Flight Price Prediction

Nowadays, it is hard to predict flight prices due to given various condition that are current at that particular time since Airline companies used complex algorithm to calculate flight prices.
Today, the amount of people travelling by flight has increased outstandingly.
This has made it even tougher for airlines to maintain their prices as prices change dramatically due to distinct conditions.

With this, I will attempt to use machine learning algorithms to solve this problem. This can go a long way to aid airlines by predicting what prices can be preserve and can further help clients to predict upcoming flight prices and plan their trip properly without fear.

Data Analysis

The data is collected from India Airline Companies where have independents variable which influencing the flight ticket price and it available

on GitHub for data scientists and machine learning enthusiasts on the link: https://github.com/dsrscientist/Data-Science-ML-Capstone-Projects

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

```
In [2]: data = pd.read_excel('Data_Train.xlsx')
```

```
In [3]: data.describe()
```

Out[3]:

	Price
count	10683.000000
mean	9087.064121
std	4611.359167
min	1759.000000
25%	5277.000000
50%	8372.000000
75%	12373.000000
max	79512.000000

from the above we can see statistical distribut
that it is only column which has numerical dat

As we can see, the data is loaded using those libraries and by using the describe() method

we can see only the price attribute our target variable has numerical data and this mean our attribute(columns) have text data.

Null values checking*: The null values or missing values (NAN) is something frequently found in the raw data.*
The human errors are common and also something when we are asked to fill out a form many of us will leave
some fields empty and this is where the missing values is general generated and they can be treated in several ways
but should our many is present in our dataset.

```
data.isna().sum()
```

```
Airline            0
Date_of_Journey    0
Duration           0
Total_Stops        1
Price              0
dtype: int64
```

The above code show again clearly that there is
"Total_Stops" column. and we will fill it by the m

Checking for unwanted attributes columns: *Sometime when you start a project which consist to build a model to predict an outcome*
you need to some basic idea about which data and good data your need to overcome because you can use any data you want but using
the right data is the first step to predict something right.

```
## This will help us to see how is the shape of

for column in data.columns:
    print(data[column].value_counts())
    print('**'*20)
```

```
Jet Airways                          3700
IndiGo                               2043
Air India                            1695
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
****************************************
```

```
****************************************
2h 50m     544
1h 30m     386
2h 45m     335
2h 55m     332
2h 35m     329
           ...
30h 25m      1
30h 15m      1
42h 5m       1
28h 55m      1
47h 40m      1
Name: Duration, Length: 368, dtype: int
****************************************
1 stop      5625
non-stop    3475
2 stops     1318
3 stops       43
4 stops        1
Name: Total_Stops, dtype: int64
****************************************
10262    258
10844    212
7229     161
4804     160
4823     131
         ...
8925       1
11774      1
16497      1
8853       1
12648      1
Name: Price, Length: 1870, dtype: int64
****************************************
```

EDA (Exploratory Data Analysis) and Pre-Processing Pipeline.

```
## fill the missing values
data["Total_Stops"] = data["Total_Stops"].fillna('1
```

'Duration' let's split it and get the minutes

```
data["Duration"].head(4)
```

```
0     2h 50m
1     7h 25m
2        19h
3     5h 25m
Name: Duration, dtype: object
```

```
items  = [x.split(' ') for x in data["Duration"]]
```

```
for x in items:
    if len(x) ==1:
        x.append('0m')
```

```
items[:5]
```

```
[['2h', '50m'], ['7h', '25m'], ['19h', '0m'], ['5h',
```

```
## Now will seperate minutes and hours

Duration_hours = []
for x in items:
    Duration_hours.append(x[0])
```

```
## geting the minutes

Duration_minutes = []
for x in items:
    Duration_minutes.append(x[1])
```

```
Duration_hours[:5]
```

```
['2h', '7h', '19h', '5h', '4h']
```

```
Duration_minutes[:5]
```

```
['50m', '25m', '0m', '25m', '45m']
```

Let's let's remove the character 'h' and "m" Using r

```
## let's import regular expression

import re
```

```
## now let's remove the "h" and "m" characters

Hours = list()
for x in Duration_hours:
    Hours.append(re.sub("[h]","",x))
```

```
Minutes = list()
for x in Duration_minutes:
    Minutes.append(re.sub("[m]","",x))
```

```
## checking the length

print(len(Hours),len(Minutes))
```

```
10463 10463
```

```
for x in Duration_hours:
    if 'm' in x:
        print(x)
```

```
5m
```

we can see that is character "m" for on item. maybe in place of 'h', so let's solve

```
### This gives no output and it means the Duration_m

for x in Duration_minutes:
    if 'h' in x:
        print(x)
```

```
data['Duration_hours'] = Hours
data['Duration_minutes'] = Minutes
```

```
## let's solve the single "m" problem
data['Duration_hours'] =data['Duration_hours'].replac
```

```
print(data.dtypes)
```

```
Airline              object
Date_of_Journey      object
Duration             object
Total_Stops          object
Price                 int64
Duration_hours       object
Duration_minutes     object
dtype: object
```

we can see that the two new columns 'Duration_hou object type so let's convert them into dtypes ("int")

```
data['Duration_hours'] = data['Duration_hours'].astyp
data['Duration_minutes'] = data['Duration_minutes'].a
```

```
print(data.dtypes)
```

```
Airline              object
Date_of_Journey      object
Duration             object
Total_Stops          object
Price                 int64
Duration_hours        int64
Duration_minutes      int64
dtype: object
```

I have converted successfull the both columns fron
Great.

:
```
data.head()
```

:

	Airline	Date_of_Journey	Duration	Total_Stops	Price	D
0	IndiGo	24/03/2019	2h 50m	non-stop	3897	2
1	Air India	1/05/2019	7h 25m	2 stops	7662	7
2	Jet Airways	9/06/2019	19h	2 stops	13882	1
3	IndiGo	12/05/2019	5h 25m	1 stop	6218	5
4	IndiGo	01/03/2019	4h 45m	1 stop	13302	4

:
```
data.drop('Duration',axis=1,inplace=True)
```

:
```
## let's verify it.

data.head(2)
```

:

	Airline	Date_of_Journey	Total_Stops	Price	Duration_hours
0	IndiGo	24/03/2019	non-stop	3897	2
1	Air India	1/05/2019	2 stops	7662	7

