FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

Kishan Barochiya

ACKNOWLEDGMENT

Here, I acknowledge that below presented data is as per my best view. Data are scraped from different sites and this data is for study purpose only. For best view purpose, here use different libraries and also taken help from google and other blogs. Here, some considerations are taken as per my best knowledge and I acknowledge that all data and models here I used are proceed and treated as my best view.

https://www.makemytrip.com/flights/?cmp=SEM|D|DF|G|Brand|B_M_Makemytrip_Variants|Brand-Variants-Exact|RSA|Regular|V2|529579610088&s_kwcid=AL!1631!3!529579610088!e!!g!!make%20my%20trip-&ef_id=Cj0KCQjw29CRBhCUARIsAOboZbJMMMTUZB1xr4RWADKUmKFlFPeclRHWabt3aZP2ltrEReDxQ0M8AngaAo__EALw_wcB:G:s&gclid=Cj0KCQjw29CRBhCUARIsAOboZbJMMMTUZB1xr4RWADKUmKFlFPeclRHWabt3aZP2ltrEReDxQ0M8AngaAo__EALw_wcB

https://www.yatra.com/?utm_source=google&utm_medium=cpc&utm_campaign=&gclid=Cj0KCQjw29CRBhCUARIsAOboZbJdXM1DPRuhR4ZyTBXOMcb70nv0KBREC8fclbqNtVdbGp76TzXJGmkaAty8EALw_wcB

INTRODUCTION

• Business Problem Framing

As we, all know about flight and now transportation via flight is increasing day by day. There are many flight operators and portal where customer can book their flight with best price and offers. Here, we will study past data of flight and prices and make predictive model, which will revert us, tentative price of flight on the base of input data. It will helpful to study changes in price according to various factor.

• Conceptual Background of the Domain Problem

Here, we have data of one website, which provides facilities to book flight ticket for international or internal travel. Now there are many types of customers, many wants to reach at destination in short time, many wants to book at cheapest price. Some's are looking for facilities etc. so here we study the data and create the prediction model, which revert us the tentative price for flight ticket booking. We might notice that sometimes in holiday or festival season there are high travelling and that is why pricing of flights price going to high. Here we will consider that all features and its effect on flights price.

• Review of Literature

We scraped the data from website yatra.com which one is the best portal where one can book their flights with best facilities and option like different companies, facilitates, stops, price, offers etc.
We cleared this data and we have 10683 rows available with 11 columns or variables. We will clear that data first like null values, outliers, data types, duplicates etc then we will do EDA for it to understand data deeply and according to it proceed with data then pre- process data like scale down, feature selection then with the best data format we will feed it into model to check accuracy on testing dataset and on best 2 model we will go with hyper parameter tuning to increase accuracy at maximum level. Final step is to cross validation where we re-run all model 4 times and take it average score as final accuracy score. Highest accuracy contain model will dump as final best-fit model and then apply this model on unknown model to check accuracy.

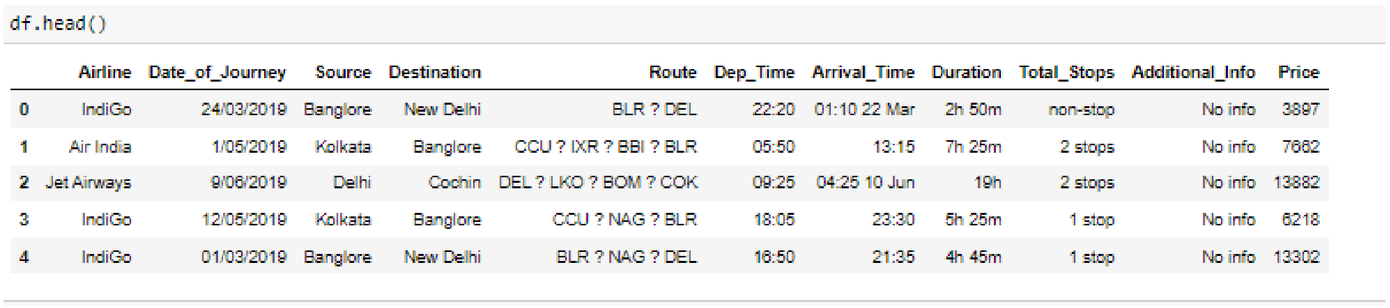
• Motivation for the Problem Undertaken

This model is very useful to predict price of flights with reference to all features and company might be suggest best suit flight option by using this model. We need to change the input data with time span to consider changes with actual market condition and features. As company level using this model, they can compare all competitive platform price for same flights

Analytical Problem Framing

Mathematical/ Analytical Modelling of the Problem

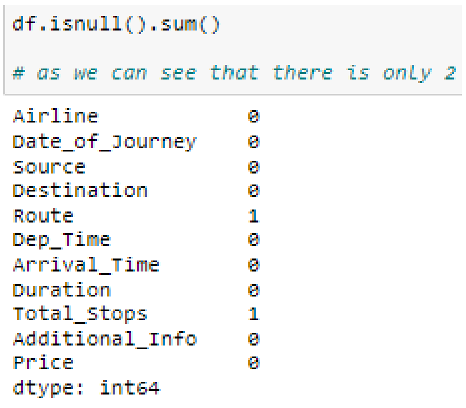
First we load the dataset and check some basic statistically as per below.



```
df.head()
```

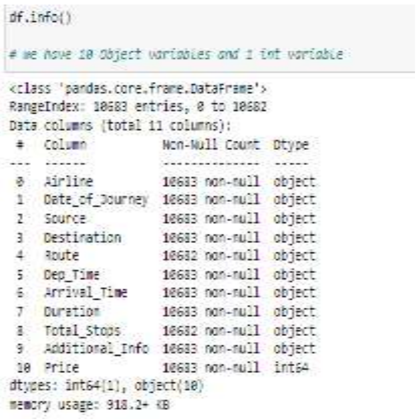
	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	24/03/2019	Banglore	New Delhi	BLR ? DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	1/05/2019	Kolkata	Banglore	CCU ? IXR ? BBI ? BLR	05:50	13:15	7h 25m	2 stops	No 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

What is statistics say for our data?



