FLIP ROBO

NAME OF THE PROJECT

FLIGHT PRICE PREDICTION PROJECT

Submitted by:

Yogesh.C.Mudliar

ACKNOWLEDGMENT

It is a genuine pleasure to express my deep sense of thanks and gratitude to my guide, Ms.Khushboo Garg, for allowing me to work on this project. It was a great way to expose myself to the actual research environment.

I thank FLIPROBO for permitting me to work with them.
I take this opportunity to say heartfelt thanks to Dr. Deepika Sharma, VP-learning And development DataTraind for her overall dedication, devotion, and support towards me. I convey my sincere regards to all the DataTraind team thanks for supporting me during academic years of my post-graduation course in data science.

I express my profound sense of gratitude to my mentor Ms.Khushboo Garg, FLIPROBO for her guidance at every step of my research work.

Apart from the project, I learned a lot from her, she gave me valuable thought- "To think"; that I will benefit from, for a long time to come. I am indebted to her more than she knows.

INTRODUCTION

Business Problem Framing

Anyone who has booked a flight ticket knows how unexpectedly the prices vary. The cheapest available ticket on a given flight gets more and less expensive over time. This usually happens as an attempt to maximize revenue based on –

1. Time of purchase patterns (making sure last-minute purchases are expensive)

2. Keeping the flight as full as they want it (raising prices on a flight which is filling up in order to reduce sales and hold back inventory for those expensive last-minute expensive purchases)

So, you have to work on a project where you collect data of flight fares with other features and work to make a model to predict fares of flights

Conceptual Background of the Domain Problem

The cheapest available ticket on a given flight gets more and less expensive over time. Flights do generally get more expensive the closer you get to flight time. This is because seats at the lowest fare levels sell out as a flight receives more and more bookings.

Review of Literature

flight ticket prices change during the morning and evening time of the day. Also, it changes with the holidays or festival season. There are several different factors on which the price of the flight ticket depends. The seller has information about all the factors, but buyers are able to access limited information only which is not enough to predict the airfare prices. Considering the features such as departure time, the number of days left for departure and time of the day it will give the best time to buy the ticket. The purpose of the paper is to study the factors which influence the fluctuations in the airfare prices and how they are related to the change in the prices. Then using this information, build a system that can help buyers whether to buy a ticket or not.

- Motivation for the Problem Undertaken

It is very difficult for the customer to purchase a flight ticket at the minimum price. For this several techniques are used to obtain the day at which the price of air ticket will be minimum. Most of these techniques are using sophisticated artificial intelligence (AI) research is known as Machine Learning.

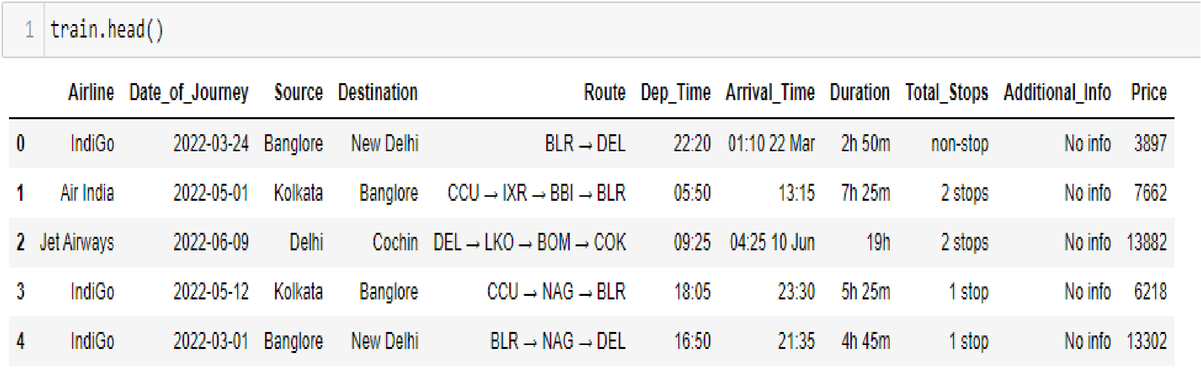
Analytical Problem Framing

- ## Mathematical/ Analytical Modeling of the Problem

I done Extensive EDA has to be performed to gain relationships of important variable and price. I used RandomForestRegressor (sklearn.ensemble RandomForestRegressor) analysis A random forest is a meta estimator that fits a number of classifying decision trees on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The variable you want to predict is called the dependent variable. The variable you are using to predict the other variable's value is called the independent variable.

- ## Data Sources and their formats

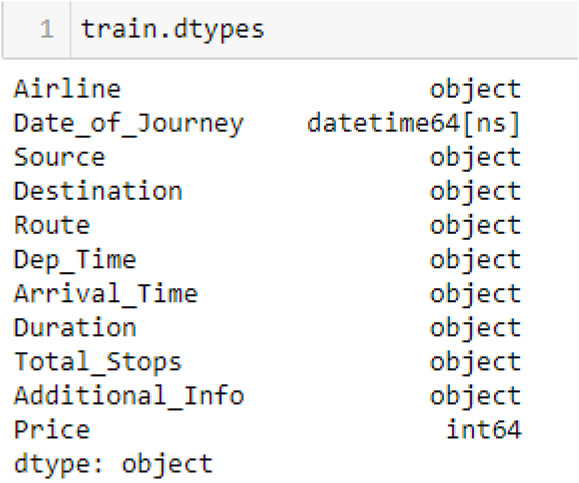
I scrape this data having 10683 rows and 11 columns



```
train.head()
```

	Airline	Date_of_Journey	Source	Destination	Route	Dep_Time	Arrival_Time	Duration	Total_Stops	Additional_Info	Price
0	IndiGo	2022-03-24	Banglore	New Delhi	BLR → DEL	22:20	01:10 22 Mar	2h 50m	non-stop	No info	3897
1	Air India	2022-05-01	Kolkata	Banglore	CCU → IXR → BBI → BLR	05:50	13:15	7h 25m	2 stops	No info	7662
2	Jet Airways	2022-06-09	Delhi	Cochin	DEL → LKO → BOM → COK	09:25	04:25 10 Jun	19h	2 stops	No info	13882
3	IndiGo	2022-05-12	Kolkata	Banglore	CCU → NAG → BLR	18:05	23:30	5h 25m	1 stop	No info	6218
4	IndiGo	2022-03-01	Banglore	New Delhi	BLR → NAG → DEL	16:50	21:35	4h 45m	1 stop	No info	13302