"Date_of_Journey" , let's convert it into "year", "mo

:
```
data['Date_of_Journey'].head(2)
```

:
```
0    24/03/2019
1     1/05/2019
Name: Date_of_Journey, dtype: object
```

:
```
data['year'] = pd.DatetimeIndex(data['Date_of_Journe
data['month'] = pd.DatetimeIndex(data['Date_of_Jourr
data['day'] = pd.DatetimeIndex(data['Date_of_Journey
```

```
## let's verify it.

data['year'].head(3)
```

```
0    2019
1    2019
2    2019
Name: year, dtype: int64
```

```
## we can drop the"Date_of_Journey" column.

data.drop('Date_of_Journey',axis=1,inplace=True)
```

```
## let's verify it.
data.head(2)
```

	Airline	Total_Stops	Price	Duration_hours	Duration_minutes
0	IndiGo	non-stop	3897	2	50
1	Air India	2 stops	7662	7	25

"Airline" column. let's Encoder it using OneHoteEnc

```
data.Airline.unique()
```

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJ
       'Multiple carriers', 'GoAir', 'Vistara', 'Air
       'Vistara Premium economy', 'Jet Airways Busin
       'Multiple carriers Premium economy', 'Trujet'
```

```
### import libraries to Encoder the Airline data


from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import make_column_transformer
```

```
print(data['Airline'].value_counts())
print('**'*20)
```

```
Jet Airways                          3700
IndiGo                               2043
Air India                            1695
Multiple carriers                    1196
SpiceJet                              815
Vistara                               478
Air Asia                              319
GoAir                                 194
Multiple carriers Premium economy      13
Jet Airways Business                    6
Vistara Premium economy                 3
Trujet                                  1
Name: Airline, dtype: int64
****************************************
```

I will reduce the number classes and categorized th

```
data['Airline'] = data['Airline'].replace({'Multiple
arriers','Jet Airways Business':'Jet Airways','Vista
t':'Jet Airways'})
```

```
print(data['Airline'].value_counts())
print('**'*20)
```

```
Jet Airways                3707
IndiGo                     2043
Air India                  1695
Multiple carriers          1209
SpiceJet                    815
Vistara                     481
Air Asia                    319
GoAir                       194
Name: Airline, dtype: int64
****************************************
```

```
data.Airline.unique()
```

```
array(['IndiGo', 'Air India', 'Jet Airways', 'SpiceJ
       'Multiple carriers', 'GoAir', 'Vistara', 'Air
```

Now we can see that the Airline have only 8 classes

```
### Instantiate the method
ohe = OneHotEncoder()

ct = make_column_transformer(
(ohe,['Airline']),remainder='drop')
```

```
transformed_data = ct.fit_transform(data)
```

```
transformed_data[:,0]
```

```
<10463x1 sparse matrix of type '<class 'numpy.float6
        with 319 stored elements in Compressed Spars
```

```
transformed_data.shape
```

```
(10463, 8)
```

```
Indigo= transformed_data[:,0]
AirIndia =  transformed_data[:,1]
Jet_Airways =  transformed_data[:,2]
SpiceJet = transformed_data[:,3]
Multiple_carriers = transformed_data[:,4]
GoAir = transformed_data[:,5]
Vistara = transformed_data[:,6]
Air_Asia =transformed_data[:,7]
```

```
### let's create new columns with each airline


data['Indigo']       =    Indigo.getnnz(1)
data['Air India']    =    AirIndia.getnnz(1)
data['Jet Airways'] = Jet_Airways.getnnz(1)
data['SpiceJet']    =SpiceJet.getnnz(1)
data['Multiple carriers']=Multiple_carriers.getnnz(1
data['GoAir']        =GoAir.getnnz(1)
data['Vistara']      =Vistara.getnnz(1)
data['Air Asia']     =Air_Asia.getnnz(1)
```

```
## let's drop Airline column

data.drop('Airline',axis=1,inplace=True)
```

```
data.head(2)
```

	Total_Stops	Price	Duration_hours	Duration_minutes	year	m
0	non-stop	3897	2	50	2019	3
1	2 stops	7662	7	25	2019	1

Now let's Encoder again the "Total_Stops'

```
data.Total_Stops.unique()
```

```
array(['non-stop', '2 stops', '1 stop', '3 stops', '
       dtype=object)
```

Now let's Encoder them, from my persona are scable or measurable, because flight t always high pricing than with "stop".