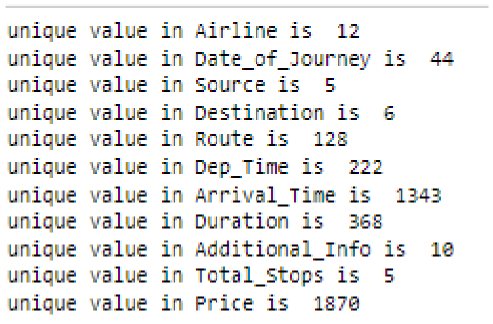
```
df.isnull().sum()

# as we can see that there is only 2
```

```
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
```



```
df.info()

# we have 10 Object variables and 1 int variable
```

```
<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
```

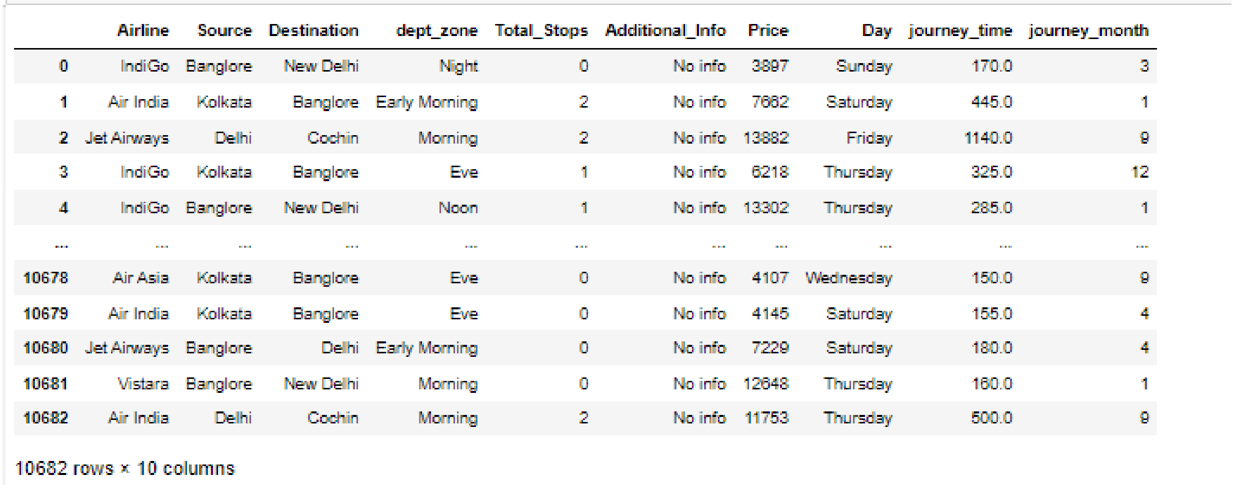


```
unique value in Airline is  12
unique value in Date_of_Journey is  44
unique value in Source is  5
unique value in Destination is  6
unique value in Route is  128
unique value in Dep_Time is  222
unique value in Arrival_Time is  1343
unique value in Duration is  368
unique value in Additional_Info is  10
unique value in Total_Stops is  5
unique value in Price is  1870
```

1) We have 10 objective data types and one int format dtypes in 11 variables.
2) We have only 2 null values in dataset and we will proceed after drop it.
3) We have higher unique values in Arrival time and duration.

- ## Data Pre-processing:

  - We take day and month from date of journey column and drop the original column.
  - Total stops is converted into int format.
  - Dept time is converted in 5 zone for better data understanding
  - After pre-processing we have data format as per below snap.

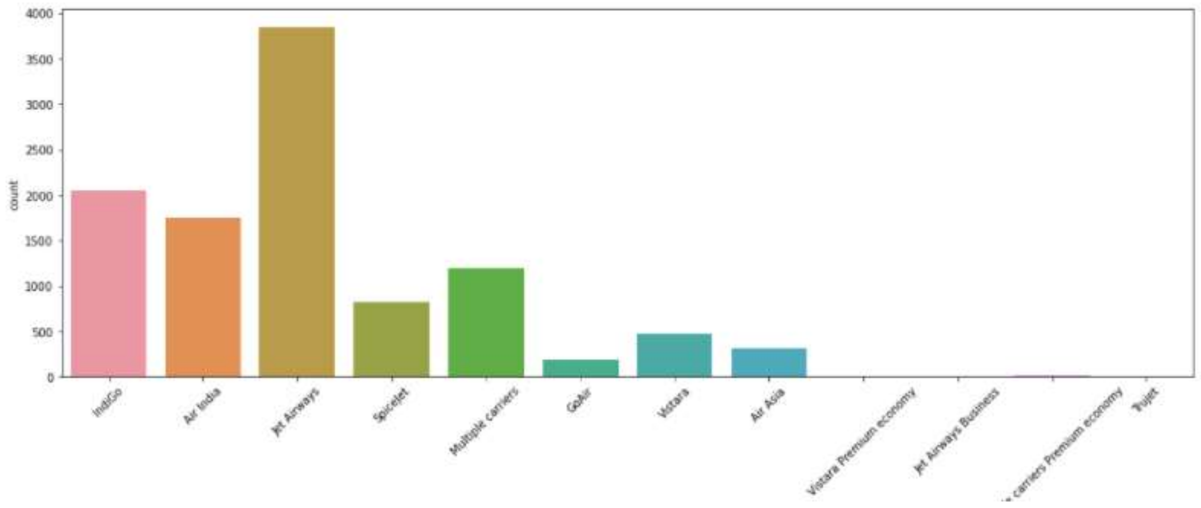


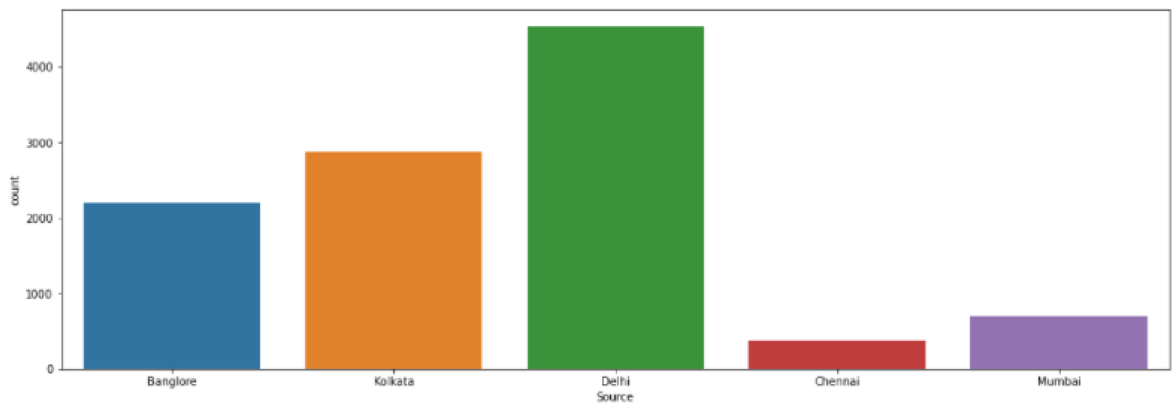
	Airline	Source	Destination	dept_zone	Total_Stops	Additional_Info	Price	Day	journey_time	journey_month
0	IndiGo	Banglore	New Delhi	Night	0	No info	3897	Sunday	170.0	3
1	Air India	Kolkata	Banglore	Early Morning	2	No info	7662	Saturday	445.0	1
2	Jet Airways	Delhi	Cochin	Morning	2	No info	13882	Friday	1140.0	9
3	IndiGo	Kolkata	Banglore	Eve	1	No info	6218	Thursday	325.0	12
4	IndiGo	Banglore	New Delhi	Noon	1	No info	13302	Thursday	285.0	1
...	...	...	...	...	...	...	...	...	...	...
10678	Air Asia	Kolkata	Banglore	Eve	0	No info	4107	Wednesday	150.0	9
10679	Air India	Kolkata	Banglore	Eve	0	No info	4145	Saturday	155.0	4
10680	Jet Airways	Banglore	Delhi	Early Morning	0	No info	7229	Saturday	180.0	4
10681	Vistara	Banglore	New Delhi	Morning	0	No info	12648	Thursday	160.0	1
10682	Air India	Delhi	Cochin	Morning	2	No info	11753	Thursday	500.0	9