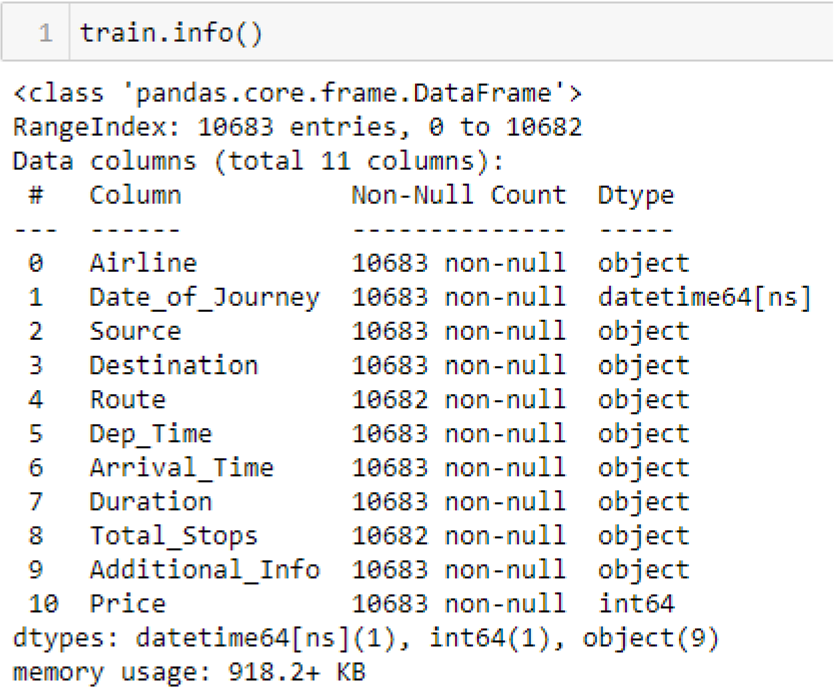
Data types:



```
train.dtypes
```

```
Airline                    object
Date_of_Journey    datetime64[ns]
Source                     object
Destination                object
Route                      object
Dep_Time                   object
Arrival_Time               object
Duration                   object
Total_Stops                object
Additional_Info            object
Price                       int64
dtype: object
```

Data information:



```
train.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  datetime64[ns]
 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: datetime64[ns](1), int64(1), object(9)
memory usage: 918.2+ KB
```

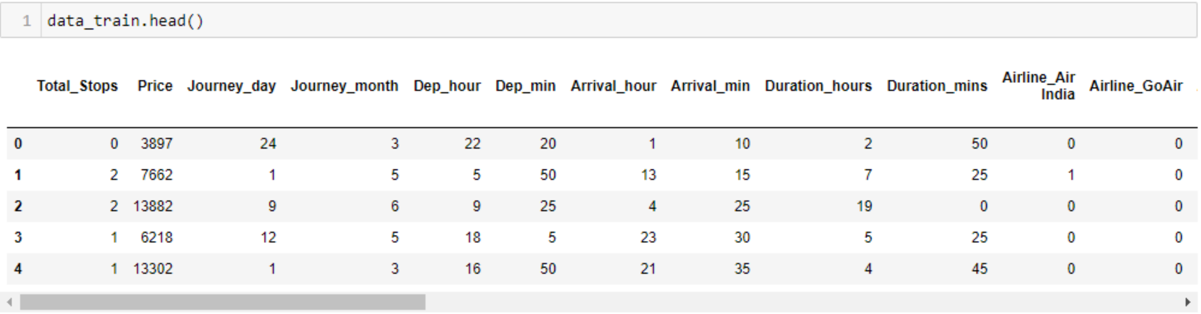
- Data Preprocessing Done

After collecting a data, the next step is to get it ready for analysis. This means cleaning, or ‘scrubbing’ it, and is crucial in making sure that you’re working with high quality data.

- **Removing unwanted data points**—extracting irrelevant observations that have no bearing on intended analysis.
- **Bringing structure to data**—fixing layout issues, which will help to map and manipulate this data more easily.
- **Filling in major gaps**—this data contains null values and I notice that important data are missing. Once we identified gaps, we can go about filling them.

During cleaning the data we have to do EDA(Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations). This helps identify initial trends and characteristics, and can even refine our hypothesis.

After cleaning dataset I got this data having 10682 rows and 30 columns.



```
data_train.head()
```

	Total_Stops	Price	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_Air India	Airline_GoAir
0	0	3897	24	3	22	20	1	10	2	50	0	0
1	2	7662	1	5	5	50	13	15	7	25	1	0
2	2	13882	9	6	9	25	4	25	19	0	0	0
3	1	6218	12	5	18	5	23	30	5	25	0	0
4	1	13302	1	3	16	50	21	35	4	45	0	0

• Data Inputs- Logic- Output Relationships

There are some inputs which is important to find our outputs like the price of flight with the available independent variables.

We have different Airline names, Date of Journey, Source (from where the Flight is departure, Destination, Route, departure Time, Arrival Time, Duration, Total Stops, Additional Info, Price.

• Hardware and Software Requirements and Tools Used

Software requirements:

numpy : library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

pandas : software library written for the Python programming language for data manipulation and analysis

sklearn: Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support-vector machines

seaborn: Seaborn is a library that uses Matplotlib underneath to plot graphs. It will be used to visualize random distributions.

matplotlib.pyplot : is a plotting library for the Python programming language and its numerical mathematics extension NumPy.

sklearn.metrics _ mean_squared_error,mean_absolute_error

sklearn.model_selection _ train_test_split: to get training and test sets

sklearn.impute import SimpleImputer: scikit-learn class which is helpful in handling the missing data in the predictive model dataset.

Sklearn_ preprocessing: provides several common utility functions and transformer classes to change raw feature vectors into a representation that is more suitable for the downstream estimators.

sklearn.preprocessing import LabelEncoder: to normalize labels. It can also be used to transform non-numerical labels (as long as they are hashable and comparable) to numerical labels.

Model/s Development and Evaluation

Identification of possible problem-solving approaches (methods)

After collecting a data, the next step is to get it ready for analysis. This means cleaning, or 'scrubbing' it, and is crucial in making sure that you're working with high quality data.

- Removing unwanted data points—extracting irrelevant observations that have no bearing on intended analysis.
- Bringing structure to data—fixing layout issues, which will help to map and manipulate this data more easily.
- Filling in major gaps—This data contains null values and I notice that important data are missing. Once we identified gaps, we can go about filling them.

During cleaning the data we have to do EDA(Exploratory Data Analysis (EDA) is an approach to analyse the data using visual techniques. It is used to discover trends, patterns, or to check assumptions with the help of statistical summary and graphical representations). This helps identify initial trends and characteristics, and can even refine our hypothesis.

