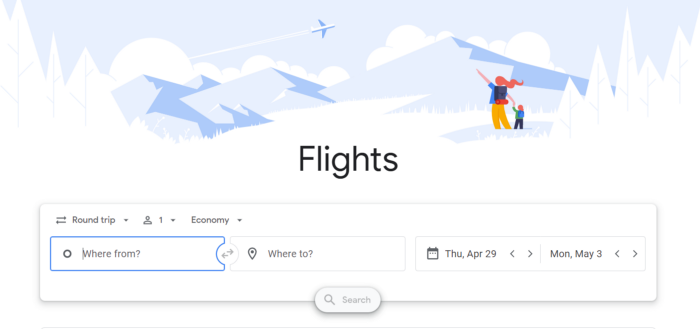
**Flight Price Prediction**



**Problem Statement**

*The fare of a flight ticket is vary based on number of aspects such as duration, number of stops, source, day, airline and many more aspects. Each airline has its own algorithms and rules to decide the fare. Machine Learning helps us to predict the price much closer than actual price. It’s always hard to guess the flight price because prices are dynamic. The purpose of this blog is to predict the fare of the flight ticket, with the help of dataset. The model prediction accuracy is very good i.e., 88.6 R squared Score with dataset. The dataset is from March 2019 to June 2019 that to includes various source and destination.*

**Plan**

*The dataset which will going to use here is Flight Price Fare took it from Kaggle. Here given dataset will be analyzed based based on all the given information. The target value will be continuous that is Fare of the flight. Though here Regression algorithms will be coming in use with the help of Python. Comparison between different — different algorithm will happen here just to get highest accuracy, which will help to get accurate price of the flight.*

**Data Analysis**



Data Analysis is a procedure for gathering raw data than converting it into useful and informative data that will help for making decisions clear by the user. Data will be collect, analyzed to answer the questions.



Flight\_Train.shape #Checking the shape of our data  
(10683, 11)

The Flight dataset contains 10683 rows and 11 unique columns.

Flight\_Train.dtypes #Checking the datatype of each attribute#Output  
Airline object  
Date\_of\_Journey object  
Source object  
Destination object  
Route object  
Dep\_Time object  
Arrival\_Time object  
Duration object  
Total\_Stops object  
Additional\_Info object  
Price int64  
dtype: object

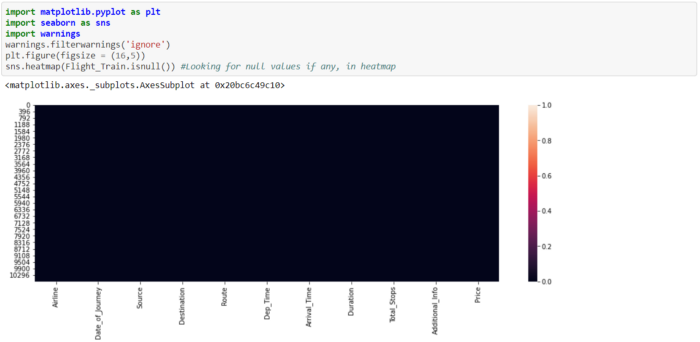
There are 10683 examples, 10 columns + 1 target variable. 10 columns are object type.

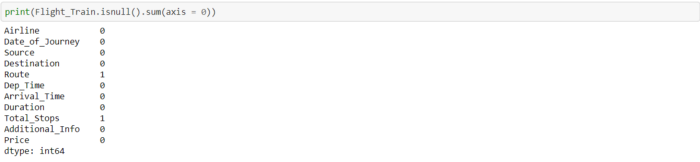
Airline —> Name of the Flight company.  
Date\_of\_Journey —> The date on which flight departure.  
Source —> From where Flight departure.  
Destination —> Arrival place.  
Route —> Route followed by Flight, Where flight stopped in between.  
Dep\_Time -> Time on which Flight Departure.  
Arrival\_Time -> Arrival time of the flight.  
Duration -> Total journey hours.  
Total\_Stops -> How many times flight stopped in between.  
Additional\_Info -> Hand bag allowed, business class etc.  
Price -> Cost of the Journey.

Target variable (Price) is int64 i.e., continuous, So Regression will be used to learn the model.