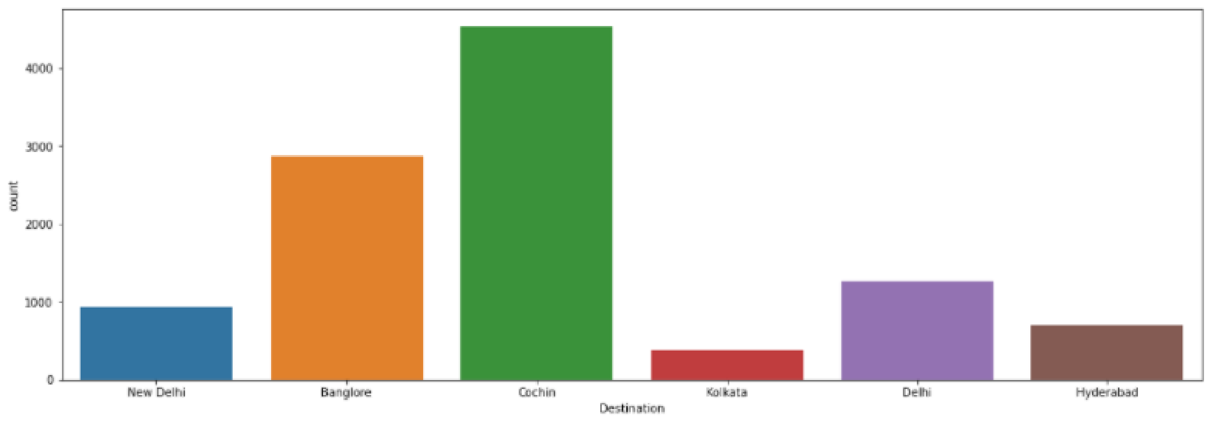
10682 rows × 10 columns

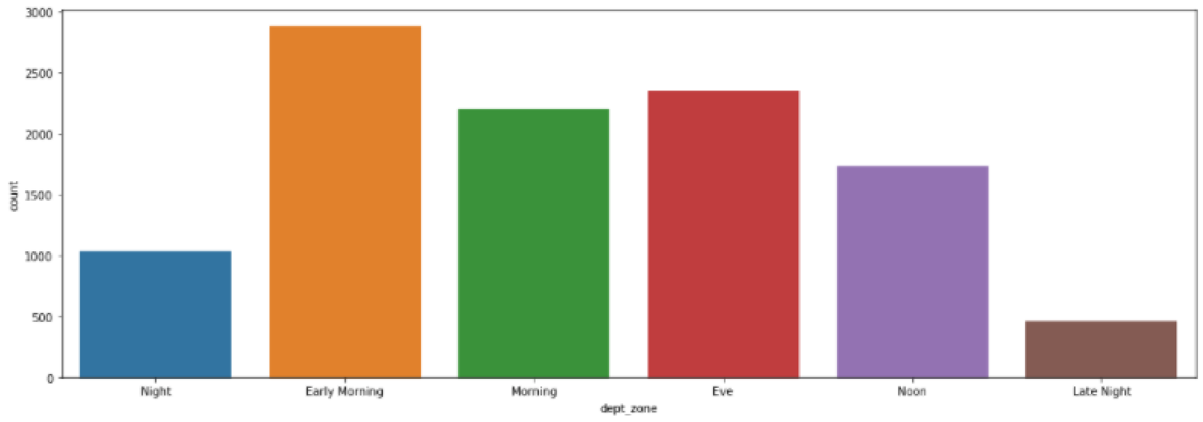
- ## Data Inputs- Logic- Output Relationships

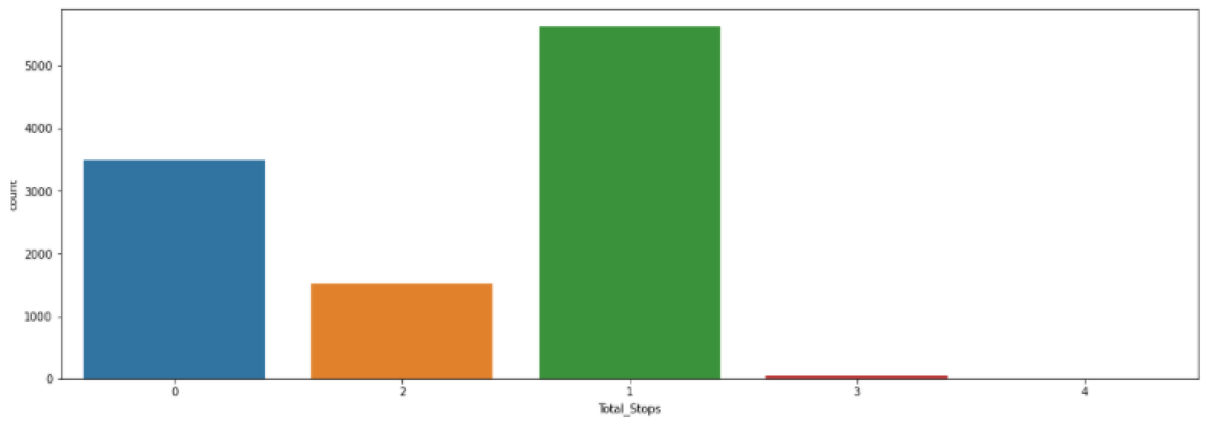
  First, we will check about data distribution of columns