I done Predictive analysis (Predictive analytics encompasses a variety of statistical techniques from data mining, predictive modelling, and machine learning that analyse current and historical facts to make predictions about future or otherwise unknown events)

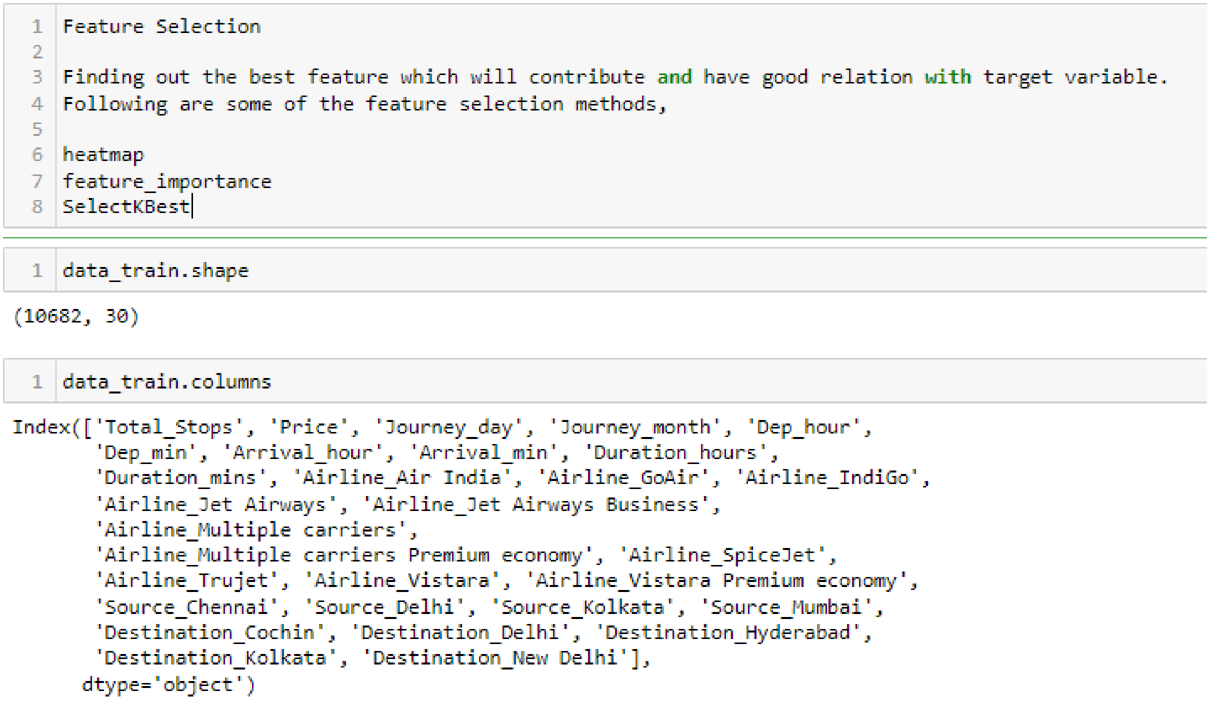
predictions about Which variables are important to predict the price of variable, and How do these variables describe the price of the flight tickets.

Testing of Identified Approaches (Algorithms)

The averaging makes a Random Forest better than a single Decision Tree hence improves its accuracy and reduces overfitting. A prediction from the Random Forest Regressor is an average of the predictions produced by the trees in the forest.

- Run and Evaluate selected models

Feature selections:



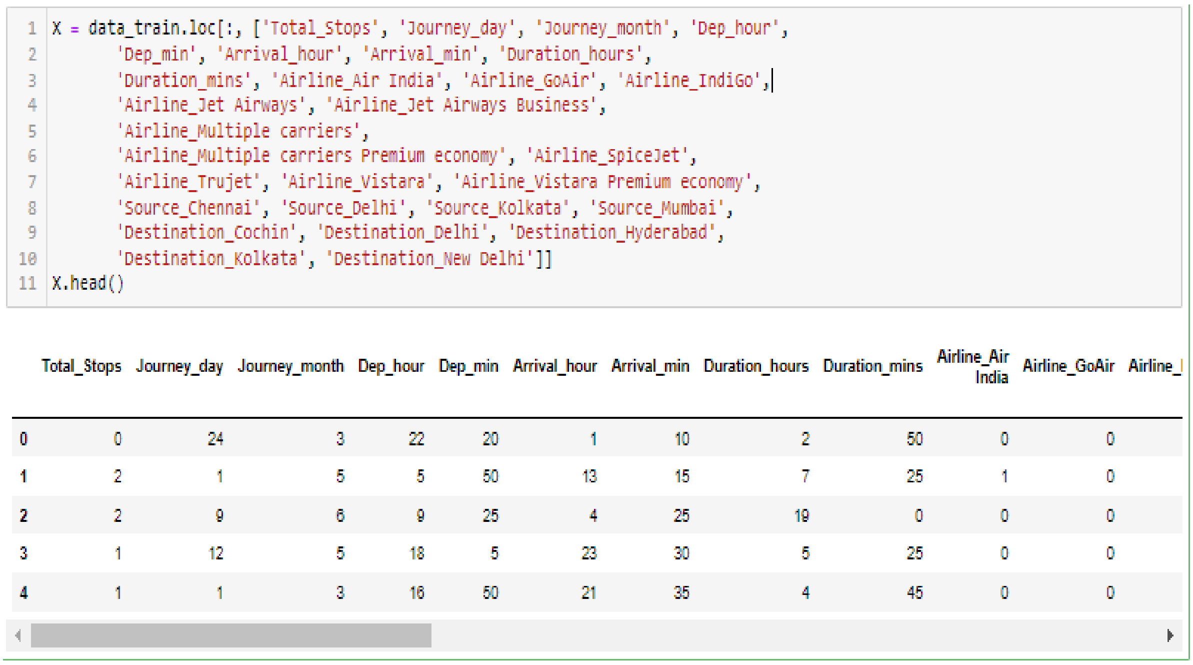
```
Feature Selection

Finding out the best feature which will contribute and have good relation with target variable.
Following are some of the feature selection methods,

heatmap
feature_importance
SelectKBest
```

```
data_train.shape
```

```
(10682, 30)
```

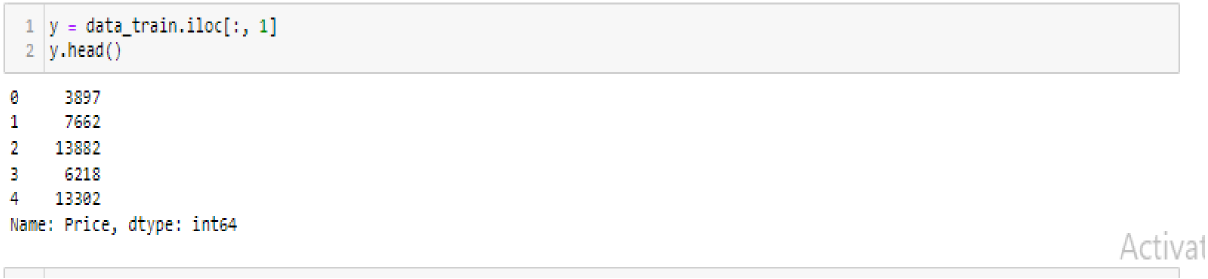
```
data_train.columns
```

```
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',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi'],
      dtype='object')
```



```
X = data_train.loc[:, ['Total_Stops', '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',
       'Source_Chennai', 'Source_Delhi', 'Source_Kolkata', 'Source_Mumbai',
       'Destination_Cochin', 'Destination_Delhi', 'Destination_Hyderabad',
       'Destination_Kolkata', 'Destination_New Delhi']]
X.head()
```

