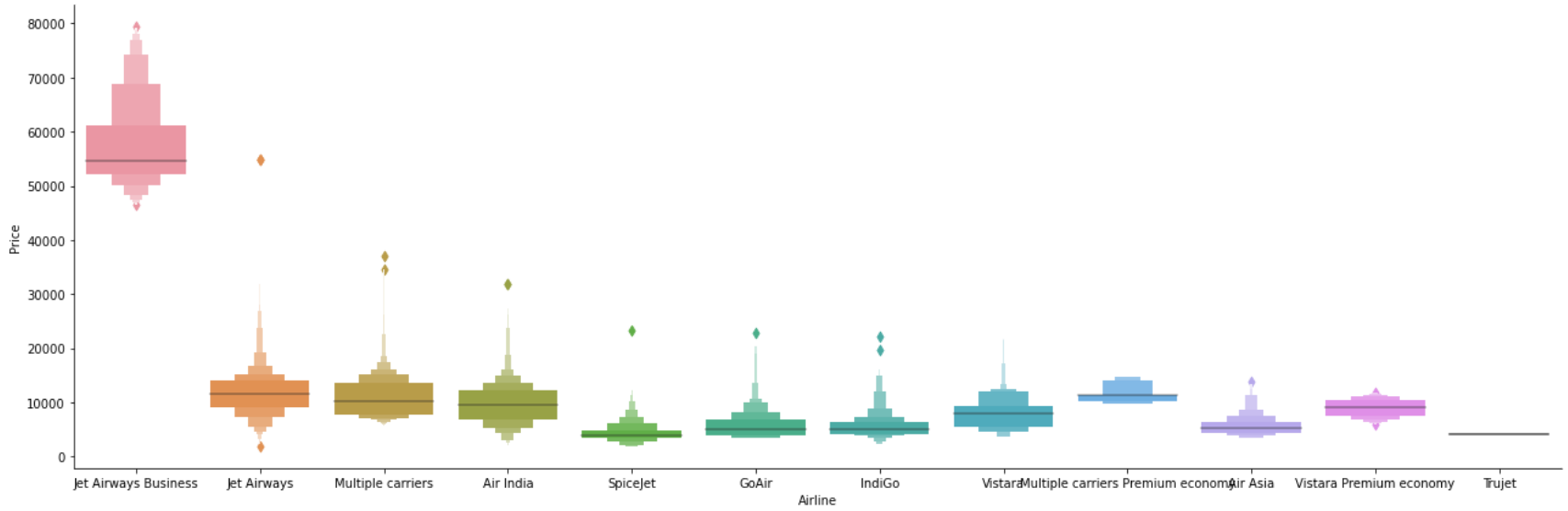
Handling Categorical Data

In [23]:
```
train_data['Airline'].value_counts()
```

Out[23]:
```
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 [24]:
```
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the 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),kind='boxen',height=
plt.show()
```



In [25]:
```
# As Airline is Nominal Categorical data we will perform OneHotEncoding

Airline = train_data[['Airline']]

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

Airline.head()
```

Out[25]:

	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	Airlin
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	

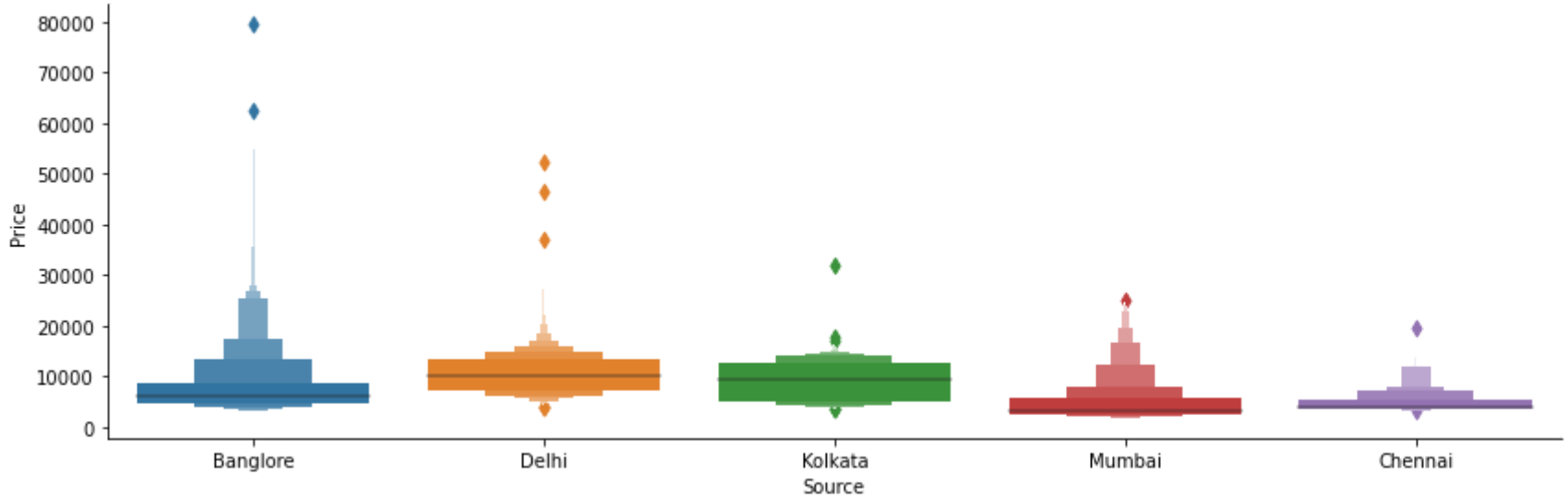
In [26]:
```
train_data['Source'].value_counts()
```

Out[26]:
```
Delhi       4536
Kolkata     2871
Banglore    2197
Mumbai       697
Chennai      381
Name: Source, dtype: int64
```

In [27]:
```
# Source Vs Price