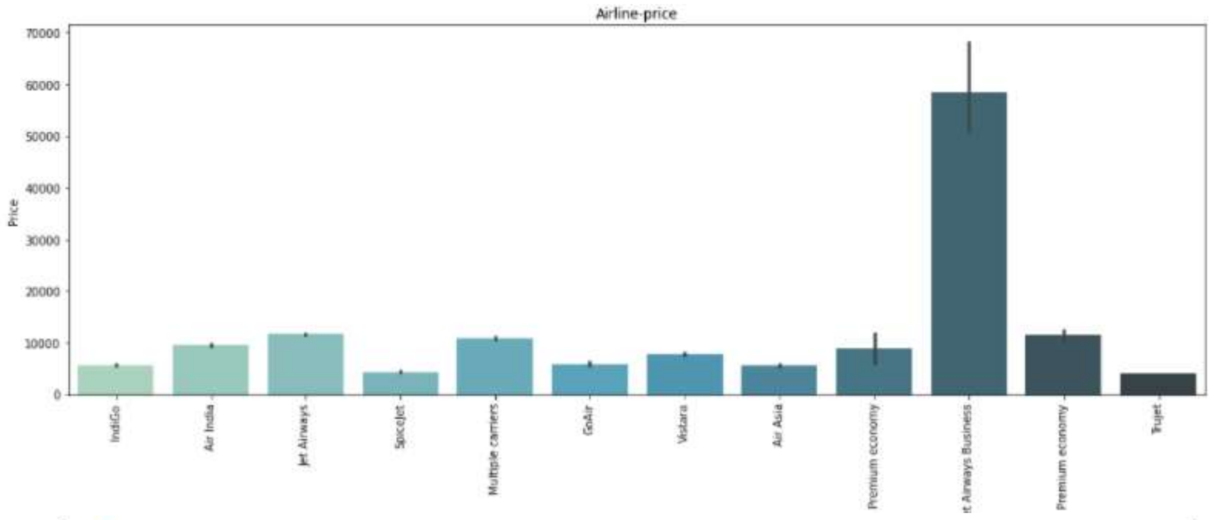
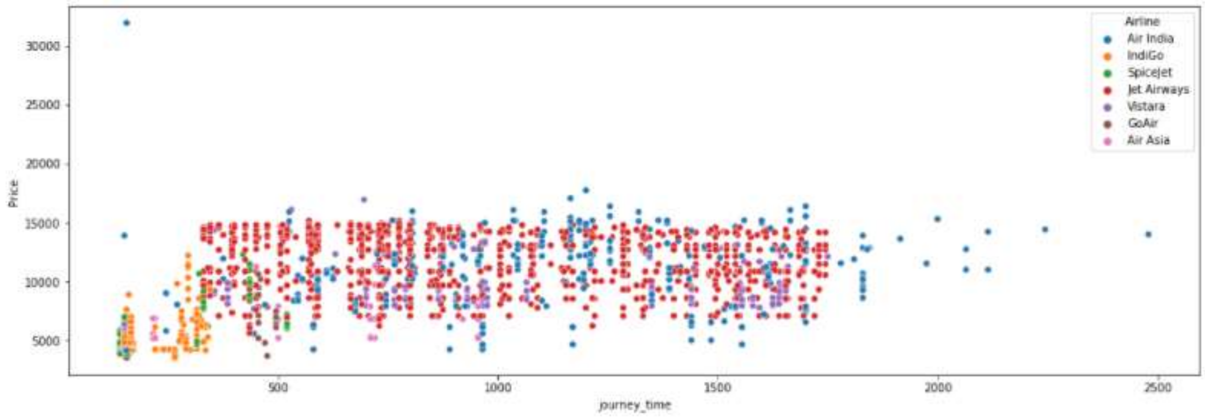