```
data["Total_Stops"] = data["Total_Stops"].replace({'
stop':3,'non-stop':4})
```

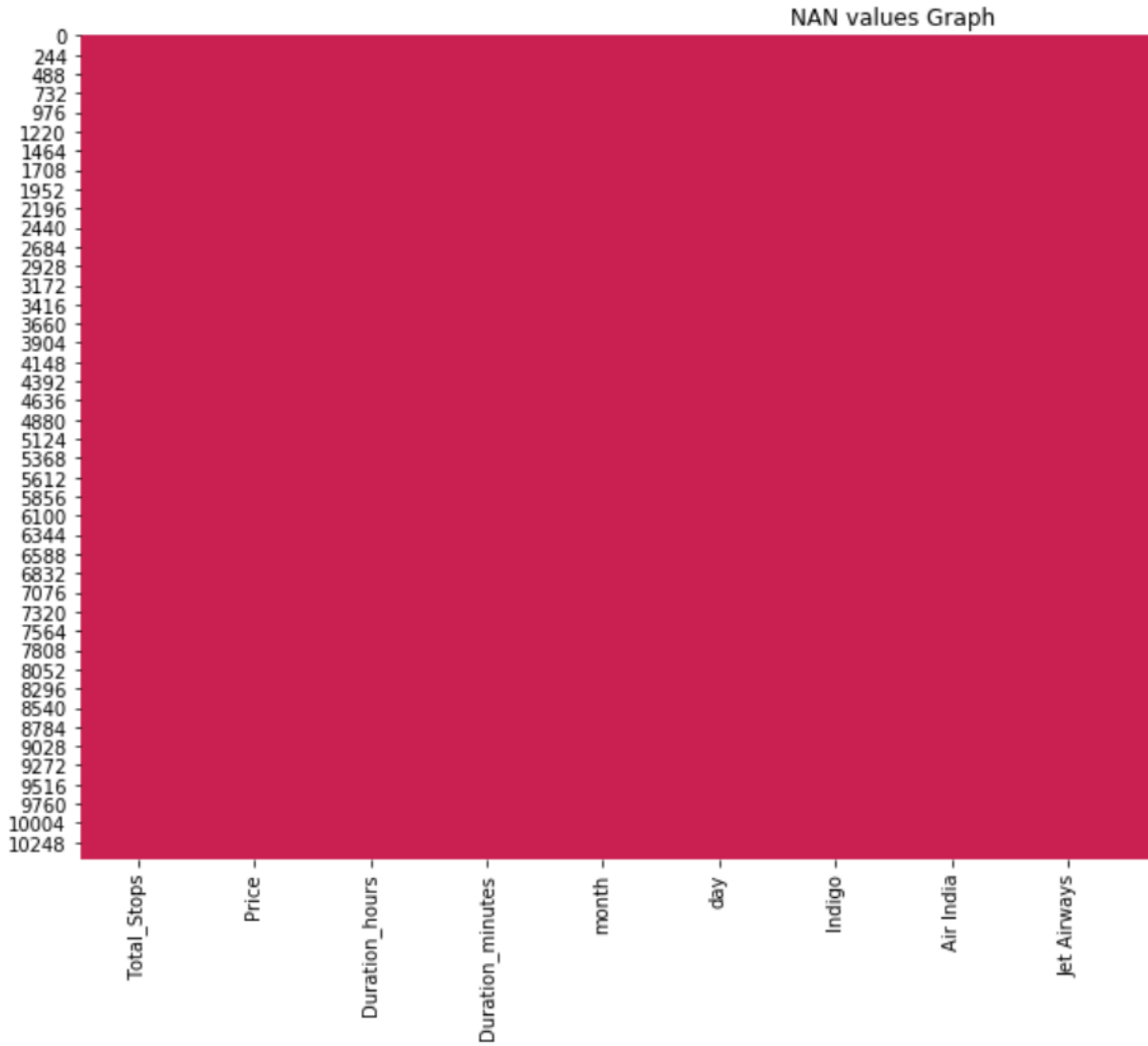
```
data.head(3)
```

	Total_Stops	Price	Duration_hours	Duration_minutes	year	m
0	4	3897	2	50	2019	3
1	2	7662	7	25	2019	1
2	2	13882	19	0	2019	9

From here we got our data clean and it contain only numerical values.

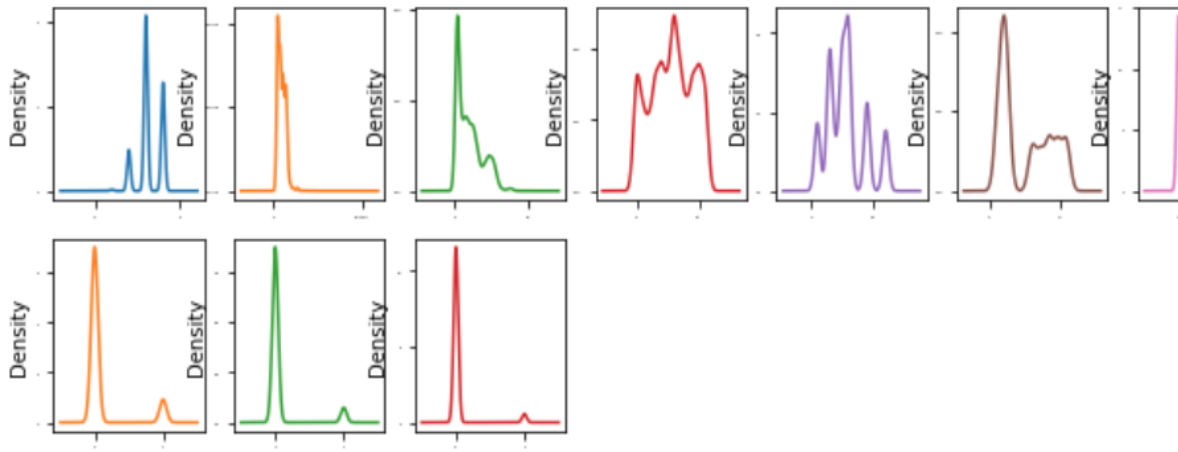
Now it time to do some data visualization and check our data shape and it distribution using graphs.

```
plt.figure(figsize=(16,8))
sns.heatmap(new_data.isna())
plt.title("NAN values Graph")
plt.tight_layout()
```



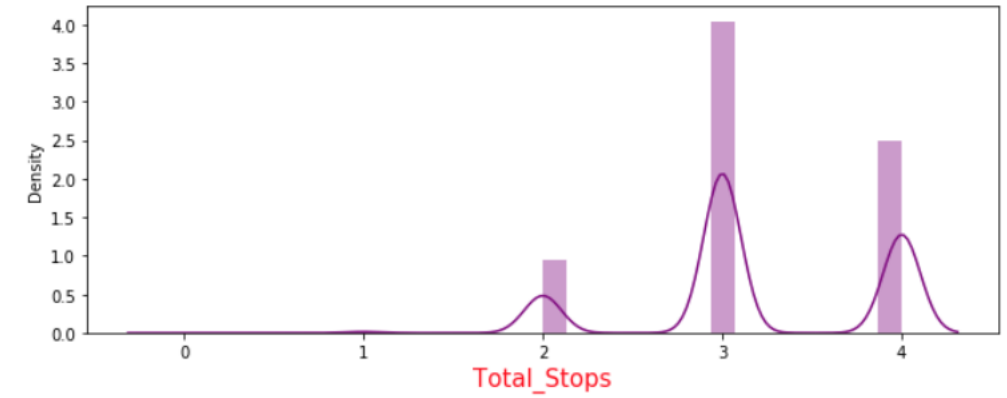
All the surface of the heatmap is red and this means dataset. great.

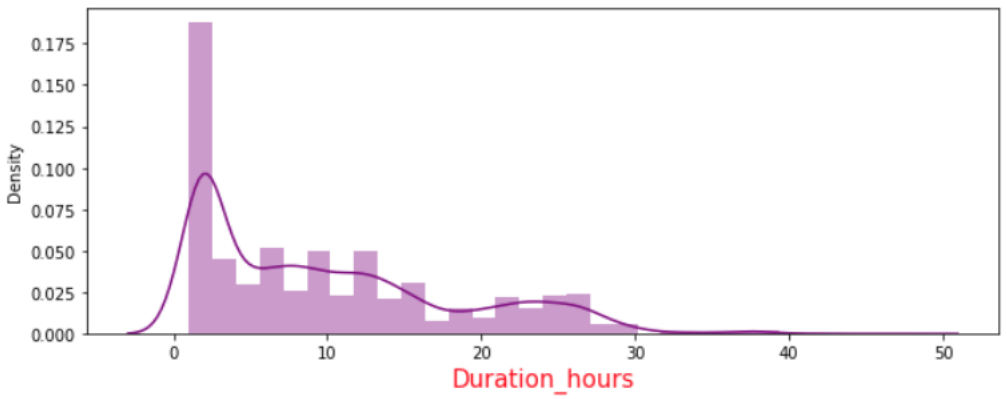
```
new_data.plot(kind='density',subplots=True,
              layout=(6,11),sharex=False,
              legend=False,fontsize='2',
              figsize =(18,12))
plt.show()
```

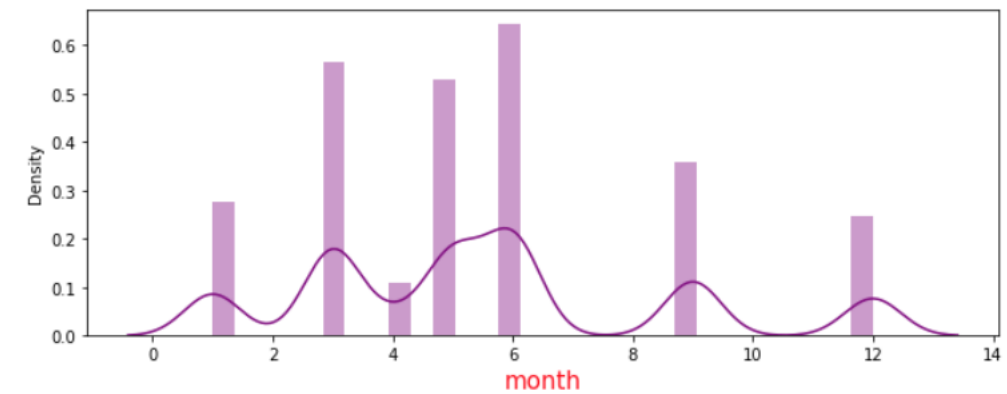


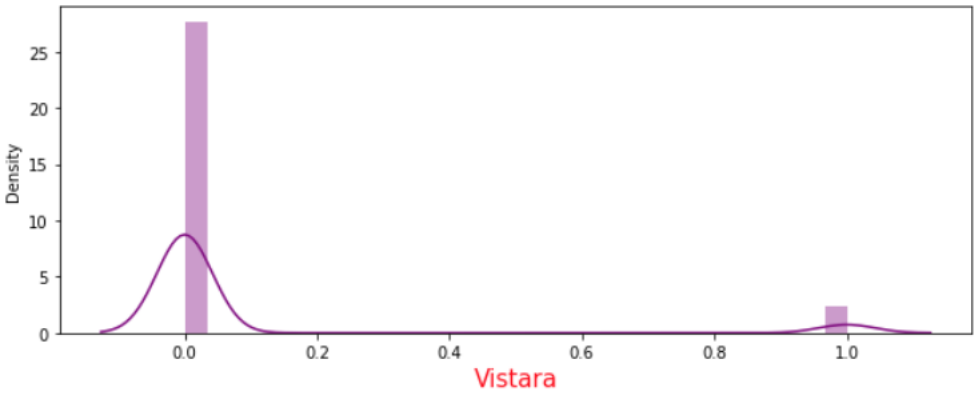
From this density graph we can see that most the co distribution shape.