sns.catplot(y='Price',x='Source',data=train_data.sort_values("Price",ascending=False),kind='boxen',height=4
```

Out[27]: <seaborn.axisgrid.FacetGrid at 0x24525af7310>



In [28]:
```
# As Source is Nominal Categorical data we will perform OneHotEncoding

Source = train_data['Source']

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

Source.head()
```

Out[28]:

	Chennai	Delhi	Kolkata	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 [29]:
```
train_data['Destination'].value_counts()
```

Out[29]:
```
Cochin       4536
Banglore     2871
Delhi        1265
New Delhi     932
Hyderabad     697
Kolkata       381
Name: Destination, dtype: int64
```

In [30]:
```
# As Destination is Nominal Categorical data we will perform OneHotEncoding

Destination = train_data[['Destination']]

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

Destination.head()
```

Out[30]:

	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 [31]:
```
train_data['Route']
```

Out[31]:
```
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 [32]:
```
1 train_data['Additional_Info'].value_counts()
```

Out[32]:
```
No info                         8344
In-flight meal not included     1982
No check-in baggage included     320
1 Long layover                    19
Change airports                    7
Business class                     4
No Info                            3
1 Short layover                    1
2 Long layover                     1
Red-eye flight                     1
Name: Additional_Info, dtype: int64
```

In [33]:
```
1 # Additional_Info contains almost 80% no_info
2 # Route and Total_stops are related to each other
3
4 train_data.drop(["Route","Additional_Info"],axis=1,inplace=True)
```

In [34]:
```
1 train_data['Total_Stops'].value_counts()
```

Out[34]:
```
1 stop      5625
non-stop    3491
2 stops     1520
3 stops       45
4 stops        1
Name: Total_Stops, dtype: int64
```

In [35]:
```
1 # As this is case of Ordinal Categorical type we will perform LabelEncoding
2 # Here values are assigned with corresponding keys
3
4 train_data.replace({'non-stop':0,'1 stop':1,'2 stops':2,'3 stops':3,'4 stops':4},inplace=True)
```

In [36]:
```
1 train_data.head()
```

Out[36]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Durat
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 [37]:
```
1 # Concatenate Dataframe ---> train_data + Airline + Source + Destination
2
3 data_train = pd.concat([train_data,Airline,Source,Destination],axis=1)
```

In [38]:
```
1 data_train.head()
```

Out[38]:

	Airline	Source	Destination	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Durat
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 [39]:
```
data_train.drop(['Airline','Source','Destination'],axis=1,inplace=True)
```

In [40]:
```
data_train.head()
```

Out[40]:

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airli
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 [41]:
```
data_train.shape
```

Out[41]:
```
(10682, 30)
```

Test set

In [42]:
```
test_data = pd.read_excel( r'C:\Users\Arvind Kumar Yadav\Downloads\Test_set.xlsx')
```

In [43]:
```
1 test_data.head()
```

Out[43]:

	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 [44]:

```
# Preprocessing

print("Test data Info")
print("-"*75)
print(test_data.info())

print()
print()

print("Null values :")
print("-"*75)
test_data.dropna(inplace = True)
print(test_data.isnull().sum())

# EDA

# Date_of_Journey
test_data["Journey_day"] = pd.to_datetime(test_data.Date_of_Journey, format="%d/%m/%Y").dt.day
test_data["Journey_month"] = pd.to_datetime(test_data["Date_of_Journey"], format = "%d/%m/%Y").dt.month
test_data.drop(["Date_of_Journey"], axis = 1, inplace = True)

# Dep_Time
test_data["Dep_hour"] = pd.to_datetime(test_data["Dep_Time"]).dt.hour
test_data["Dep_min"] = pd.to_datetime(test_data["Dep_Time"]).dt.minute
test_data.drop(["Dep_Time"], axis = 1, inplace = True)

# Arrival_Time
test_data["Arrival_hour"] = pd.to_datetime(test_data.Arrival_Time).dt.hour
test_data["Arrival_min"] = pd.to_datetime(test_data.Arrival_Time).dt.minute
test_data.drop(["Arrival_Time"], axis = 1, inplace = True)

# Duration
duration = list(test_data["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 du

# Adding Duration column to test set
test_data["Duration_hours"] = duration_hours
test_data["Duration_mins"] = duration_mins
test_data.drop(["Duration"], axis = 1, inplace = True)


# Categorical data

print("Airline")
print("-"*75)
print(test_data["Airline"].value_counts())
Airline = pd.get_dummies(test_data["Airline"], drop_first= True)

print()

print("Source")
print("-"*75)
print(test_data["Source"].value_counts())
Source = pd.get_dummies(test_data["Source"], drop_first= True)

print()

print("Destination")
print("-"*75)
print(test_data["Destination"].value_counts())
Destination = pd.get_dummies(test_data["Destination"], drop_first = True)

# Additional_Info contains almost 80% no_info
# Route and Total_Stops are related to each other
test_data.drop(["Route", "Additional_Info"], axis = 1, inplace = True)

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

# Concatenate dataframe --> test_data + Airline + Source + Destination
data_test = pd.concat([test_data, Airline, Source, Destination], axis = 1)

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

```
86
87 print()
88 print()
89
90 print("Shape of test data : ", data_test.shape)
```

```
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
None


Null values :
-----------------------------------------------------------------------------
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
Airline
-----------------------------------------------------------------------------
```

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

Source
------------------------------------------------------------------------------
Delhi       1145
Kolkata      710
Banglore     555
Mumbai       186
Chennai       75
Name: Source, dtype: int64

Destination
------------------------------------------------------------------------------
Cochin       1145
Banglore      710
Delhi         317
New Delhi     238
Hyderabad     186
Kolkata        75
Name: Destination, dtype: int64


Shape of test data :  (2671, 28)
```

In [45]:
```
1 data_test.head()
```

Out[45]:

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Air India	GoAi
0	1	6	6	17	30	4	25	10	55	0	(
1	1	12	5	6	20	10	20	4	0	0	(
2	1	21	5	19	15	19	0	23	45	0	(
3	1	21	5	8	0	21	0	13	0	0	(
4	0	24	6	23	55	2	45	2	50	0	(

Feature Selection

In [46]:
```
1 data_train.shape
```

Out[46]: (10682, 30)

In [47]:
```
1 data_train.columns
```

Out[47]:
```
Index(['Total_Stops', 'Price', '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',
       'Chennai', 'Delhi', 'Kolkata', 'Mumbai', 'Destination_Cochin',
       'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
       'Destination_New Delhi'],
      dtype='object')
```

In [48]:
```
1  X = data_train.loc[:,['Total_Stops','Journey_day', 'Journey_month', 'Dep_hour',
2        'Dep_min', 'Arrival_hour', 'Arrival_min', 'Duration_hours',
3        'Duration_mins', 'Airline_Air India', 'Airline_GoAir', 'Airline_IndiGo',
4        'Airline_Jet Airways', 'Airline_Jet Airways Business',
5        'Airline_Multiple carriers',
6        'Airline_Multiple carriers Premium economy', 'Airline_SpiceJet',
7        'Airline_Trujet', 'Airline_Vistara', 'Airline_Vistara Premium economy',
8        'Chennai', 'Delhi', 'Kolkata', 'Mumbai', 'Destination_Cochin',
9        'Destination_Delhi', 'Destination_Hyderabad', 'Destination_Kolkata',
10       'Destination_New Delhi']]
11 X.head()
```

Out[48]:

	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_Air India
0	0	24	3	22	20	1	10	2	50	0
1	2	1	5	5	50	13	15	7	25	1
2	2	9	6	9	25	4	25	19	0	0
3	1	12	5	18	5	23	30	5	25	0
4	1	1	3	16	50	21	35	4	45	0