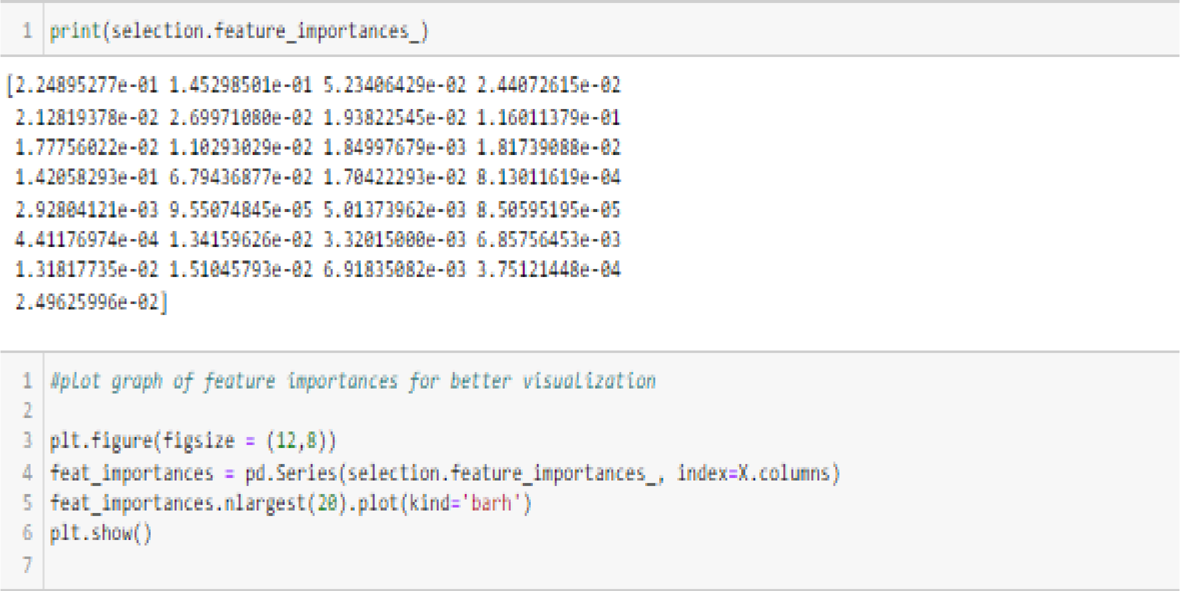
	Total_Stops	Journey_day	Journey_month	Dep_hour	Dep_min	Arrival_hour	Arrival_min	Duration_hours	Duration_mins	Airline_Air India	Airline_GoAir	Airline_
0	0	24	3	22	20	1	10	2	50	0	0	
1	2	1	5	5	50	13	15	7	25	1	0	
2	2	9	6	9	25	4	25	19	0	0	0	
3	1	12	5	18	5	23	30	5	25	0	0	
4	1	1	3	16	50	21	35	4	45	0	0	



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

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

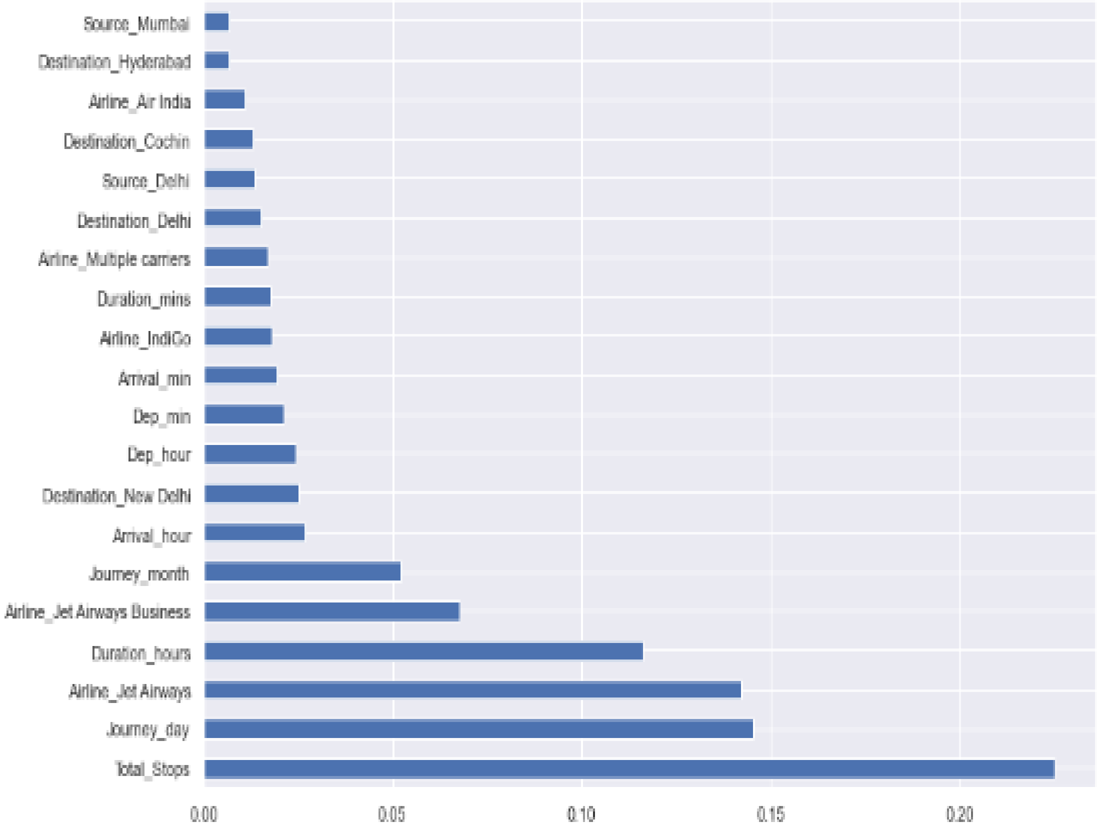
Plot graph of feature importance for better visualization:



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

```
[2.24895277e-01 1.45298501e-01 5.23406429e-02 2.44072615e-02
 2.12819378e-02 2.69971080e-02 1.93822545e-02 1.16011379e-01
 1.77756022e-02 1.10293029e-02 1.84997679e-03 1.81739088e-02
 1.42058293e-01 6.79436877e-02 1.70422293e-02 8.13011619e-04
 2.92804121e-03 9.55074845e-05 5.01373962e-03 8.50595195e-05
 4.41176974e-04 1.34159626e-02 3.32015000e-03 6.85756453e-03
 1.31817735e-02 1.51045793e-02 6.91835082e-03 3.75121448e-04
 2.49625996e-02]
```

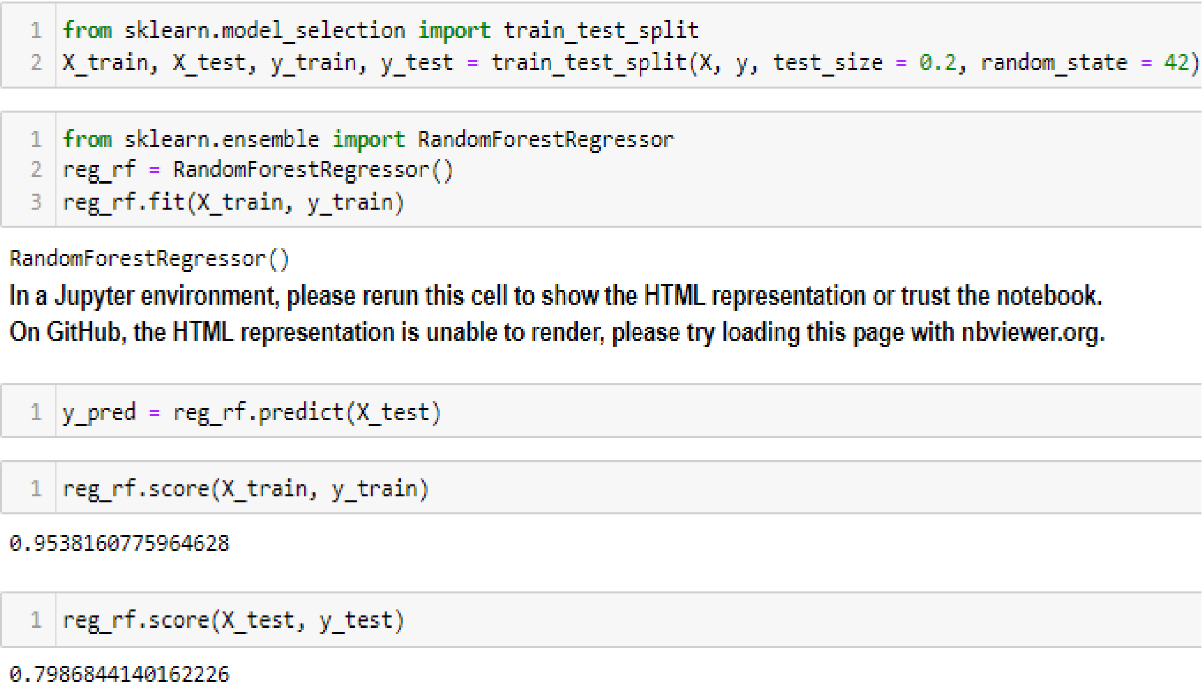
```
#plot graph of feature importances for better visualization

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

Split dataset into train and test set in order to prediction w.r.t X_test

-If needed do scaling of data -Import model -Fit the data -Predict w.r.t X_test -In regression check RSME Score -Plot graph



```python
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)
```

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

```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```python
y_pred = reg_rf.predict(X_test)
```

```python
reg_rf.score(X_train, y_train)
```

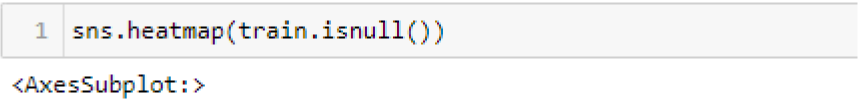
```
0.9538160775964628
```

```python
reg_rf.score(X_test, y_test)
```

```
0.7986844140162226
```

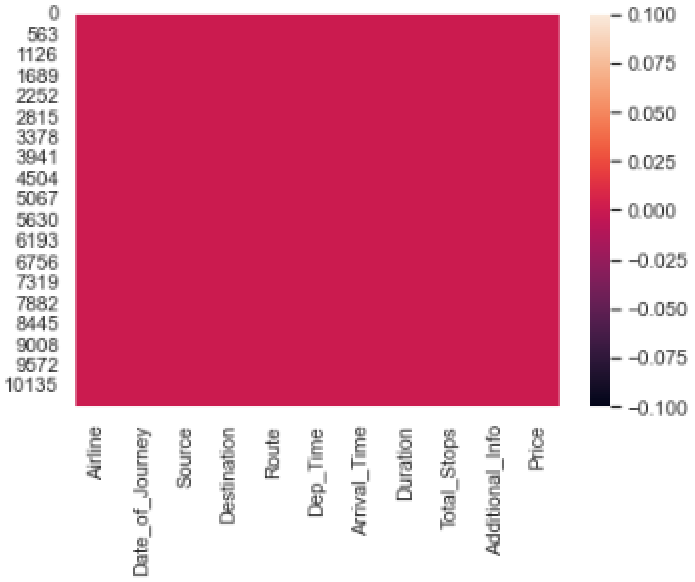
- Visualizations

Correlation heat map

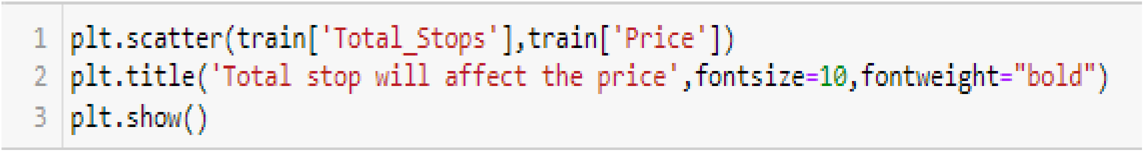