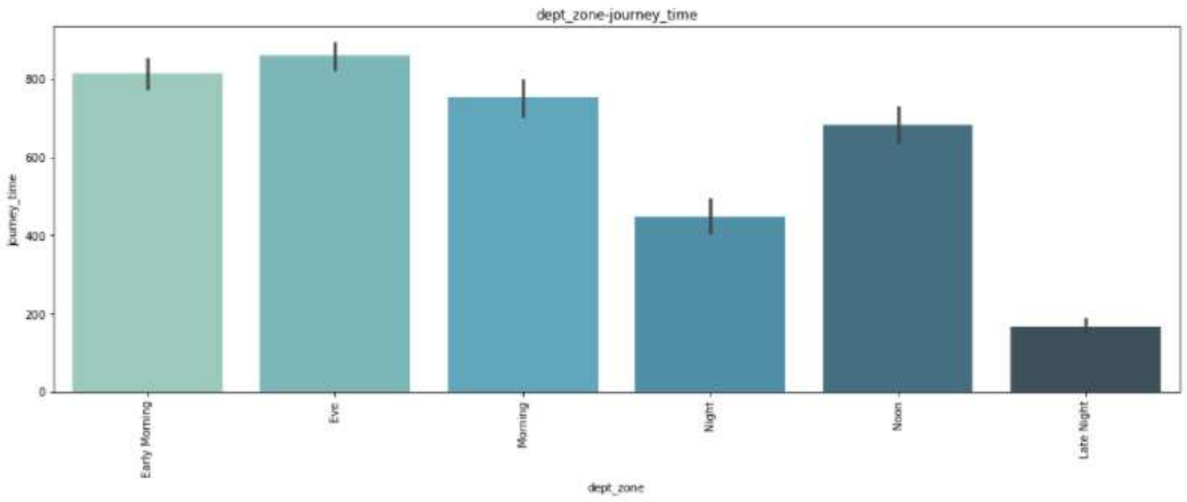

count
0
1000
2000
3000
4000
Banglore
Kolkata
Delhi
Chennai
Mumbai
Source

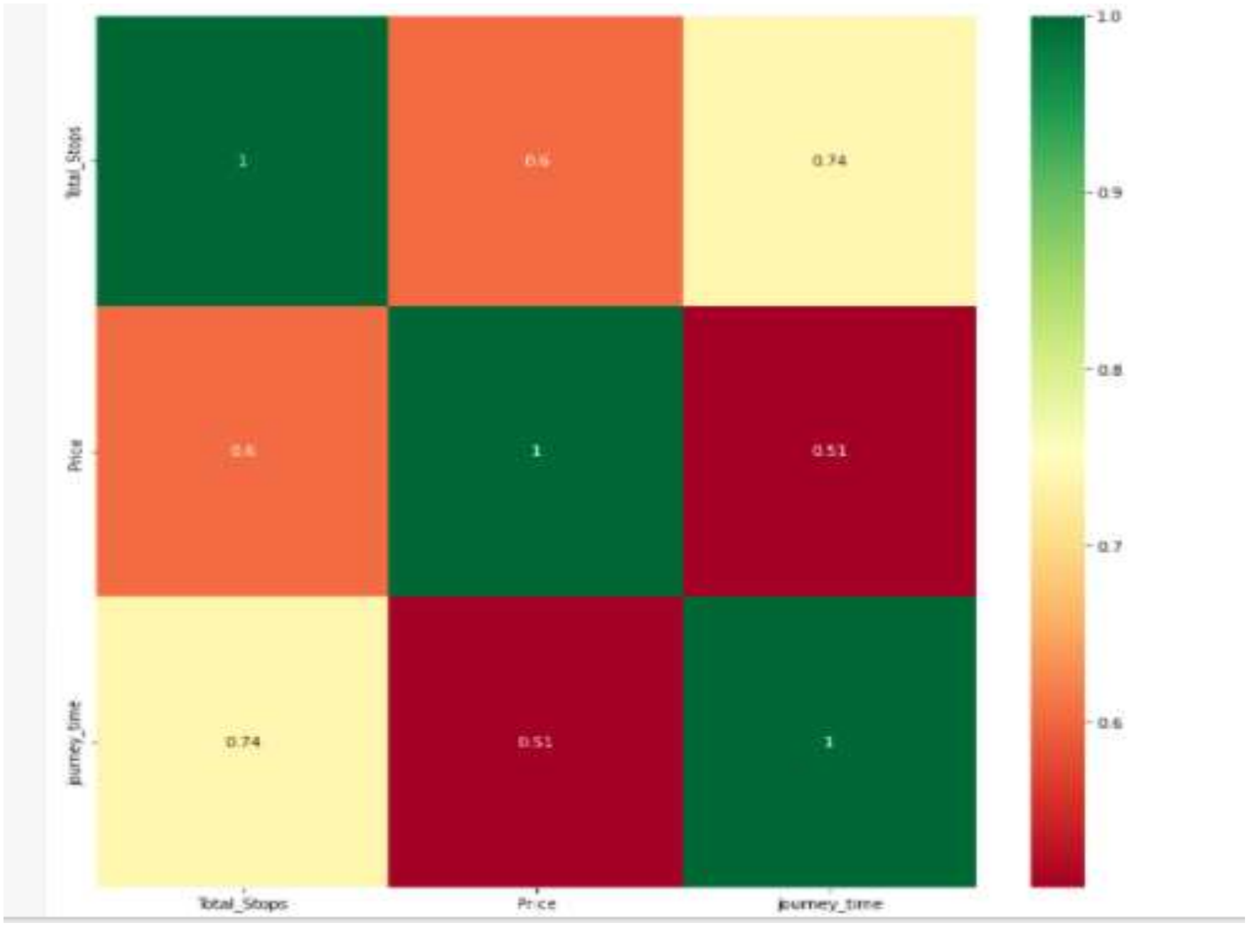

count
0
1000
2000
3000
4000
New Delhi
Banglore
Cochin
Kolkata
Delhi
Hyderabad
Destination


count
0
500
1000
1500
2000
2500
3000
Night
Early Morning
Morning
Eve
Noon
Late Night
dept_zone


count
0
1000
2000
3000
4000
5000
0
2
1
3
4
Total_Stops


Airline-price
Price
70000
60000
50000
40000
30000
20000
10000
0
IndiGo
Air India
Jet Airways
SpiceJet
Multiple carriers
GoAir
Vistara
Air Asia
Premium economy
t Airways Business
Premium economy
Trujet

Airline
Air India
IndiGo
SpiceJet
Jet Airways
Vistara
GoAir
Air Asia
Price
30000
25000
20000
15000
10000
5000
500
1000
1500
2000
2500
journey_time


dept_zone-journey_time
journey_time
0
200
400
600
800
Early Morning
Eve
Morning
Night
Noon
Late Night
dept_zone


Total_Stops
Price
journey_time
1
0.6
0.74
0.6
1
0.51
0.74
0.51
1
Total_Stops
Price
journey_time
1.0
0.9
0.8
0.7
0.6

EDA CONCLUSION:

- Price is directly effective with Journey time and total stops.
- Total stops is inversely co-related with journey time.
- Source have co-relation with stops and journey time as distance increase then travelling time also increase and total stops also may be increase.
- Journey month and journey day have no much co-relation with any variable.

State the set of assumptions (if any) related to the problem under consideration

Based on EDA, we drop column of addition info and dept. zone.

Hardware and Software Requirements and Tools Used

We will use below libraries and software used:

FOR BASIC DATA STUDY:

- import numpy as np
- import pandas as pd
- pd.get_option("display.max_columns")

FOR DATA VISULISATION

- import matplotlib.pyplot as plt
- import seaborn as sns

DATA CLEANING AND PRE_PROCESSING

- from scipy.stats import boxcox
- from scipy.stats import zscore
- from sklearn.preprocessing import StandardScaler
- from statsmodels.stats.outliers_influence import variance_inflation_factor
- from sklearn.model_selection import train_test_split
- from sklearn.decomposition import PCA

MODEL BUILDING

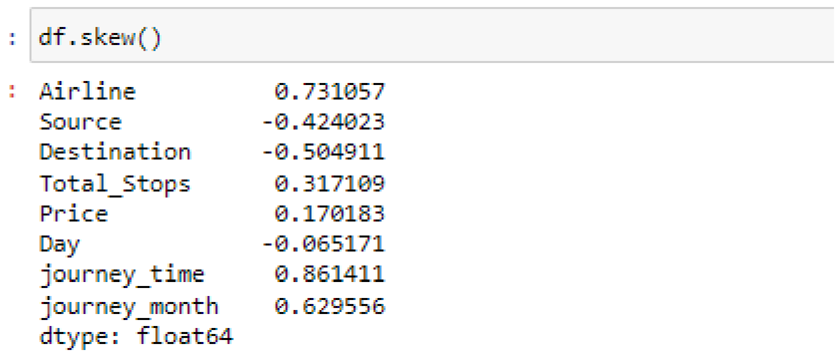
- from sklearn.tree import DecisionTreeRegressor
- from sklearn.svm import SVR
- from sklearn.neighbors import KNeighborsRegressor
- from sklearn import metrics
- from sklearn import ensemble
- from sklearn.utils import shuffle
- from sklearn.metrics import mean_squared_error, r2_scorefrom sklearn.naive_bayes import GaussianNB

- from sklearn.ensemble import RandomForestRegressor
- from sklearn.ensemble import AdaBoostRegressor
- from sklearn.ensemble import GradientBoostingRegressor
- from sklearn.model_selection import GridSearchCV
- from sklearn.model_selection import RandomizedSearchCV
- from sklearn.model_selection import cross_val_score

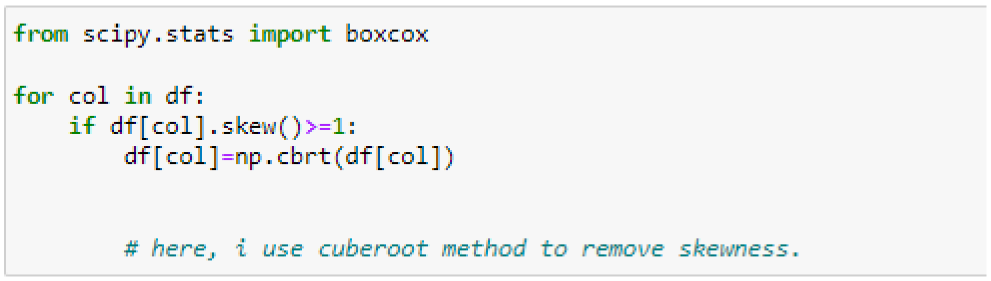
Model/s Development and Evaluation

- ### Identification of possible problem-solving approaches (methods)

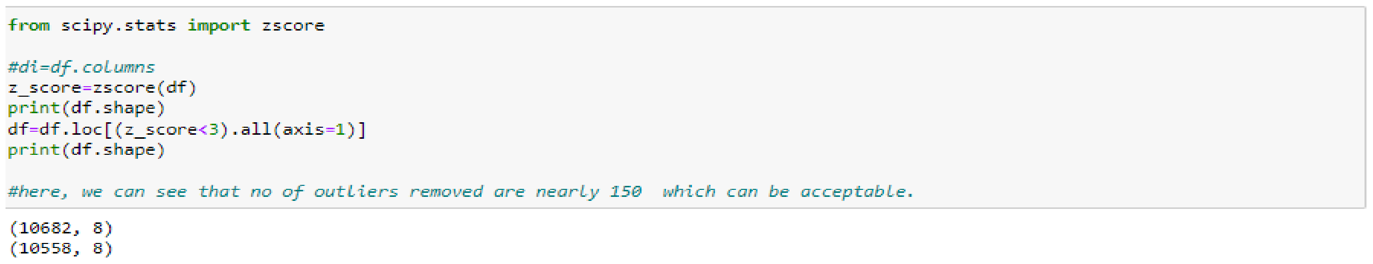
Now our data is ready for modelling purpose but here we will give our best effort to get maximum accuracy. For that we will scaledown data so we can get low error ratio. Remove skewness of data.