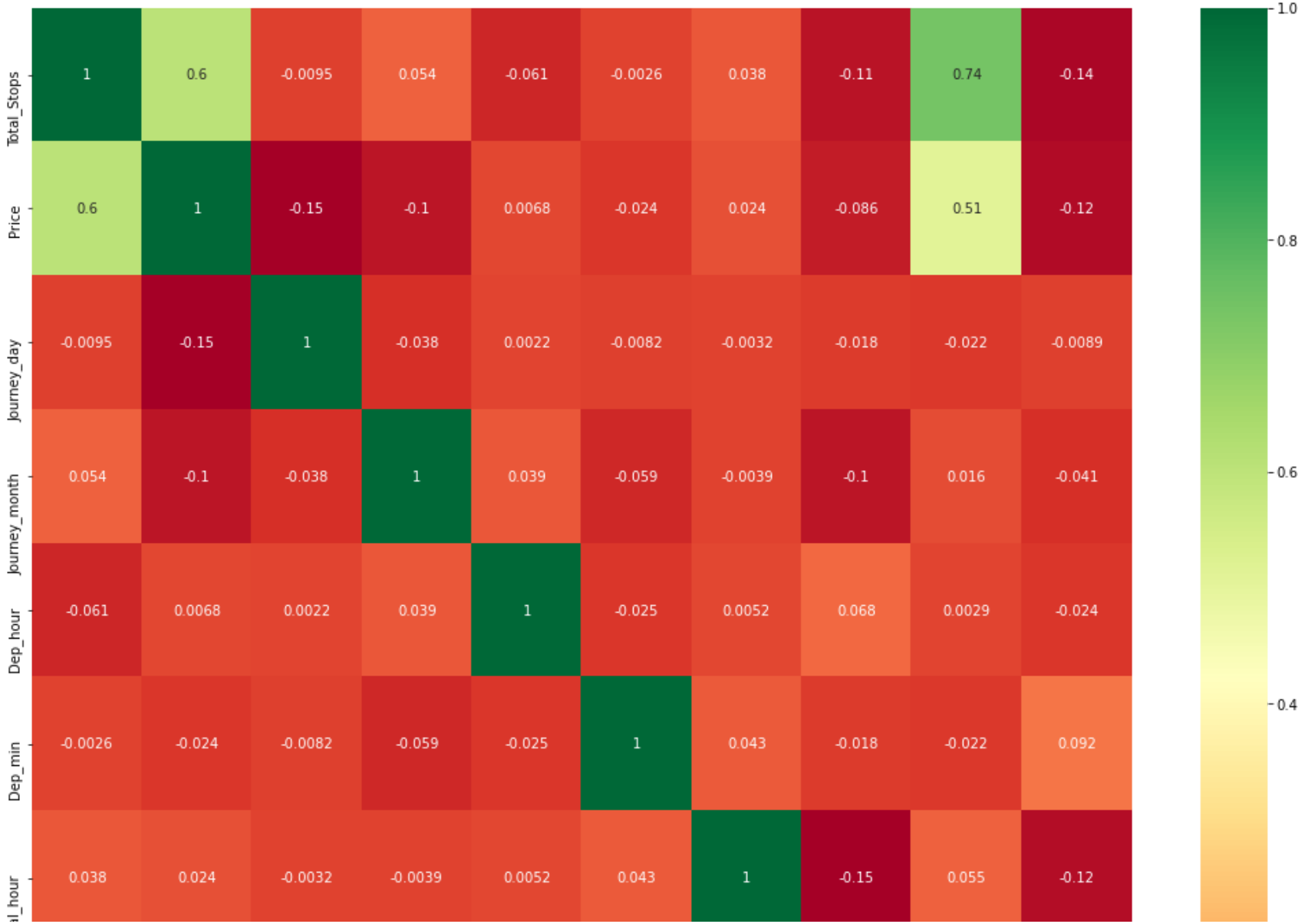
In [49]:
```
y = data_train.iloc[:,1]
y.head()
```

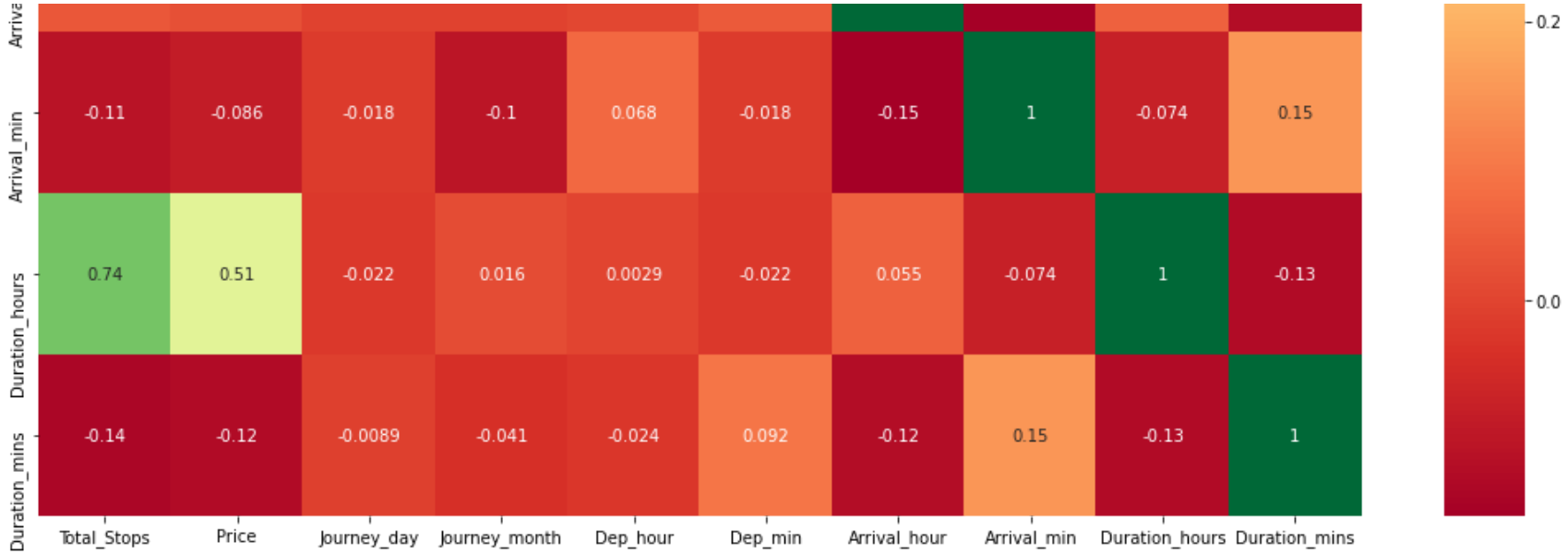
Out[49]:
```
0     3897
1     7662
2    13882
3     6218
4    13302
Name: Price, dtype: int64
```