**EDA Concluding Remark**

It’s clearly visible that NULL values present in the dataset

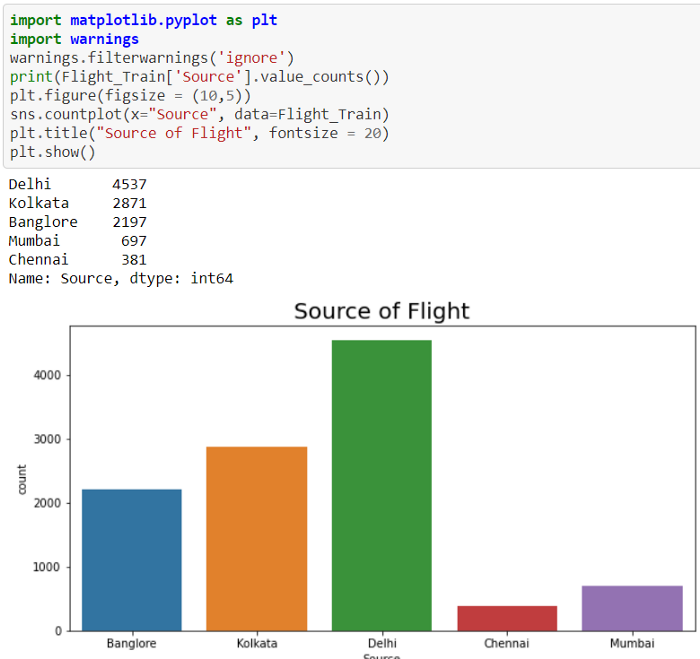




Column Route and Total\_Stops have 1–1 NULL values. There are 2 ways to get ride of these NULL values. 1st is drop null values but it’s not a good choice to go with. 2nd is replace NUL values with mean, median or mode. In this case type of variable is int. We can use mean, median or mode but in case of object we have to use mode.

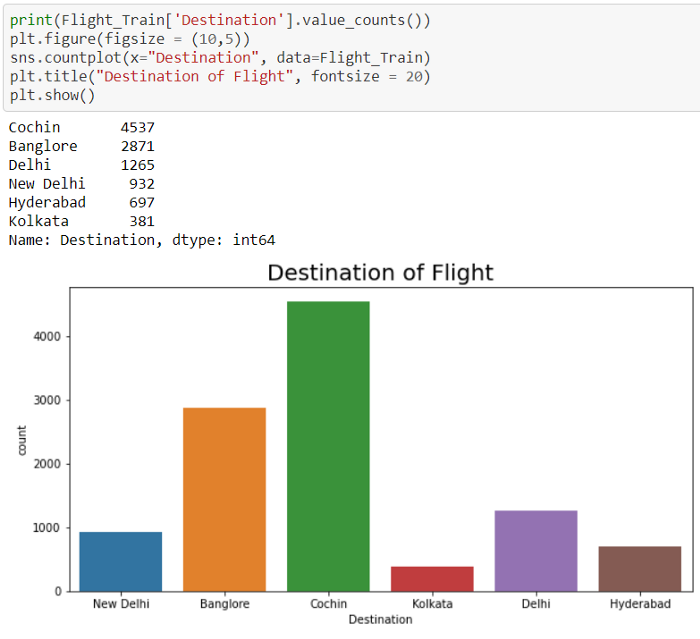
https://miro.medium.com/max/700/1*wvvwCpwd0KWog_IfvetvOA.png

Replaced NULL values with mode. Now no more NULL values present in the dataset.



Maximum number of Flights Source is Delhi i.e. 4537

Only 381 Flights Departure was Chennai compare to Delhi it’s very less.



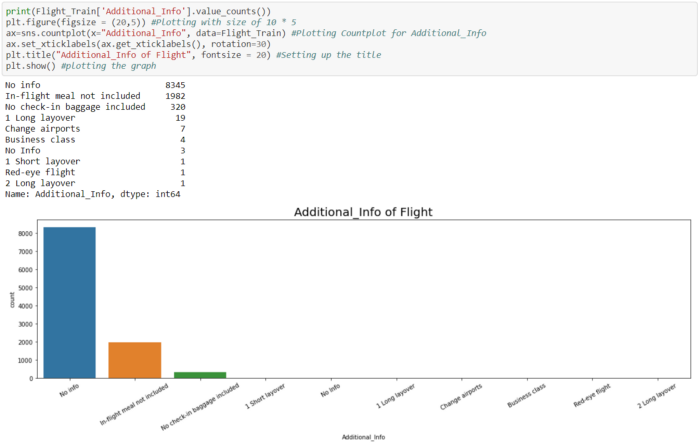
4537 Flight Destination was Cochin i.e. Maximum

Only 381 Flight Arrival was Kolkata which is very less compared to Cochin.



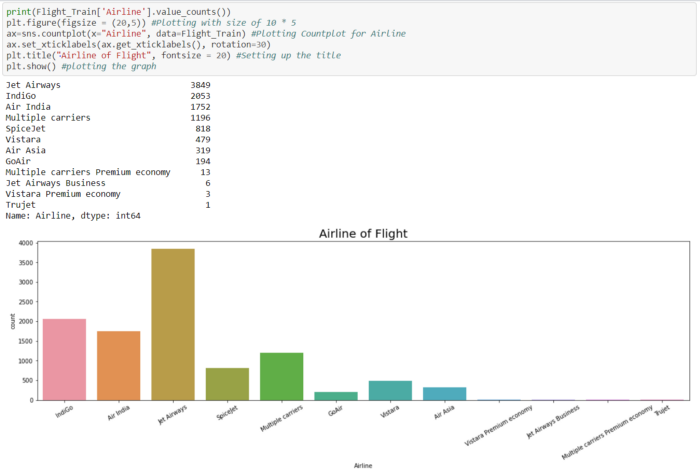
Most of the flights were having 1stop i.e. 5626 or 0stops i.e. 3491

Only 1 Flight stopped at 4 places, 45 Flight stopped at 3 places and 1520 Flights stopped at 2 places.