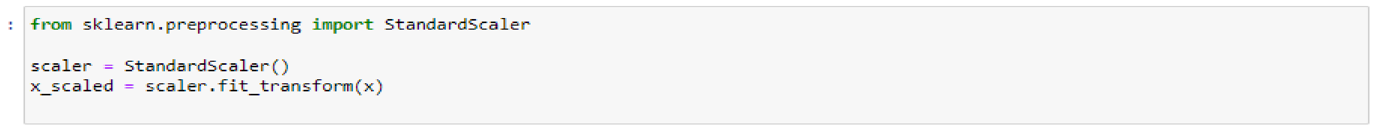
```
: df.skew()

: Airline           0.731057
  Source           -0.424023
  Destination      -0.504911
  Total_Stops       0.317109
  Price             0.170183
  Day              -0.065171
  journey_time      0.861411
  journey_month     0.629556
  dtype: float64
```



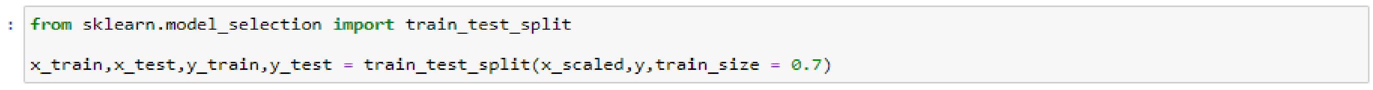
```
from scipy.stats import boxcox

for col in df:
    if df[col].skew()>=1:
        df[col]=np.cbrt(df[col])


        # here, i use cuberoot method to remove skewness.
```



```
from scipy.stats import zscore

#di=df.columns
z_score=zscore(df)
print(df.shape)
df=df.loc[(z_score<3).all(axis=1)]
print(df.shape)

#here, we can see that no of outliers removed are nearly 150  which can be acceptable.

(10682, 8)
(10558, 8)
```



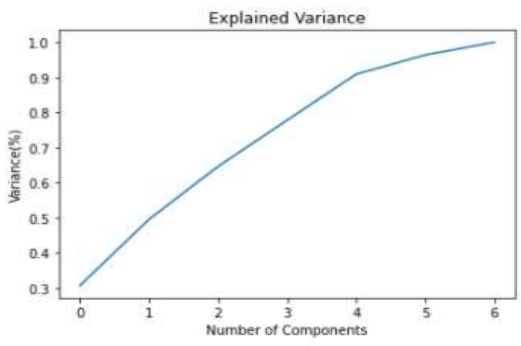
```
: from sklearn.preprocessing import StandardScaler

  scaler = StandardScaler()
  x_scaled = scaler.fit_transform(x)
```



```
: from sklearn.model_selection import train_test_split

  x_train,x_test,y_train,y_test = train_test_split(x_scaled,y,train_size = 0.7)
```

FEATURE SELECTION:

```
from sklearn.decomposition import PCA
pca = PCA()
PrincipalComponents = pca.fit_transform(x_scaled)
plt.figure()
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Number of Components')
plt.ylabel('Variance(%)')
plt.title('Explained Variance')
plt.show()
```



- ## Testing of Identified Approaches (Algorithms)

We will use below algorithm for training and testing of dataset.

- Decision Tree Regressor
- Support vector regression
- KNeighbors Regressor
- Linear regression
- RandomForest Regressor
- AdaBoost Regressor
- GradientBoosting Regressor
-

Logistic regression

```
ln=LinearRegression()
ln.fit(x_train,y_train)
pred=ln.predict(x_test)
print('Train Score',ln.score(x_train,y_train)*100)
print('Testing:-',ln.score(x_test,y_test)*100)
print('Error')
print('MAE',mean_absolute_error(y_test,pred))
print('MSE',mean_squared_error(y_test,pred))
print('RMSE',np.sqrt(mean_squared_error(y_test,pred)))
print('R2 Score',r2_score(y_test,pred)*100)
```

```
Train Score 52.18428827965857
Testing:- 49.147356790696215
Error
MAE 1.911260396936656
MSE 5.61866241674912
RMSE 2.370371788717778
R2 Score 49.147356790696215
```

Conclusion for logistic regression:

- On training data, accuracy score is 52%

- On testing data, accuracy score is 49%
- R2 score is 49%.

Decision Tree Regressor

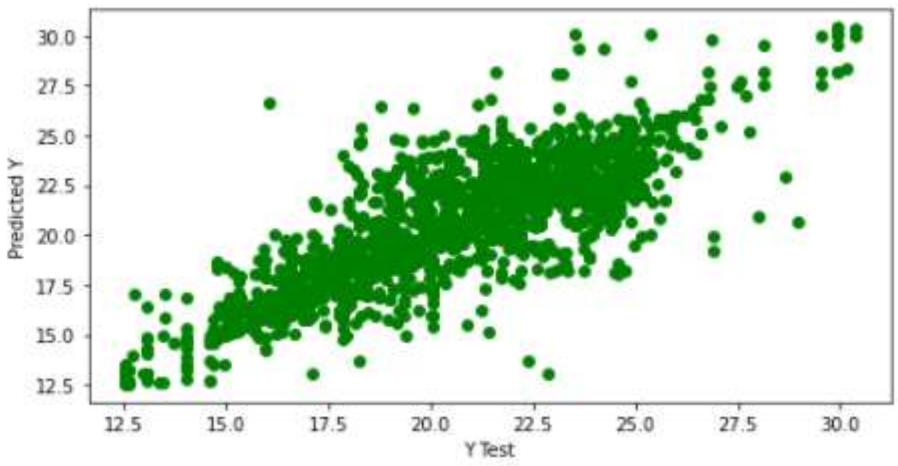
```
dtr.fit(x_train,y_train)
preddtr=dtr.predict(x_test)
print("DecisionTreeRegressor")
print('Training Score:-',dtr.score(x_train,y_train)*100)
print('Testing:-',dtr.score(x_test,y_test)*100)
print('MAE',mean_absolute_error(y_test,preddtr))
print('MSE',mean_squared_error(y_test,preddtr))
print('RMSE',np.sqrt(mean_squared_error(y_test,preddtr)))
print('R2 Score',r2_score(y_test,preddtr)*100)
```

```
DecisionTreeRegressor
Training Score:- 94.06094660244918
Testing:- 69.40188309055912
MAE 1.2491506512364172
MSE 3.3807581799586246
RMSE 1.838683817288504
R2 Score 69.40188309055912
```

Conclusion for Decision Tree Regressor:

- On training set accuracy score is 94% but on testing set it 69%, which is not good for dataset because model is not much success to predict testing data accurately.