In [50]:
```
# Finds correlation between Independent and Dependent attributes

plt.figure(figsize=(18,18))
sns.heatmap(train_data.corr(),annot=True,cmap="RdYlGn")

plt.show()
```





In [51]:
```
# Important Features using ExtraTreesRegressor

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

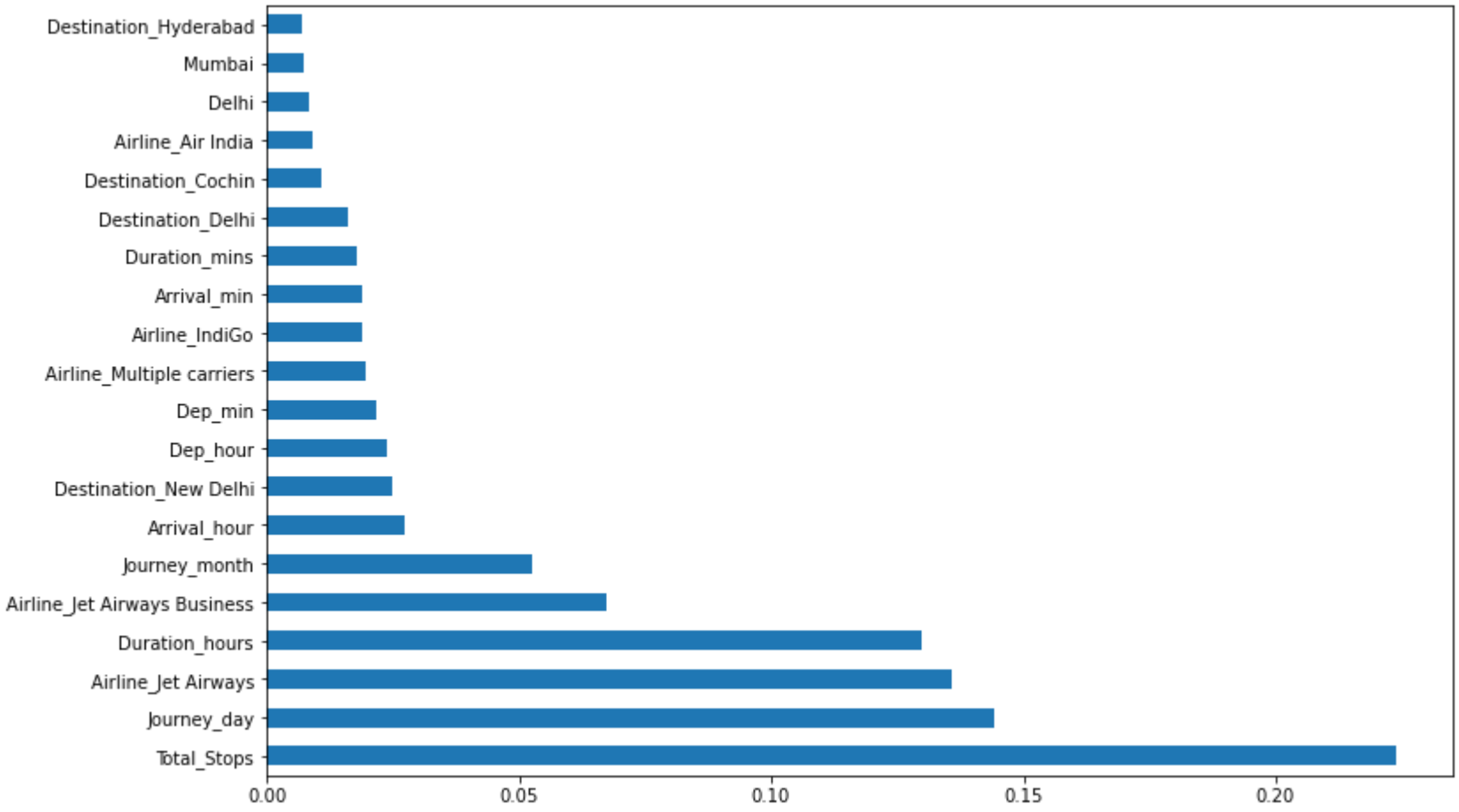
Out[51]:
```
ExtraTreesRegressor()
```

In [52]:
```
print(selection.feature_importances_)
```