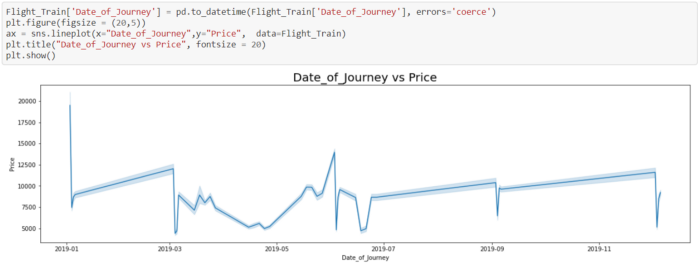
Generally there was no Information available but when it’ll available it’ll be useful.

1982 times meal not included in flight and 320 time check-in baggage was not included

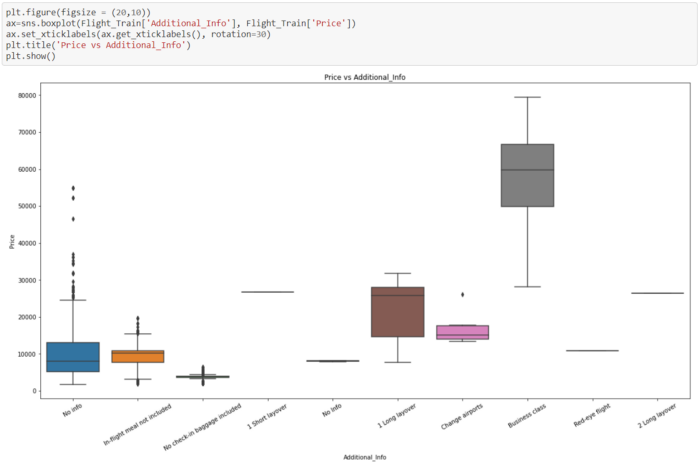


Mostly Jet Airways Airlines Flights fly in sky.

Jet Airways Airlines flights flied 3849 times where as Trujet flight flied 1 time.

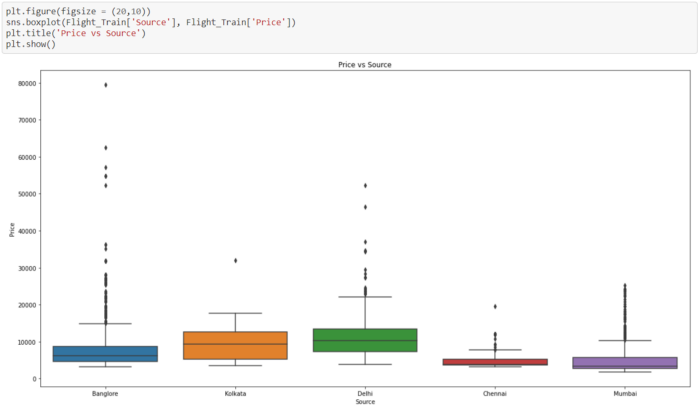


In January starting Flight prices were high(20,000) than suddenly in dropped to 7500. After that Flight prices slightly increasing from Jan to march 8000 to 12500. In March and April flight prices are slightly decreasing and increasing multiple times. From May to June prices are going high. June and July prices are going up and down. July till September price increasing suddenly price down and than again it’s going high till mid of November.



Here few things can be noticeable -

1. The Flight price is too low when No check-in baggage were allowed.
2. When customer choose Business Class that time Price goes too high.
3. When No meal provided in Flight that time flight prices are always lesser than 20,000.

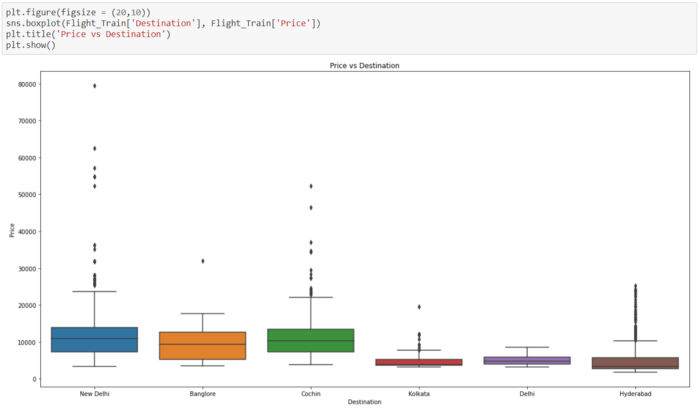


Here few things can be noticeable -

From Chennai and Mumbai the flights are cheaper as compared with other Sources.