```
sns.heatmap(train.isnull())
```

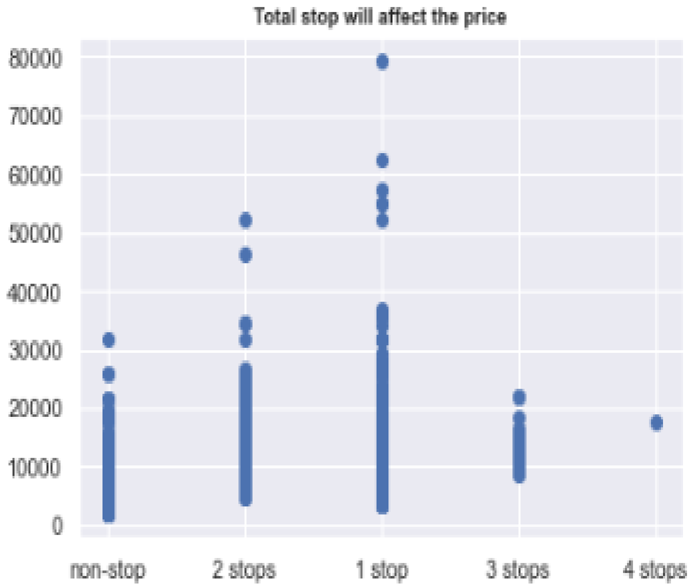
```
<AxesSubplot:>
```



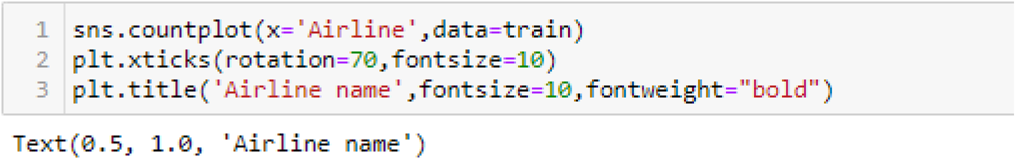
Graph plotted between total stops and price:



```
plt.scatter(train['Total_Stops'],train['Price'])
plt.title('Total stop will affect the price',fontsize=10,fontweight="bold")
plt.show()
```

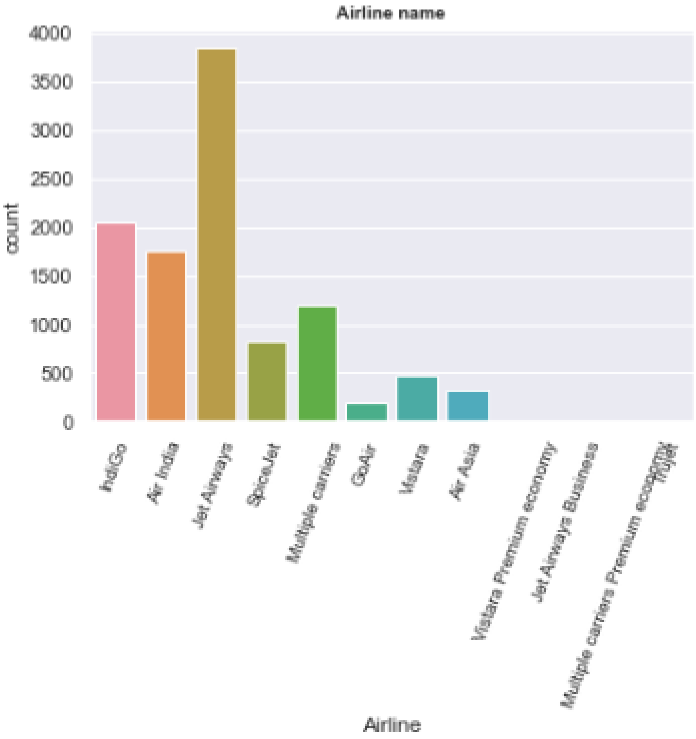


Airline count plot:



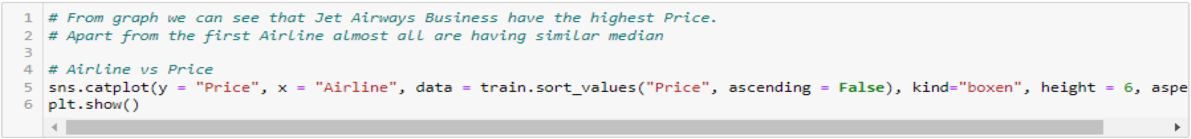
```
sns.countplot(x='Airline',data=train)
plt.xticks(rotation=70,fontsize=10)
plt.title('Airline name',fontsize=10,fontweight="bold")
```

```
Text(0.5, 1.0, 'Airline name')
```

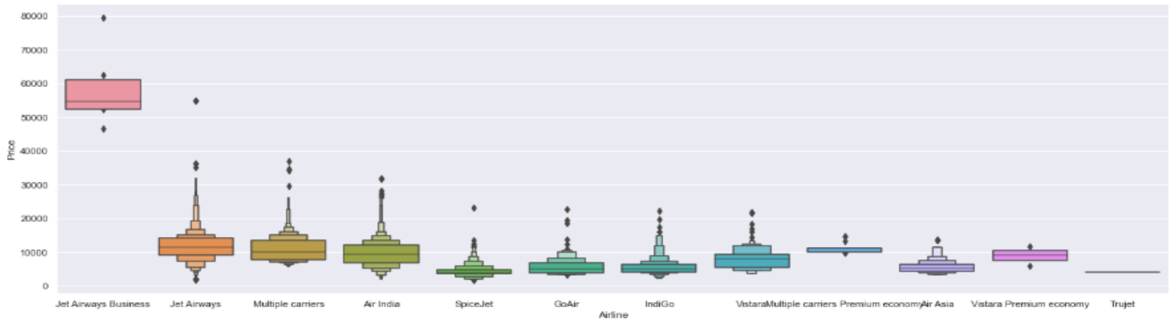


From graph we can see that Jet Airways Business have the highest Price.

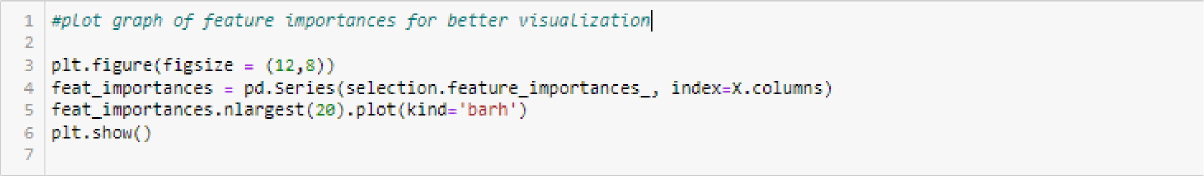
Apart from the first Airline almost all are having similar median:



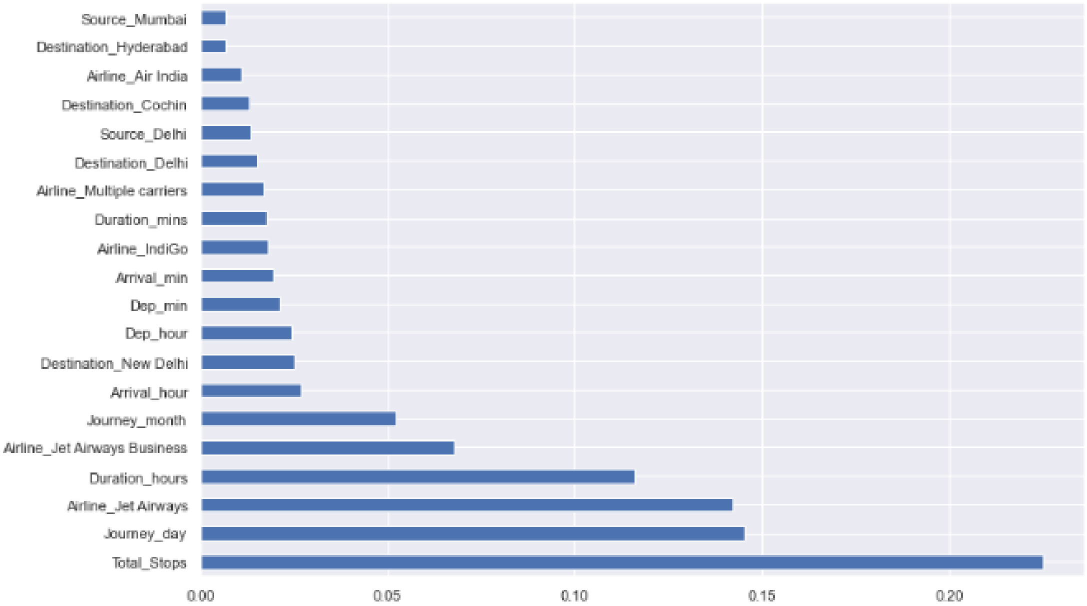
```
# From graph we can see that Jet Airways Business have the highest Price.
# Apart from the first Airline almost all are having similar median