```
[2.23774195e-01 1.44277805e-01 5.24489517e-02 2.37486704e-02
 2.16013069e-02 2.73877379e-02 1.88498364e-02 1.29725652e-01
 1.77354474e-02 9.05266256e-03 2.06802237e-03 1.90067438e-02
 1.35740789e-01 6.71819382e-02 1.94676872e-02 7.74488089e-04
 3.21541118e-03 1.00092280e-04 4.97591756e-03 8.47484169e-05
 5.29971295e-04 8.42282980e-03 3.45081618e-03 7.16828037e-03
 1.07084872e-02 1.60306621e-02 6.91204514e-03 5.12675997e-04
 2.50461290e-02]
```

In [53]:
```
# plot graph of feature importances 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()
```



Fitting model using Random Forest

In [54]:
```
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)
```

In [55]:
```
from sklearn.ensemble import RandomForestRegressor
reg_rf = RandomForestRegressor()
reg_rf.fit(X_train,y_train)
```

Out[55]:
```
RandomForestRegressor()
```

In [56]:
```
y_pred = reg_rf.predict(X_test)
```

In [57]:
```
y_pred
```

Out[57]:
```
array([16786.01      ,  5493.7       ,  8911.36      , ...,
        6540.9       , 12763.02416667, 12888.425     ])
```

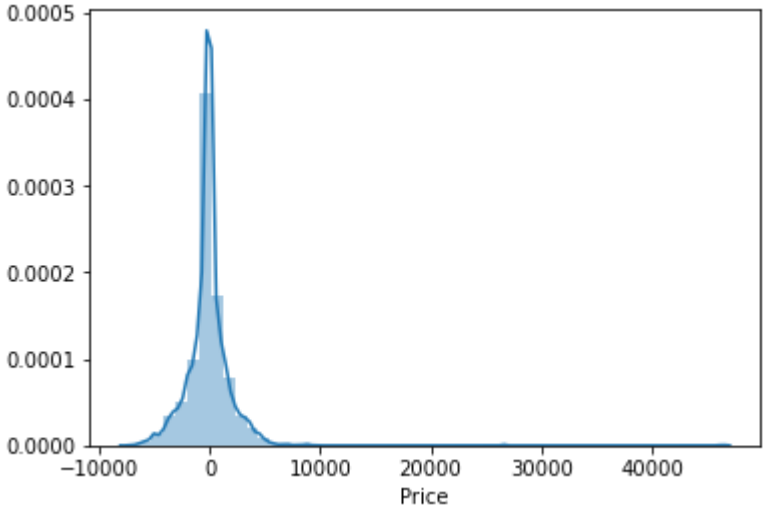
In [59]:
```
reg_rf.score(X_train,y_train)
```

Out[59]:
```
0.9527308946256529
```

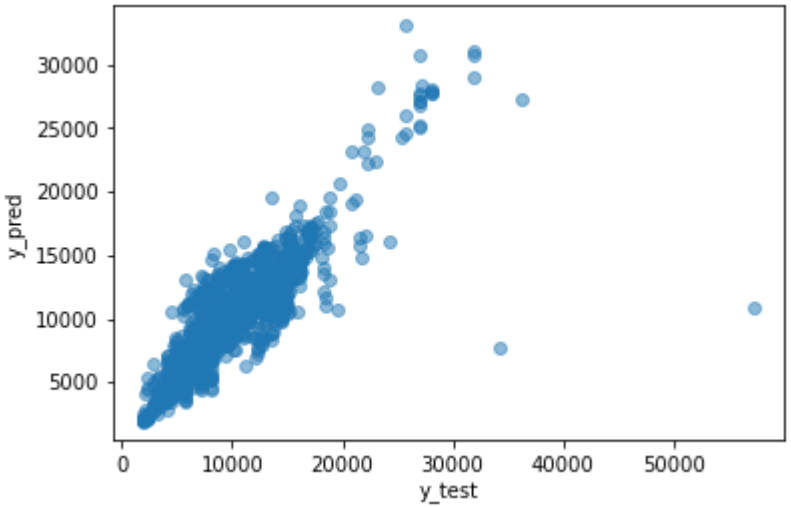
In [60]:
```
reg_rf.score(X_test,y_test)
```

Out[60]:
```
0.7963386583419167
```

In [61]:
```
1 sns.distplot(y_test-y_pred)
2 plt.show()
```



In [62]:
```
plt.scatter(y_test,y_pred,alpha=0.5)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