Flights which fly from Kolkata have price less than 20k and only one time went near 32k.

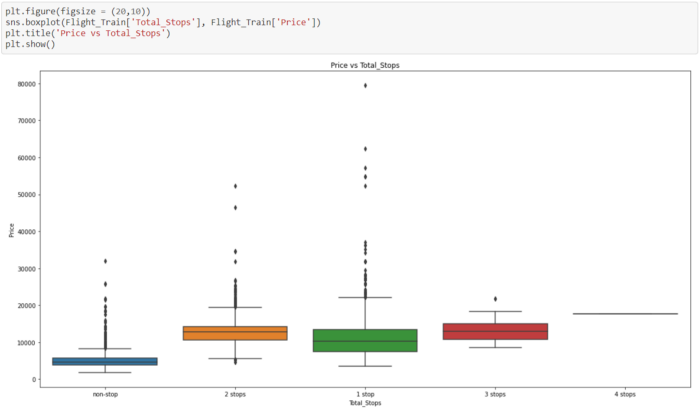
Flight which fly from Banglore have flight prices too high.



Kolkata, Delhi and Hyderabad flights are cheaper than other Destination price.

New Delhi flights fare are too high where as Cochin flights fare are also high.

Banglore flight prices are always lesser than 20k but only time it went to 32k.



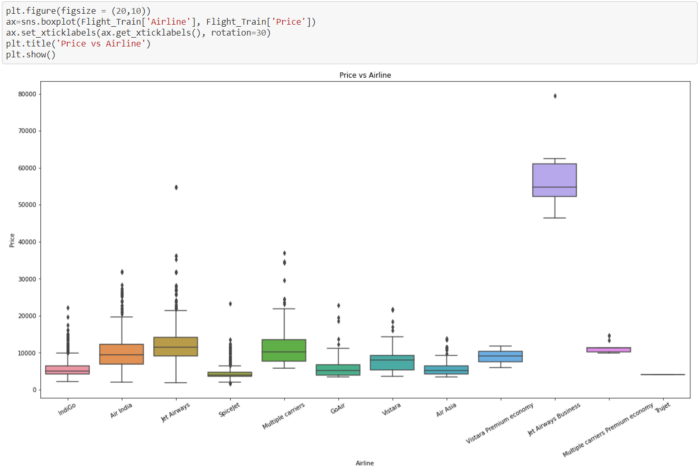
Here few things can be noticeable -

When number of stops are 3 that time prices goes around 20k

In case of 2 and 3 stops flight prices goes more than 30k many times.

When stops are 0 that time flight prices are very less.

Non-Stops flights are cheaper as compared to other flights.

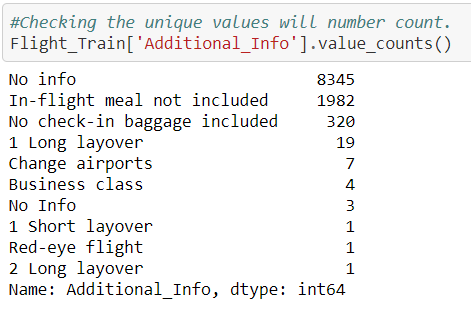


Jet Airways Business flights are always expensive and SpiceJet flights are cheaper.

Further EDA can be seen after Pre-Processing and Data Cleaning.

**Pre-Processing Pipeline**

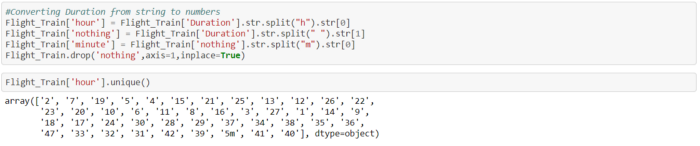
Here will do extraction of the information from all the columns which will help to do better and accurate prediction. In Additional\_Info when some information is given that highly reflect to flight price because When a person travel on business class that is additional information, fare of the flight goes too high and when No check-in baggage included fare prices goes too low. Total number of stops will also help to predict the fare of flight, When flights are non stop, fares are too low, when 1 stop is there is journey price increases and so on. Duration of the journey is one the biggest factor which can help to predict the fare of flight. When the duration goes high, cost of travel increases that impact on flight fare directly.



It’s clearly visible that No Info is No info, it’s just typo error.

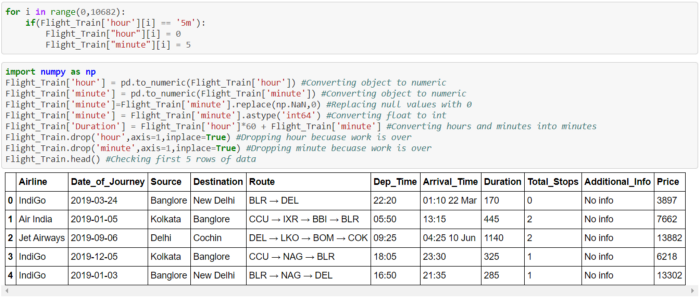


In order to do good and accurate prediction, need to get Duration of journey in minutes from hours and minutes. That will give us journey time on one scale and it’ll boost up the accuracy of the project.