```
# plot a  distplot check the distribution

plt.figure(figsize=(18,25))
plotnumber = 1

for column in new_data:
    if plotnumber <= 14:
        ax = plt.subplot(7,2,plotnumber)
        sns.distplot(new_data[column],bins=30,color='
        plt.xlabel(column,fontsize = 15,color='red')
    plotnumber+=1
plt.tight_layout()
```



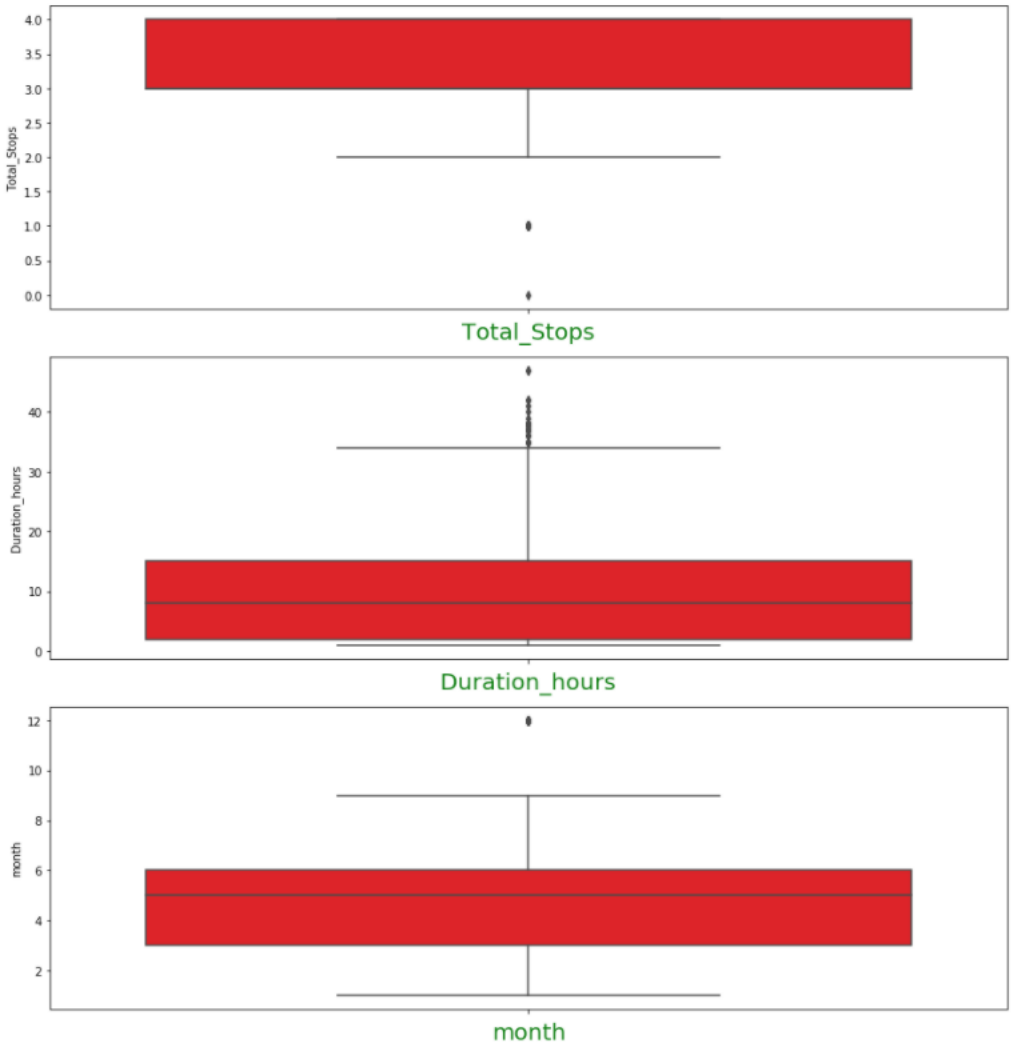


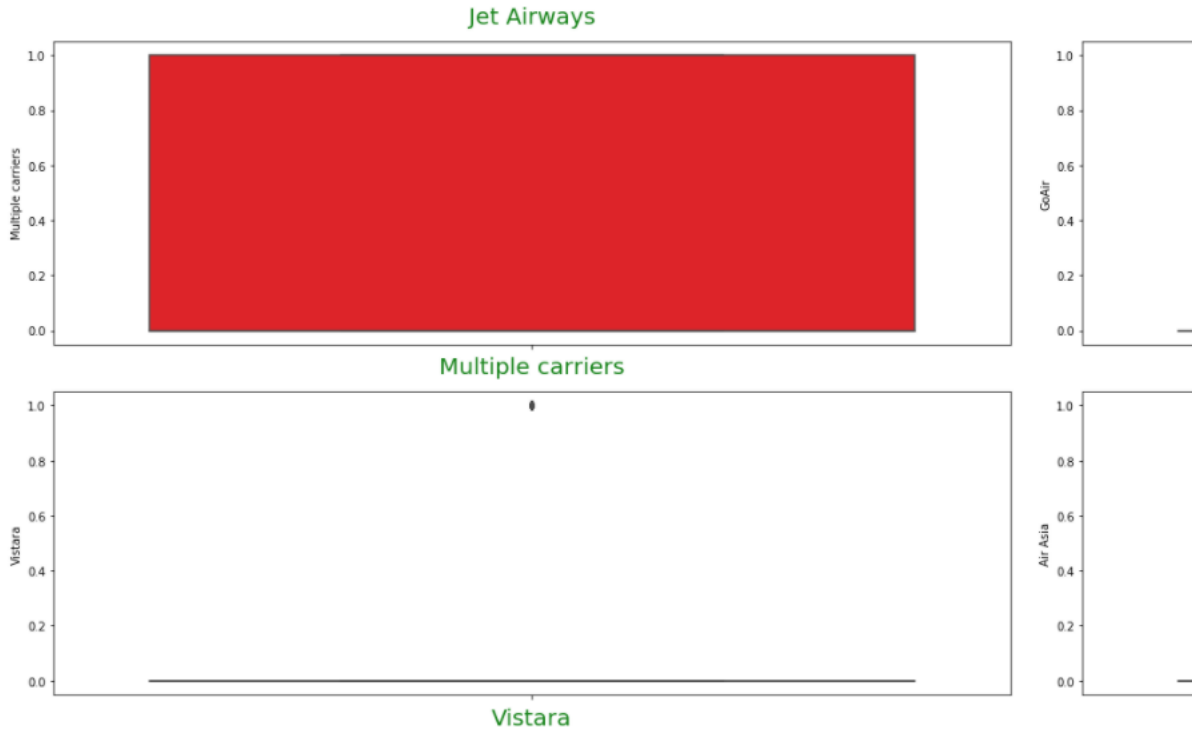




From distplot we can see that our data is not distrib
columns since we can see a lot of skewness.