- In Decision Tree Classifier, R2 score is 69%.



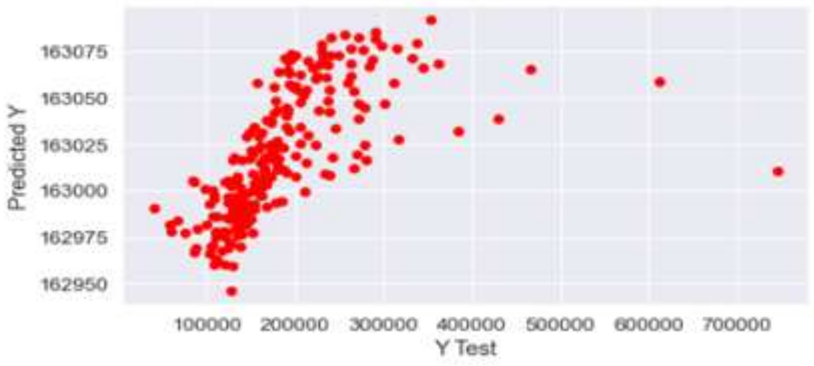
SVR:

```
]: fun(svr)
```

```
TRAINING:- 68.54206619933612
Testing:- 67.4174822233865
MAE 1.4228812567008509
MSE 3.6000128316048805
RMSE 1.8973699775228026
R2 Score 67.4174822233865
```

Conclusion on SVR:

In SVR, we have accuracy of 67% in both training and testing dataset and R2_score is also same.



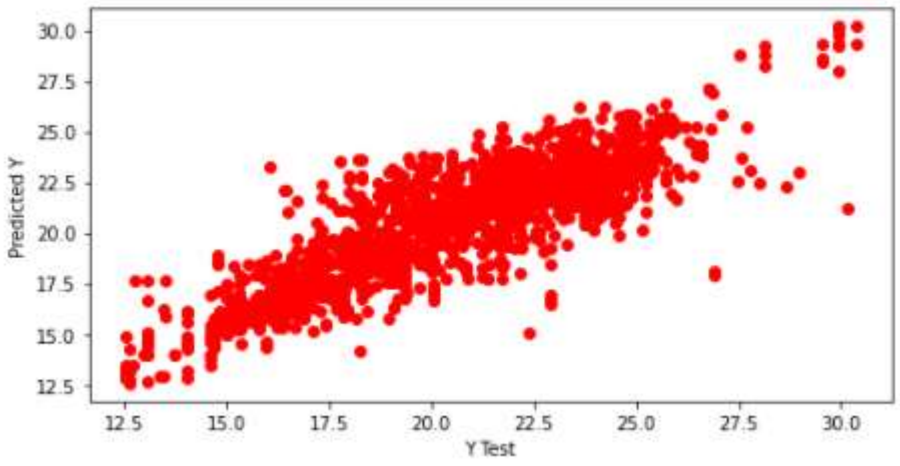
KNN:

```
fun(kn)
```

```
TRAINING:- 84.7351156014758
Testing:- 76.53303412128652
MAE 1.185475155391741
MSE 2.5928437716633415
RMSE 1.6102309684214067
R2 Score 76.53303412128652
```

Conclusion on KNN:

Training score is 84% while testing 75% and R2 score is 76%. Its looking better for prediction compare to other and as per visualise we can see that data are predicted on one line.



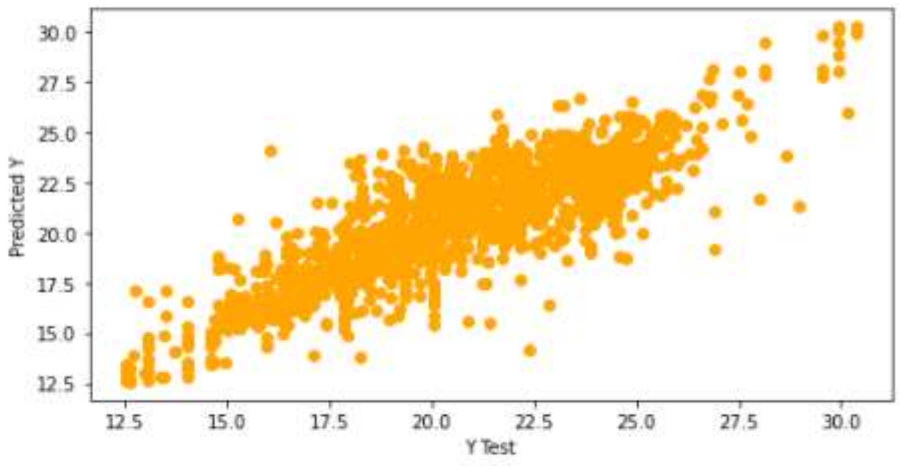
Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
rd=RandomForestRegressor()
fun(rd)
```

```
TRAINING:- 92.77439069625065
Testing:- 77.65147424099854
MAE 1.1221175680810107
MSE 2.4692683374396855
RMSE 1.5713905744402585
R2 Score 77.65147424099854
```

Conclusion on Random Forest Regressor:

We can mark that 77% accuracy in testing dataset and 92% training accuracy. R2 score is 77%.



BOOSTING TECHNIQUE

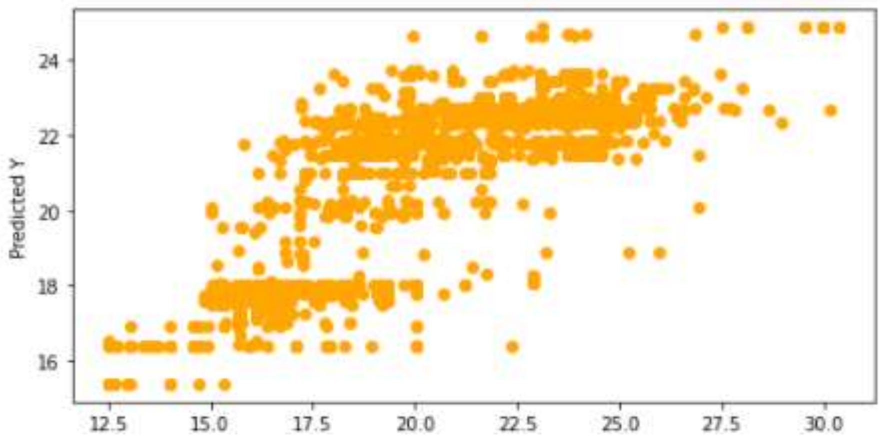
Adaboost Classifier:

```
from sklearn.ensemble import AdaBoostRegressor
ad=AdaBoostRegressor()
fun(ad)
```