5m came in hour, need to convert 5m into minutes and hours to 0

In order to get journey duration correctly had to choose this long way. There are few inbuild function which will gives us journey duration but that’s failing in few scenario’s.

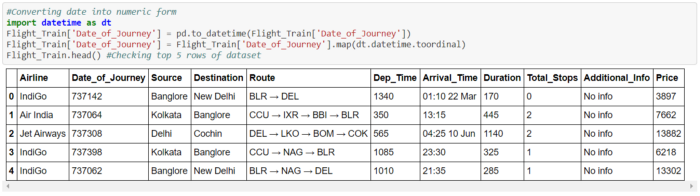


It’s clearly visible that Duration is converted into minutes.

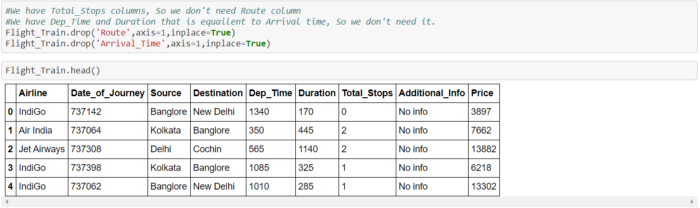
*#Converting Departure time into minutes*

*Flight\_Train[‘Dep\_Time’] = pd.to\_datetime(Flight\_Train[‘Dep\_Time’],format = ‘%H:%M’).dt.hour\*60 + pd.to\_datetime(Flight\_Train[‘Dep\_Time’],format = ‘%H:%M’).dt.minute*

Dep\_Time converted into minutes that will help for better prediction



Converted date into numbers using datetime library



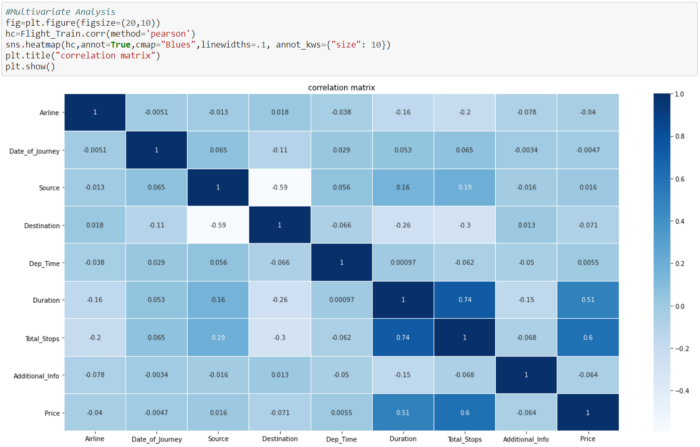
Dropped unnecessary columns, which are not useful for prediction

Label Encoding

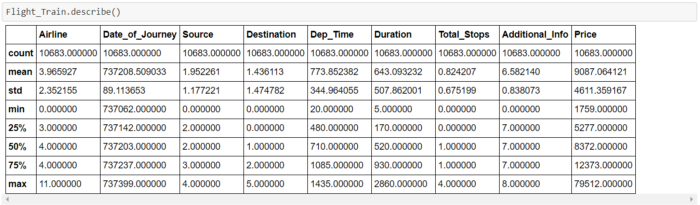
#Label encoding for the object columns*label\_list=list(Flight\_Train.select\_dtypes([‘object’]).columns)****from******sklearn.preprocessing******import****LabelEncoder le=LabelEncoder()*#Initlize LabelEncoder to le ***for****i****in****label\_list: Flight\_Train[i] = le.fit\_transform(Flight\_Train[i])*

Converted all the non numeric columns into numbers

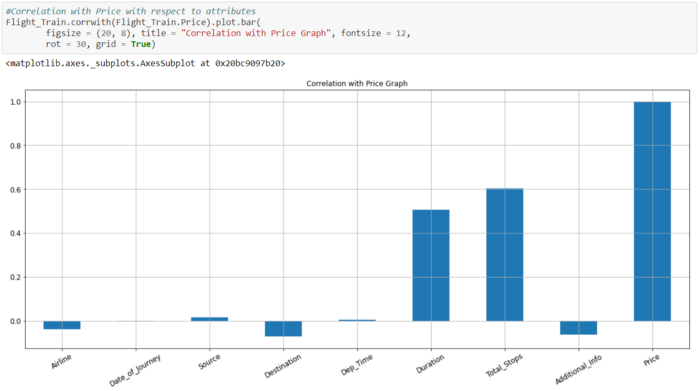
Looking into Correlation of other variables with respect to Price i.e. target variable.