```
plt.figure(figsize=(25,30))
plotnumber = 1

for column in new_data:
    if plotnumber <= 14:
        ax = plt.subplot(7,2,plotnumber)
        sns.boxplot(y= new_data[column],color='red',c
        plt.xlabel(column,fontsize = 20,color='green'
    plotnumber+=1
plt.tight_layout()
```





From the above boxplot graph we can see outliers in
Outliers the columns.

Let's use the Quantiles techniques and see if we can remove them

Outlier Dectection Formula ¶

Higher side ===> Q3 + (1.5 * IQR)

lower side ===> Q1 - (1.5 * IQR)

IQR = Inter Quantile Rate = Q3 - Q1

:
```
## Find the IQR ( Inter Quantile Range) to identify

# 1st quantile
q1 = new_data.quantile(0.25)

# 3rd quantile
q3 = new_data.quantile(0.75)

#IQR
iqr = q3 - q1
```

Will remove the outliers from the columns :"Total_St "month"

Removing the Outliers higher side

:
```
high_Price = (q3.Price + (1.5 * iqr.Price))
high_Price
```

:
```
23022.75
```

This means from the column "Price" all the records 23022.75) are Outliers.

```
## the get indexes of those records
indexes = np.where(new_data['Price'] > high_Price)
indexes
```

```
(array([  123,   396,   486,   510,   597,   628,
          935,   945,   958,   974,  1194,  1244,  1
         1474,  1625,  1650,  1778,  1909,  2044,  2
         2543,  2604,  2621,  2677,  2904,  3010,  3
         3505,  3667,  3974,  4476,  4779,  4960,  5
         5597,  5636,  5645,  5654,  5673,  5680,  5
         6332,  6497,  6509,  6526,  6902,  7254,  7
         7455,  7516,  7609,  7620,  7650,  7795,  7
         8345,  8409,  8467,  8798,  8815,  8846,  8
         9457,  9538,  9787,  9864,  9922,  9967,  9
        10160, 10179, 10232, 10302]),)
```