In [63]:
```
from sklearn import metrics
```

In [64]:
```
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: 1176.8080878572111
MSE: 4391357.319885949
RMSE: 2095.5565656612443
```

In [65]:
```
metrics.r2_score(y_test,y_pred)
```

Out[65]:
```
0.7963386583419167
```

Hyperparameter Tuning

In [66]:
```
from sklearn.model_selection import RandomizedSearchCV
```

In [67]:
```
# Numbers of trees in random forest
n_estimators = [int(x) for x in np.linspace(start=100,stop=1200,num=12)]
# Numbers 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]
```

In [68]:
```
# 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}
```

In [69]:
```
# 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_e
```

In [70]:
```
1 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 t
ime=   6.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total t
ime=   7.1s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total t
ime=   7.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total t
ime=   7.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total t
ime=   7.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total
time=  11.3s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total
time=  11.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total
time=  10.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total
time=  10.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total
time=  10.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total
time=   6.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total
time=   6.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total
time=   6.6s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total
time=   6.8s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total
time=   6.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total t
ime=  12.4s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total t
ime=  12.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total t
ime=  11.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total t
ime=  12.5s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total t
```

```
ime=  12.2s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total
time=  20.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total
time=  21.4s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total
time=  21.1s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total
time=  21.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=10, min_samples_split=5, n_estimators=700; total
time=  19.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total
time=  20.8s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total
time=  20.3s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total
time=  20.8s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total
time=  20.0s
[CV] END max_depth=25, max_features=sqrt, min_samples_leaf=1, min_samples_split=2, n_estimators=1000; total
time=  20.4s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total
time=   7.6s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total
time=   7.7s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total
time=   8.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total
time=   6.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=10, min_samples_split=15, n_estimators=1100; total
time=   6.0s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total
time=   2.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total
time=   2.9s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total
time=   2.7s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total
time=   2.8s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=1, min_samples_split=15, n_estimators=300; total
time=   2.8s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total t
ime=   3.8s
```

```
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total t
ime=   3.7s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total t
ime=   5.3s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total t
ime=   5.0s
[CV] END max_depth=5, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=700; total t
ime=   4.9s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total
time=  28.0s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total
time=  26.6s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total
time=  25.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total
time=  27.7s
[CV] END max_depth=20, max_features=auto, min_samples_leaf=1, min_samples_split=15, n_estimators=700; total
time=  26.5s
```

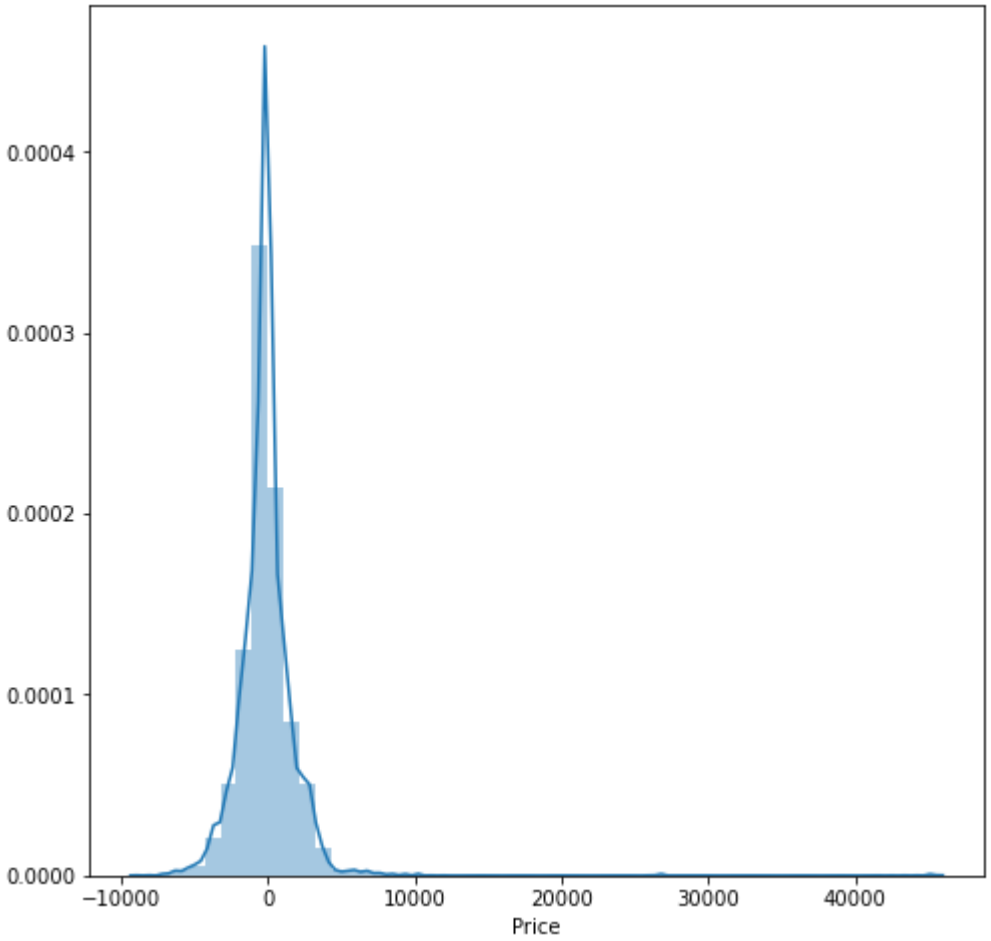
Out[70]:
```
RandomizedSearchCV(cv=5, estimator=RandomForestRegressor(), n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
                                                         900, 1000, 1100,
                                                         1200]},
                   random_state=42, scoring='neg_mean_squared_error',
                   verbose=2)
```