Price is highly corelated with Duration and Total\_Stops



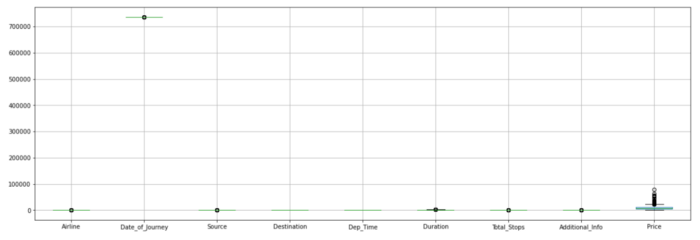
There is difference between Mean and standard deviation of Price, So Outliers must be present in it.



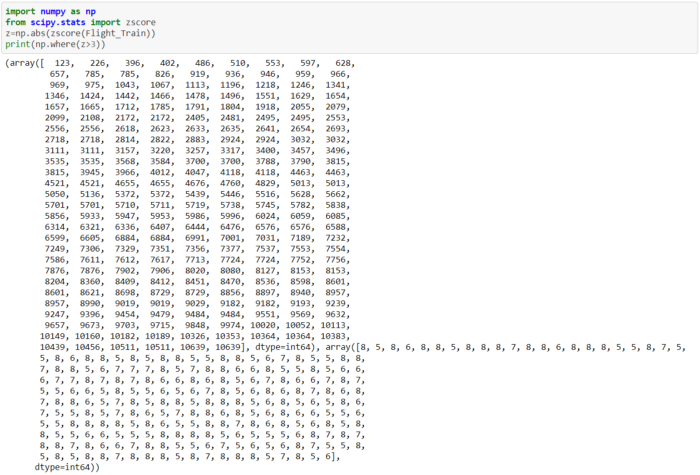
As seen earlier Duration and Total\_Stops are highly correlated with Price

Removing Skewness and Outliers

import matplotlib.pyplot as plt  
import seaborn as sns  
Flight\_Train.boxplot(figsize=[20,8])  
plt.subplots\_adjust(bottom=0.25)  
plt.show()

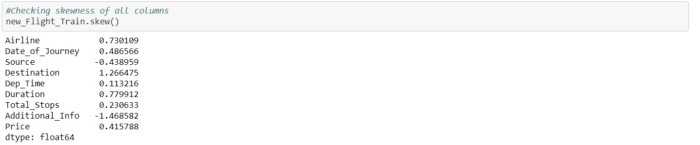


It’s clearly visible that outliers are present in Price.





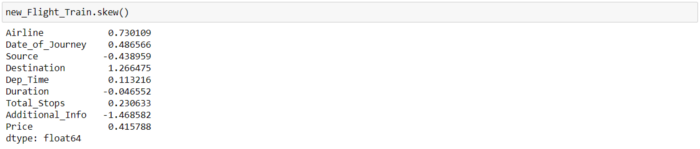
2% data is lost, it’s acceptable



Date\_of\_Journey, Dep\_Time, Duration, Total\_Stops and Price are numeric. In that skewness of Duration is high. Need to reduce it.

https://miro.medium.com/max/700/1*xOYWT4m4WHjchhyYPmNB8g.png

Reduced the skewness of Duration



Now Skewness of all columns are good.

**Building Machine Learning Models**

1. **Separating Input and Output Variables**

*y = new\_Flight\_Train[“Price”]*

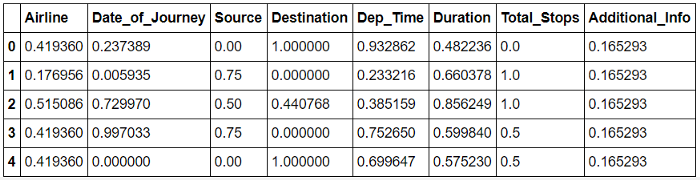
*x = new\_Flight\_Train.drop([“Price”], axis=1)*

Separated x and y variables to train and testthe data

**2. Scaling**

Scaling is required because there are huge difference in values of each columns.

**from** **sklearn.preprocessing** **import** MinMaxScaler  
scale = MinMaxScaler() *#Initializting MinMaxScaler*  
new = scale.fit(x) *#fitting our data into MinMaxScaller*  
scale\_x = new.transform(x) *#Transforming the data*  
*#Setting up the coulumns after Scaling*  
scaled\_x = pd.DataFrame(scale\_x, index=x.index, columns=x.columns)  
x=scaled\_x  
x.head() *#Priting top 5 rows of our data*



After scaling all the variables have value 0–1

**3. Finding Best Random State**

Here Linear Regression is used to get the Random state on which model is working more accurate.