```
## Let's drop indexes from the dataset

new_data= new_data.drop(new_data.index[indexes])
```

```
## reset the index
new_data.reset_index(drop=True)
```

	Total_Stops	Price	Duration_hours	Duration_minutes	m
0	4	3897	2	50	3
1	2	7662	7	25	1
2	2	13882	19	0	9
3	3	6218	5	25	12
4	3	13302	4	45	1
...	...	...	...	...	...
10364	4	4107	2	30	9
10365	4	4145	2	35	4
10366	4	7229	3	0	4
10367	4	12648	2	40	1
10368	2	11753	8	20	9

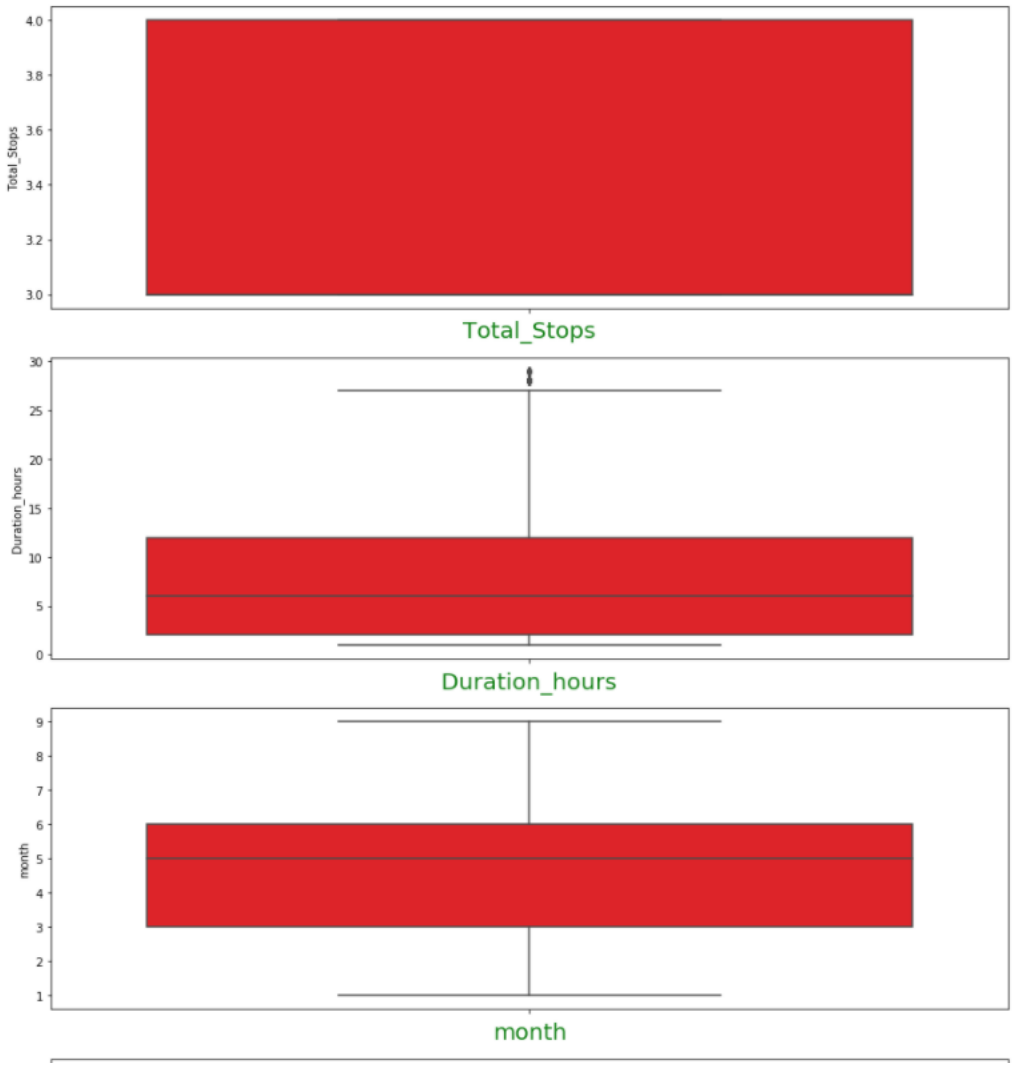
10369 rows × 14 columns

```
## "Duration_hours",column

indexes = np.where(new_data["Duration_hours"] > (q3.
ours)))
new_data= new_data.drop(new_data.index[indexes])
```

```
new_data.reset_index(drop=True)
```

	Total_Stops	Price	Duration_hours	Duration_minutes	m
0	4	3897	2	50	3
1	2	7662	7	25	1
2	2	13882	19	0	9
3	3	6218	5	25	12
4	3	13302	4	45	1
...	...	...	...	...	...
10298	4	4107	2	30	9
10299	4	4145	2	35	4
10300	4	7229	3	0	4
10301	4	12648	2	40	1
10302	2	11753	8	20	9

10303 rows × 14 columns



From the boxplot here we can say that our data loo
previous outliers.

I will remove sknewness from features col

skewness removal

```
## importing the libraries
from sklearn.preprocessing import PowerTransformer
```

```
pt=PowerTransformer(method='yeo-johnson')
```

```
features = new_data.drop('Price',axis=1)
```

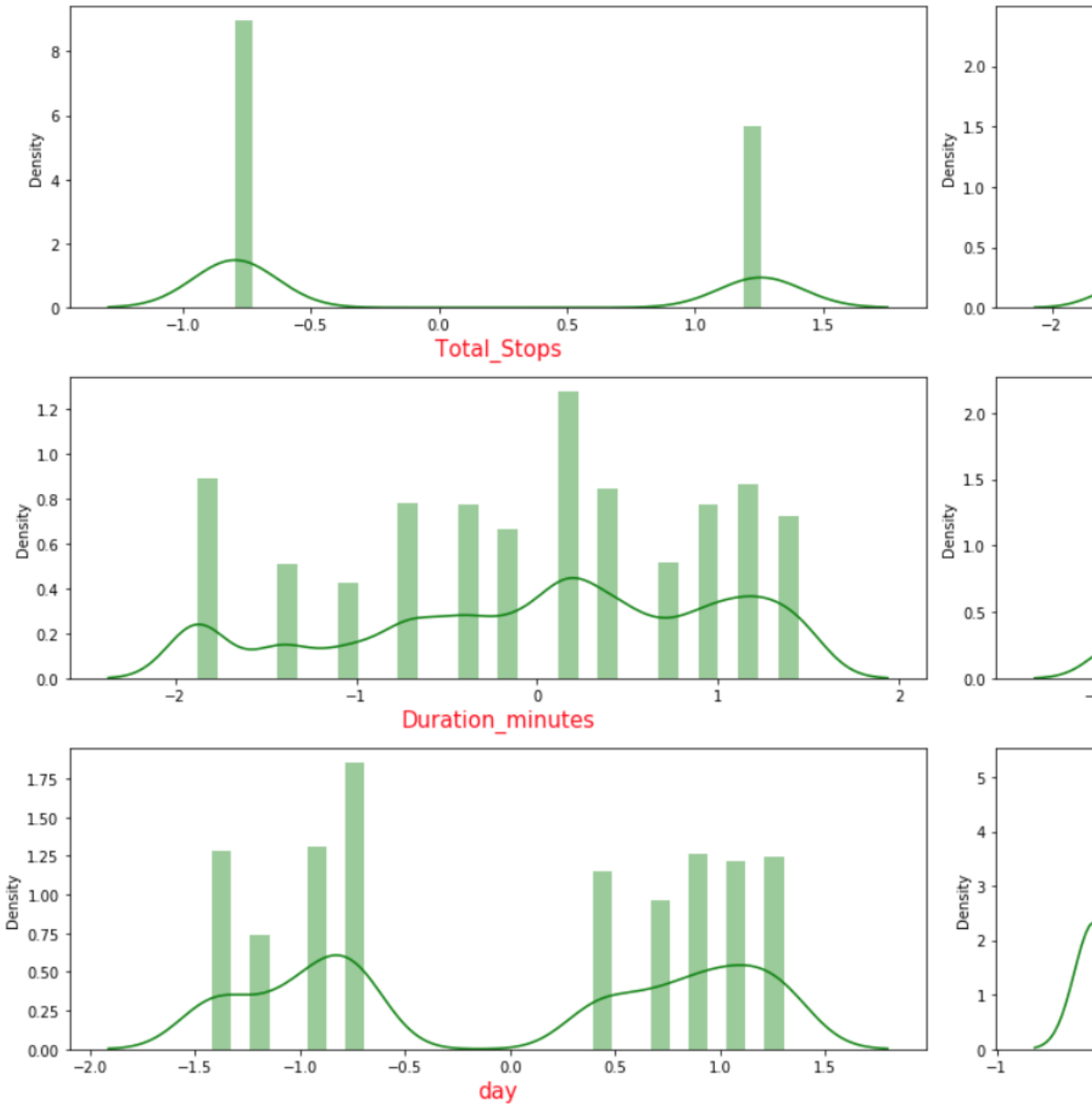
```
X_power=pt.fit_transform(features)
```