In [71]:
```
1 rf_random.best_params_
```

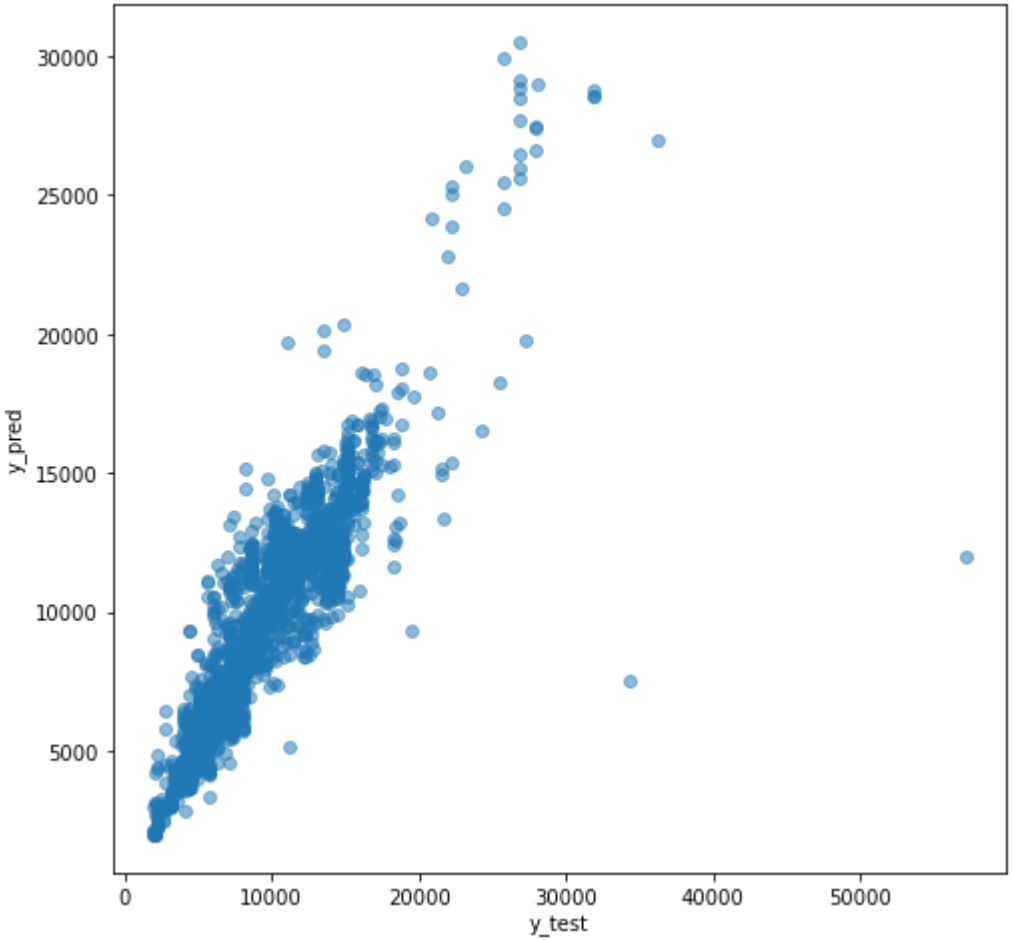
Out[71]:
```
{'n_estimators': 700,
 'min_samples_split': 15,
 'min_samples_leaf': 1,
 'max_features': 'auto',
 'max_depth': 20}
```

In [72]:
```
1 prediction = rf_random.predict(X_test)
```

In [73]:
```
plt.figure(figsize=(8,8))
sns.distplot(y_test-prediction)
plt.show()
```



In [74]:
```
plt.figure(figsize=(8,8))
plt.scatter(y_test,prediction,alpha=0.5)
plt.xlabel('y_test')
plt.ylabel('y_pred')
plt.show()
```



In [75]:
```
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: 1164.8105637219141
MSE: 4050406.8597088386
RMSE: 2012.5622623185695
```

In [82]:
```
y_prediction = rf_random.predict(X_test)
```

In [83]:
```
metrics.r2_score(y_test,y_prediction)
```

Out[83]:
```
0.8121511789592128
```

In [ ]:
```

```