**from** **sklearn.model\_selection** **import** train\_test\_split  
**from** **sklearn.linear\_model** **import** LinearRegression  
**from** **sklearn.metrics** **import** r2\_score  
maxR2\_Score=0  
maxRS=0  
**for** i **in** range(1,1000):  
 x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=.30,random\_state=i)  
 LR = LinearRegression()  
 LR.fit(x\_train, y\_train)  
 predrf = LR.predict(x\_test)  
 score = r2\_score(y\_test, predrf)  
 **if** score>maxR2\_Score:  
 maxR2\_Score=score  
 maxRS=i  
print("Best accuracy is",maxR2\_Score," on Random\_state ",maxRS)

Best accuracy is 54.33% at 360 Random state

**4. Train Test Split**

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,test\_size=.30,random\_state=maxRS)

Splitting train data and test data

**5. Finding Best Algorithm**

Using Random Forest, Decission Tree, K Neighbours, Gradient Boosting, Ridge and SVR to get best algorithm out of them.

**from** **sklearn.metrics** **import** mean\_squared\_error,r2\_score,mean\_absolute\_error  
**import** **numpy** **as** **np**  
**from** **sklearn.ensemble** **import** RandomForestRegressor  
**from** **sklearn.tree** **import** DecisionTreeRegressor  
**from** **sklearn.neighbors** **import** KNeighborsRegressor  
**from** **sklearn.ensemble** **import** GradientBoostingRegressor  
**from** **sklearn.linear\_model** **import** Ridge  
**from** **sklearn.svm** **import** SVR  
**from** **sklearn.model\_selection** **import** cross\_val\_scoremodel=[LinearRegression(),RandomForestRegressor(), DecisionTreeRegressor(),KNeighborsRegressor(), GradientBoostingRegressor(),Ridge(),SVR()]  
print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")  
 print('accuracy score of ->', m)  
 m.fit(x\_train,y\_train)  
 pred = m.predict(x\_test)  
 print("R2 Score: ", r2\_score(y\_test,pred))  
 print("Mean Absolute Error: ", mean\_absolute\_error(y\_test,pred))  
 print("Mean Squared error: ", mean\_squared\_error(y\_test,pred))  
 print("Root Mean Squared Error: ", np.sqrt(mean\_squared\_error(y\_test,pred)))  
 score=cross\_val\_score(m,x,y,cv=8)  
 print(score)  
 print("cross validation score: ",score.mean())  
 print("Difference between R2 score and cross validatio score is - ",r2\_score(y\_test,pred)-abs(score.mean()))  
 print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

Output

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> LinearRegression()  
R2 Score: 0.5433243452764487  
Mean Absolute Error: 2157.0645389788497  
Mean Squared error: 7470645.752834696  
Root Mean Squared Error: 2733.24820549373  
[0.5218934 0.51507901 0.54378328 0.5228583 0.49113157 0.51343271  
 0.48271281 0.5369189 ]  
cross validation score: 0.515976248246264  
Difference between R2 score and cross validation score is - 0.027348097030184704  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> RandomForestRegressor()  
R2 Score: 0.885903554464673  
Mean Absolute Error: 767.0572819190037  
Mean Squared error: 1866475.9494744928  
Root Mean Squared Error: 1366.190304999451  
[0.88915617 0.90513084 0.89481681 0.89084888 0.87334998 0.88657588  
 0.88511913 0.8767304 ]  
cross validation score: 0.887716011984078  
Difference between R2 score and cross validation score is - -0.001812457519405064  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> DecisionTreeRegressor()  
R2 Score: 0.826820494508945  
Mean Absolute Error: 821.8988227807827  
Mean Squared error: 2833001.3299218724  
Root Mean Squared Error: 1683.1522004625347  
[0.80775438 0.8430543 0.80841163 0.83173055 0.8087391 0.79911125  
 0.82199225 0.80462527]  
cross validation score: 0.8156773412615379  
Difference between R2 score and cross validation score is - 0.011143153247407112  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> KNeighborsRegressor()  
R2 Score: 0.7579099612721012  
Mean Absolute Error: 1275.641425389755  
Mean Squared error: 3960291.951014954  
Root Mean Squared Error: 1990.0482283138151  
[0.77432135 0.77088439 0.7989919 0.77039909 0.76446026 0.74411146  
 0.74516321 0.75672783]  
cross validation score: 0.7656324369849502  
Difference between R2 score and cross validation score is - -0.0077224757128490085  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> GradientBoostingRegressor()  
R2 Score: 0.8158497976839774  
Mean Absolute Error: 1235.1327113445648  
Mean Squared error: 3012468.285940572  
Root Mean Squared Error: 1735.6463597002046  
[0.81711833 0.82080217 0.8298909 0.83029614 0.81273099 0.80307627  
 0.81181412 0.81836611]  
cross validation score: 0.8180118784038888  
Difference between R2 score and cross validation score is - -0.002162080719911419  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> Ridge()  
R2 Score: 0.5432707905353087  
Mean Absolute Error: 2157.21681158434  
Mean Squared error: 7471521.84179478  
Root Mean Squared Error: 2733.4084659623745  
[0.52207565 0.51480954 0.54364364 0.52278706 0.49126627 0.51340793  
 0.48270987 0.53708249]  
cross validation score: 0.5159728068760017  
Difference between R2 score and cross validation score is - 0.027297983659307024  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
accuracy score of -> SVR()  
R2 Score: 0.1089793836924835  
Mean Absolute Error: 3164.56907892165  
Mean Squared error: 14575989.138145354  
Root Mean Squared Error: 3817.8513771682306  
[0.14289441 0.1008356 0.12298676 0.12321475 0.12151827 0.14127857  
 0.13084478 0.13082065]  
cross validation score: 0.1267992226985708  
Difference between R2 score and cross validation score is - -0.0178198390060873  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Random Forest Regressor model have highest accuracy i.e. 88.59% with 88.77% cross validation score which is good and the difference is too less.