```
X_power
```

```
array([[ 1.2565683 , -1.04855191,  1.20179769, ...,
        -0.31930969, -0.23474661],
       [-0.79581826, -0.37927702,  0.95425537, ...,
        -0.31930969, -0.23474661],
       [ 1.2565683 , -1.04855191, -0.10931643, ...,
         3.13175586, -0.23474661],
       ...,
       [ 1.2565683 , -1.04855191,  0.43954947, ...,
        -0.31930969, -0.23474661],
       [ 1.2565683 , -0.66825011, -1.875644  , ...,
        -0.31930969, -0.23474661],
       [ 1.2565683 , -1.04855191,  0.70048918, ...,
        -0.31930969,  4.25991234]])
```

```
df=pd.DataFrame(X_power,columns=features.columns)
```

```
plt.figure(figsize=(18,25))
plotnumber = 1

for column in df:
    if plotnumber <= 13:
        ax = plt.subplot(7,2,plotnumber)
        sns.distplot(df[column],bins=30,color='green'
        plt.xlabel(column,fontsize = 15,color='red')
    plotnumber+=1
plt.tight_layout()
```



From here can see that our data look better than be

Building Machine Learning Models using regression algorithms

Importing the libraries

```
from sklearn.model_selection import train_t
est_split
from sklearn.preprocessing import StandardS
caler
from sklearn.model_selection import Randomi
zedSearchCV
from sklearn.tree import DecisionTreeRegres
sor
from sklearn.linear_model import LinearRegr
ession
from sklearn.ensemble import AdaBoostRegres
sor
from sklearn.linear_model import Ridge,Ridg
eCV,Lasso,LassoCV
from sklearn.ensemble import RandomForestRe
gressor
from sklearn.metrics import mean_absolute_e
rror
from sklearn.metrics import r2_score, mean_
squared_error
from sklearn import metrics
```

```
## Features standarization
scale = StandardScaler()
X_scaled = scale.fit_transform(df)
```

```
X_scaled
```

```
array([[ 1.2565683 , -1.04855191,  1.20179769, .
        -0.31930969, -0.23474661],
       [-0.79581826, -0.37927702,  0.95425537, .
        -0.31930969, -0.23474661],
       [ 1.2565683 , -1.04855191, -0.10931643, .
         3.13175586, -0.23474661],
       ...,
       [ 1.2565683 , -1.04855191,  0.43954947, .
        -0.31930969, -0.23474661],
       [ 1.2565683 , -0.66825011, -1.875644  , .
        -0.31930969, -0.23474661],
       [ 1.2565683 , -1.04855191,  0.70048918, .
        -0.31930969,  4.25991234]])
```

```
##  let's build the model with this random state

x_train, x_test, y_train,y_test = train_test_spl



Lr = LinearRegression()
Lr.fit(x_train, y_train)
```

```
LinearRegression()
```

```
y_lr_pred = Lr.predict(x_test)
```

```
print('*****Accuracy ****\n')
print(metrics.r2_score(y_test,y_lr_pred))
```

```
*****Accuracy ****

0.6762890427659581
```

```
### let's check the mean_absolute_error

MAE = mean_absolute_error(y_test,y_lr_pred)
print(MAE)
```

```
1820.0333548756207
```

```
## let's also verify the score
score = Lr.score(x_test,y_test)
print(score)
```

```
0.6762890427659581
```

```
## Models evaluation using Mean Square Error (MS

print('\nThe Mean Square Erro of LinearRegressio
r_pred))
```

```
The Mean Square Erro of LinearRegression model is
```

The r2_score 67% is not bad and also t
build more models see which one fit be

:
```
### instantiate the models

dtr = DecisionTreeRegressor(random_state=99)
ada = AdaBoostRegressor(random_state=99)
rid = Ridge(random_state=99)
las = Lasso(random_state=99)
rfr = RandomForestRegressor(random_state=99)
```

:
```
dtr.fit(x_train,y_train)
ada.fit(x_train,y_train)
rid.fit(x_train,y_train)
las.fit(x_train,y_train)
rfr.fit(x_train,y_train)
print('Models are accessfully fitted.!!!')
```

```
Models are accessfully fitted.!!!
```

:
```
dt_ypred = dtr.predict(x_test)
ada_ypred = ada.predict(x_test)
rid_ypred = rid.predict(x_test)
las_ypred = las.predict(x_test)
rfr_pred = rfr.predict(x_test)
print('Models are accessfully predicted.!!!')
```

```
Models are accessfully predicted.!!!
```

Models Evaluation

:
```
## Models evaluation using Mean Absolute Error
```

```
## Models evaluation using Mean Absolute Erro

dt_mae  = mean_absolute_error(y_test,dt_ypred)
ada_mae = mean_absolute_error(y_test,ada_ypred)
rid_mae = mean_absolute_error(y_test,rid_ypred)
las_mae = mean_absolute_error(y_test,las_ypred)
rfr_mae = mean_absolute_error(y_test,rfr_pred)

print('\n','-'*50)
print('\nThe Mean Absolute Erro of DecisionTreeR
print('\n','-'*50)
print('\nThe Mean Absolute Erro of AdaBoostRegre
print('\n','-'*50)
print('\nThe Mean Absolute Erro of Ridge model i
print('\n','-'*50)
print('\nThe Mean Absolute Erro of Lasso model i
print('\n','-'*50)
print('\nThe Mean Absolute Erro of RandomForestR
```