# Airline vs Price
sns.catplot(y = "Price", x = "Airline", data = train.sort_values("Price", ascending = False), kind="boxen", height = 6, aspe
plt.show()
```

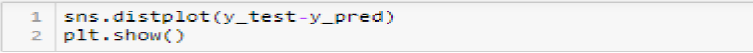


Plot graph of feature importance for better visualization:

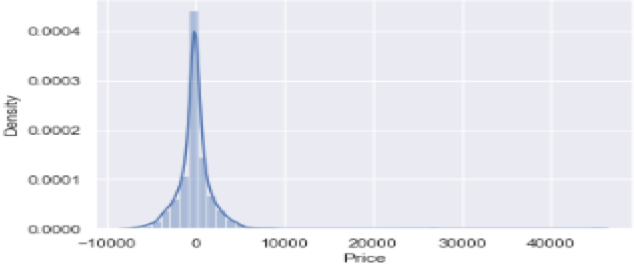


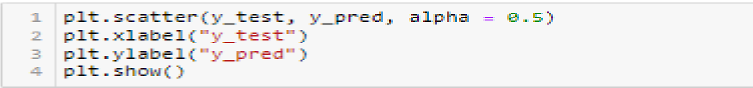
```python
#plot graph of feature importances for better visualization

plt.figure(figsize = (12,8))
feat_importances = pd.Series(selection.feature_importances_, index=X.columns)
feat_importances.nlargest(20).plot(kind='barh')
plt.show()

```



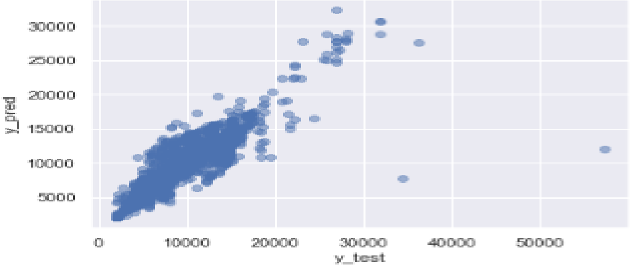


```python
sns.distplot(y_test-y_pred)
plt.show()
```

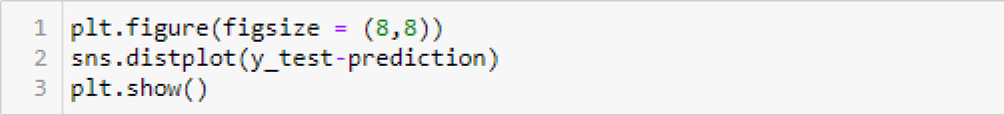




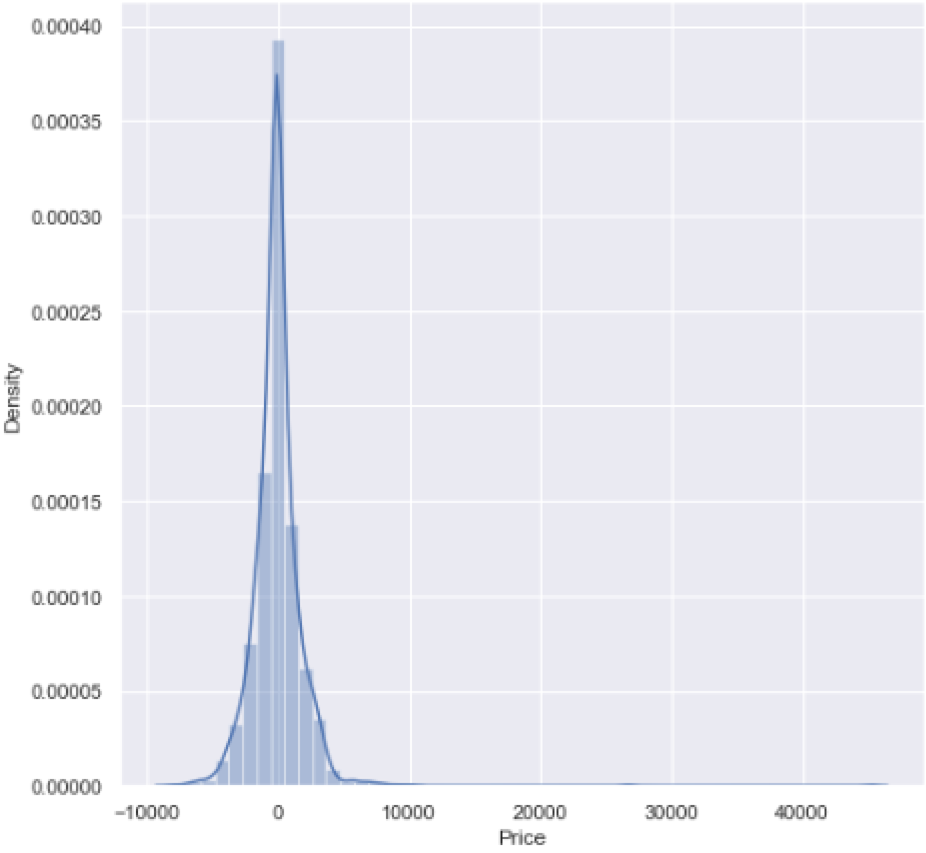
```python
plt.scatter(y_test, y_pred, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