**6. Hyper Parameter Tuning**

***from******sklearn.model\_selection******import****GridSearchCV  
  
parameters = {"max\_depth":range(21,25),  
"criterion":['mse'],  
"max\_features": ['auto', 'sqrt'],  
"min\_samples\_leaf":range(1,5)}  
  
clf = GridSearchCV(RandomForestRegressor(), parameters)  
clf.fit(x\_train,y\_train)*#fitting train and test data *clf.best\_params\_*#Best parameters

*#Output*

*{'criterion': 'mse',  
'max\_depth': 22,  
'max\_features': 'auto',  
'min\_samples\_leaf': 1}*

*clf\_pred=clf.best\_estimator\_.predict(x\_test)  
r2\_score(y\_test, clf\_pred)*

*#Output  
0.8861565029246709*

Now the model learnt almost 89%, which is a good score.

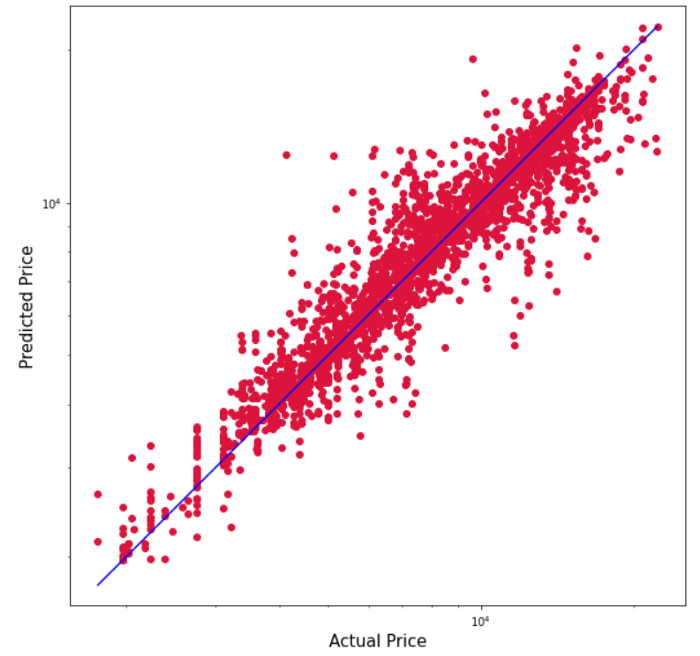
**7. Saving the model**

There is always good practice to save the learnt model because in real world only saved object file will be required and called from the website etc. to predict new data.

**import** **joblib**  
joblib.dump(clf.best\_estimator\_,"Flight.obj")  
RF\_from\_joblib=joblib.load('Flight.obj')  
Predicted = RF\_from\_joblib.predict(x\_test)  
Predicted#Output  
array([10480.86 , 10515.74 , 12352.53 , ...,  
 7747.29033333, 9315.82718222, 4954.65 ])

Saving the learnt model into Flight.obj file and reading back object file predicting the test data

plt.figure(figsize=(10,10))  
plt.scatter(y\_test, Predicted, c='crimson')  
plt.yscale('log')  
plt.xscale('log')  
p1 = max(max(Predicted), max(y\_test))  
p2 = min(min(Predicted), min(y\_test))  
plt.plot([p1, p2], [p1, p2], 'b-')  
plt.xlabel('Actual Price', fontsize=15)  
plt.ylabel('Predicted Price', fontsize=15)  
plt.axis('equal')  
plt.show()



Graph of Predicted Price and Actual Price

**Concluding Remarks**

In this case study, a Machine Learning model is developed to predict the airlines fare. Here several features were mined from the dataset and combined together with the help of Machine Leaning, to do the flight price prediction. With the help of the above techniques, proposed model is able to predict the flight fare with an adjusted R squared score of 88.59%. However, there is still ways to do improvement in this model.

In the future, our model can be predict the flight fare more accurately, if we get some of information such as seat location, when ticket was booked, special occasion on departure date etc.