```
--------------------------------------------------

The Mean Absolute Erro of DecisionTreeRegressor m

--------------------------------------------------

The Mean Absolute Erro of AdaBoostRegressor model

--------------------------------------------------

The Mean Absolute Erro of Ridge model is: 1821.14

--------------------------------------------------

The Mean Absolute Erro of Lasso model is: 1820.71

--------------------------------------------------

The Mean Absolute Erro of RandomForestRegressor m
```

From the above verification we can se model :MAE =1271 and RandomForest better our data. ¶

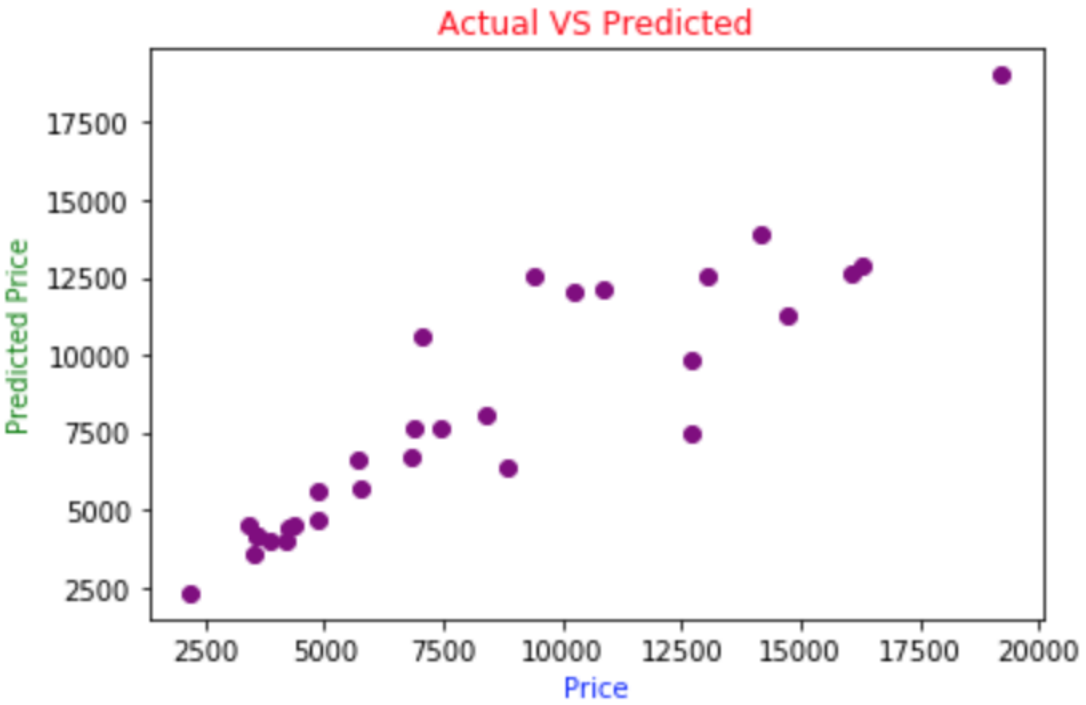
]:
```
## Models evaluation using Mean Square Error (

print('\nThe Mean Square Erro of DecisionTreeRe
t,dt_ypred))
print('\n','-'*50)
print('\nThe Mean Square Erro of AdaBoostRegres
a_ypred))
print('\n','-'*50)
print('\nThe Mean Square Erro of Ridge model is
print('\n','-'*50)
print('\nThe Mean Square Erro of Lasso model is
print('\n','-'*50)
print('\nThe Mean Square Erro of RandomForestRe
t,rfr_pred))
```

```
The Mean Square Erro of DecisionTreeRegressor m

 ----------------------------------------------

The Mean Square Erro of AdaBoostRegressor model

 ----------------------------------------------

The Mean Square Erro of Ridge model is: 6812200

 ----------------------------------------------

The Mean Square Erro of Lasso model is: 6811212

 ----------------------------------------------

The Mean Square Erro of RandomForestRegressor m
```

From the above verification we can see that Ra =3627425 and RandomForestRegressor model

```
### let's plot the actual value and the predicte

plt.scatter(y_test,rfr_pred,color ='purple')
plt.xlabel('Price',color='blue')
plt.ylabel('Predicted Price',color='green')
plt.title('Actual VS Predicted',color='r')
```

```
Text(0.5, 1.0, 'Actual VS Predicted')
```



From above scatter plot we can say that our bes

Hyper parameter Tuning (HPT)

```
### importing the libraries
from sklearn.model_selection import GridSearchCV
```

```
grid_param1 = {
    'criterion':['mse', 'mae'],
    'max_depth':range(2,10,3),
    'min_samples_leaf':range(1,10,2),
    'min_samples_split':range(2,10,2),
    'max_features':["auto", "sqrt", "log2"]
    }
```

```
grid_search1 = GridSearchCV(estimator=rfr,
                            param_grid=grid_param1,
                            cv=5,n_jobs=-1)
```

```
## again train the model

grid_search1.fit(x_train,y_train)
```

```
GridSearchCV(cv=5, estimator=RandomForestRegress
             param_grid={'criterion': ['mse', 'm
                         'max_depth': range(2, 1
                         'max_features': ['auto'
                         'min_samples_leaf': ran
                         'min_samples_split': ra
```

```
# best parameters

best_parameters = grid_search1.best_params_
print(best_parameters)
```

```
{'criterion': 'mse', 'max_depth': 8, 'max_feature
mples_split': 6}
```

Now we got the best paramters

```
## so let's build again the decisionTreeRegresso

rfr_model = RandomForestRegressor(criterion='mse
                                  max_depth=8,
                                  max_features= 'auto'
                                  min_samples_leaf=1,
                                  min_samples_split=6,
```

```
## let's fit gain the model with the data

rfr_model.fit(x_train,y_train)
pred_drfr = rfr_model.predict(x_test)

print("\n=============Test Result==============="

print(f"Accuracy :{metrics.r2_score(y_test,pred_
print("_____________________________________________
```

```
=============Test Result==============
Accuracy :86.79%

_____________________________________________
```

Great we are getting good a Accuracy , from 82%

Conclusion: our best model is Randoml accuracy = 86%.

```
print("*"*45)
print("*"*45)
```

```
*********************************************
*********************************************
```

model saving

```
import joblib
```

```
joblib.dump(rfr_model,'flight_price_prediction_m
```

```
['flight_price_prediction_model.pkl']
```

```
print("*"*45)
```

Conclusion:

In this article we saw that RandomForestRegressor fits better our given dataset than other regressor algorithms.

I hope this article helped you to understand Data Analysis, Data Preparation, and Model building approaches in simpler way.

Did I Miss Anything?
I would like to hear from you !

About the Author:
Assie Houphouet Olivier