```
TRAINING:- 62.023660166246856
Testing:- 60.64157840278648
MAE 1.7269296474280686
MSE 4.348676297918992
RMSE 2.085348004031699
R2 Score 60.64157840278648
```

Conclusion on Adaboost Regressor:

We got accuracy of 62% in raining and in testing 60% accuracy.
R2 score is 60%



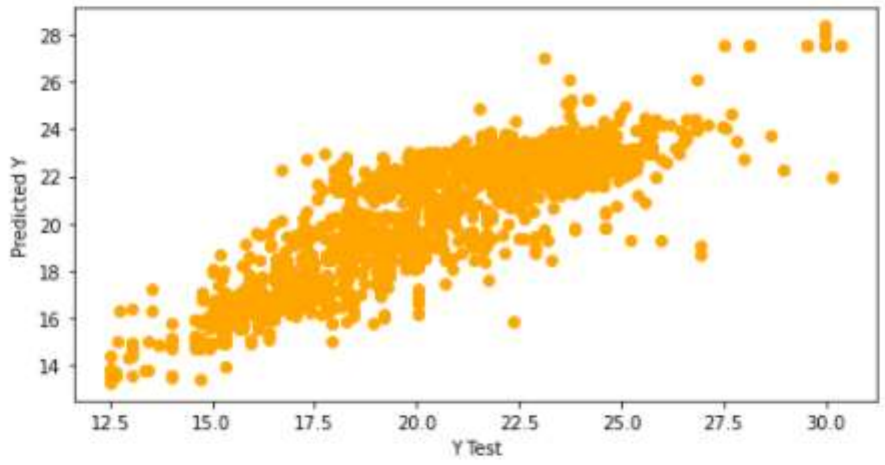
Gradient Boosting Classifier

```
from sklearn.ensemble import GradientBoostingRegressor
gd=GradientBoostingRegressor()
fun(gd)
```

```
TRAINING:- 78.73786347132928
Testing:- 77.40092771789801
MAE 1.2156661361400423
MSE 2.4969509954914617
RMSE 1.5801743560415926
R2 Score 77.40092771789801
```

Conclusion on Gradient boosting classifier

Both training and testing accuracy is 78% and 77% R2 score.



CROSS VALIDATION TECHNIQUE:

When we tested our model on testing data set, it take observation and give result but what happened if this model applied on different data? For this, we will use cross validation technique, in this technique, we will take average of accuracy tried for 4 times on random dataset from testing data.

```
score of cross validation score for svr 67.64604460732461
score of cross validation score for dtr 69.6672906873677
score of cross validation score for rd 77.87629114115558
score of cross validation score for Gd 77.6127960591505
score of cross validation score for ad 63.67091518293395
score of cross validation score for rd hyper 80.85881048467462
score of cross validation score for gd hyper 83.61071122941004
```

- Key Metrics for success in solving problem under consideration

Cross validation:

Here, we tried many algorithms and predict accuracy nearby 90 in every case, which is good predication but we should know that it is one time training and testing and based on it we can not say that this is accurate model so we should go with some more time trail and average result of it, which is called Cross validation.

Hyper parameter tuning:

It is a accuracy increasing technique, in which we find out the best parameter for that model. After finding that parameter, we will re-fit the model and check the accuracy.

```
n_estimators = [20,50,100,600] # number of trees in the random forest
learning_rate = [.001,0.01,.1]
max_depth = [1,2,4,5,10] # maximum number of levels allowed in each decision tree
min_samples_split = [10,50,100,500] # minimum sample number to split a node
min_samples_leaf = [1, 3, 4] # minimum sample number that can be stored in a leaf node
random_state = [1] # method used to sample data points

random_grid = {'n_estimators': n_estimators,

'learning_rate' : learning_rate,

'max_depth': max_depth,

'min_samples_split': min_samples_split,

'min_samples_leaf': min_samples_leaf,

'random_state': random_state}
```

```
GBR = RandomizedSearchCV(estimator = gd,param_distributions = random_grid,
               n_iter = 100, cv = 5, verbose=2, random_state=35, n_jobs = -1)
```

```
GBR.fit(x_train,y_train)
GBR.best_params_
```

```
{'random_state': 1,
 'n_estimators': 600,
 'min_samples_split': 10,
 'min_samples_leaf': 3,
 'max_depth': 5,
 'learning_rate': 0.1}
```

```
GD1 = GradientBoostingRegressor(n_estimators = 600, min_samples_split = 40,
                                min_samples_leaf= 4,max_depth = 6, learning_rate = 0.1)
GD1.fit(x_train, y_train)
```

```
GradientBoostingRegressor(max_depth=6, min_samples_leaf=4, min_samples_split=40,
                          n_estimators=600)
```

```
fun(GD1)
```

```
TRAINING:- 89.72097480398827
Testing:- 83.99021632414183
MAE 0.9674404301077683
MSE 1.7689064749218462
RMSE 1.3300024341789176
R2 Score 83.99021632414183
```

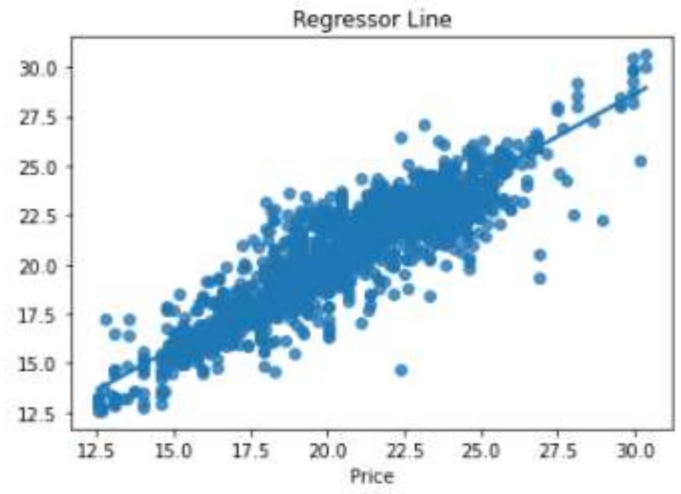
As we can see that as best parameter, we are getting higher accuracy near 3% .

- Visualizations

```
res=pd.DataFrame()
res['Actual']=y_test
res['Predict']=gd1_pred

res

print(res)
sns.regplot(y_test,gd1_pred)
plt.title('Regressor Line')
```

```
          Actual    Predict
10423  15.798248  16.430943
3628   16.895553  16.584021
5984   24.504372  21.203045
4521   29.523968  28.433288
2933   15.033997  15.642147
...          ...        ...
837    23.192462  22.840725
9794   18.641477  19.761307
2130   18.186336  17.989809
1178   16.463033  16.473597
859    20.476873  22.937221
```

PREDICATED RESULT:



As we can see that, we are getting good graph for actual vs predicted result. So our model working fine.

- Interpretation of the Results

First we cleaned that data, normalised it then we remove outliers, convert all objective columns to numerical columns and then scale down data for better accuracy. We done some process like PCA, balancing data, hyper parameter tuning for increase accuracy for model. By using some records and observation finally, we take Gradient boost hyper parameter tuned model as best-fit model. We applied model on testing dataset and check actual result.

CONCLUSION

We create the model which working with 83% accuracy so it will be beneficial for client, ticket booking portal as well as flight company to run process smoothly and they can plan according to demand of flight and client can book their ticket as best price. We need to change input data and re-run model over time span to cover continuous trends. Over the time accuracy may be changed which is cons of this model.