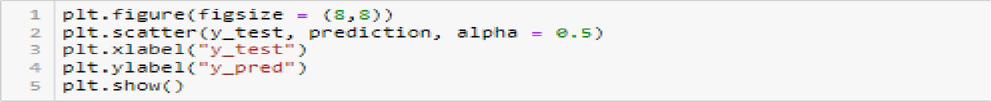
Distplot y test-prediction:



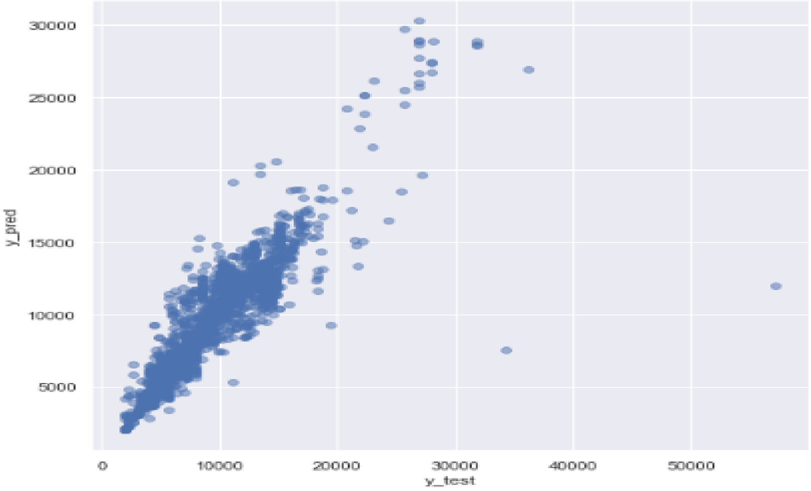
```
plt.figure(figsize = (8,8))
sns.distplot(y_test-prediction)
plt.show()
```



Scatter plot between y test and prediction:

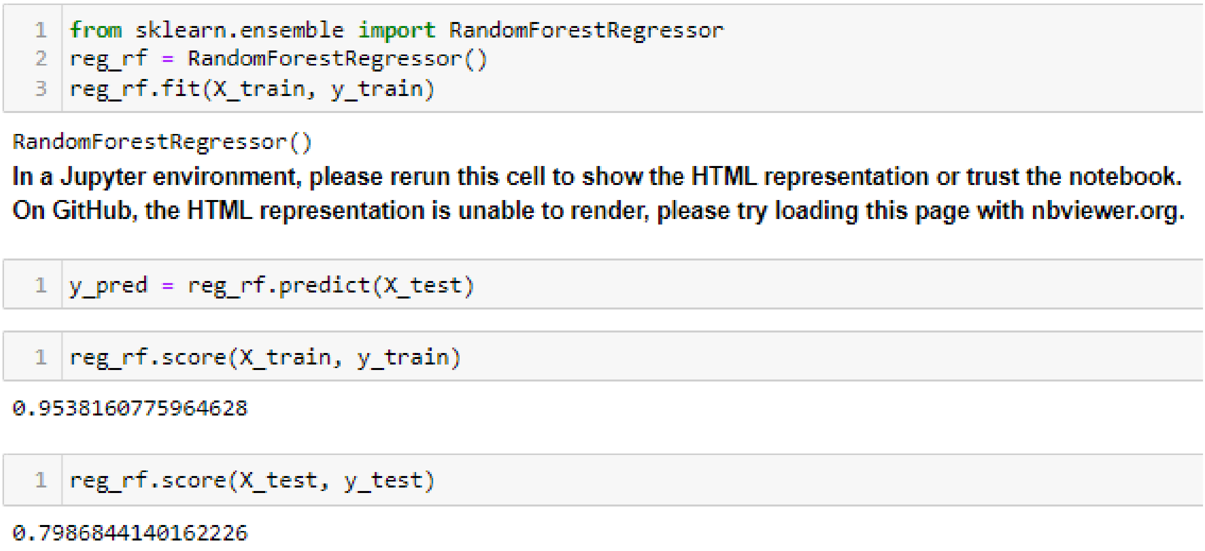


```
plt.figure(figsize = (8,8))
plt.scatter(y_test, prediction, alpha = 0.5)
plt.xlabel("y_test")
plt.ylabel("y_pred")
plt.show()
```



- ## Interpretation of the Results

We can see the train score and the test score:



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

```
RandomForestRegressor()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook. On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

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

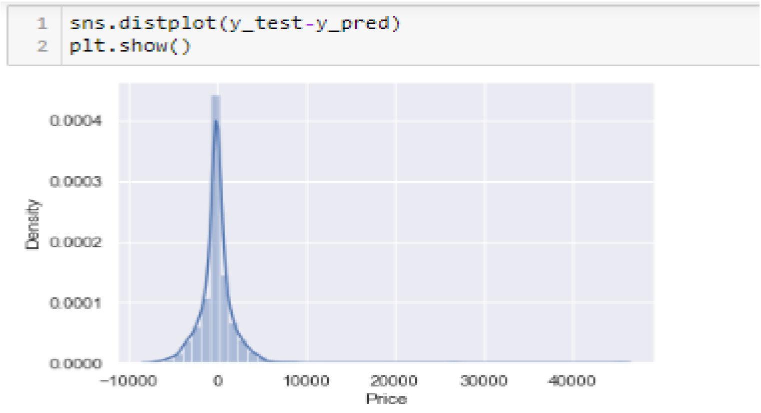
```
reg_rf.score(X_train, y_train)
```

```
0.9538160775964628
```

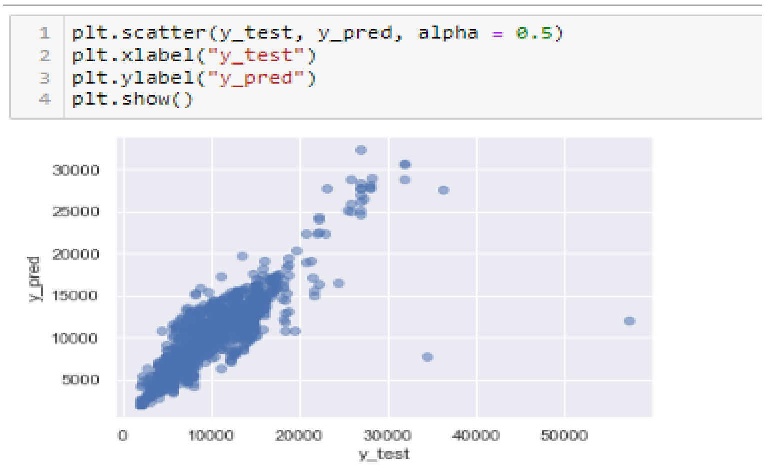
```
reg_rf.score(X_test, y_test)
```

```
0.7986844140162226
```

We can see the plot between y test and prediction:

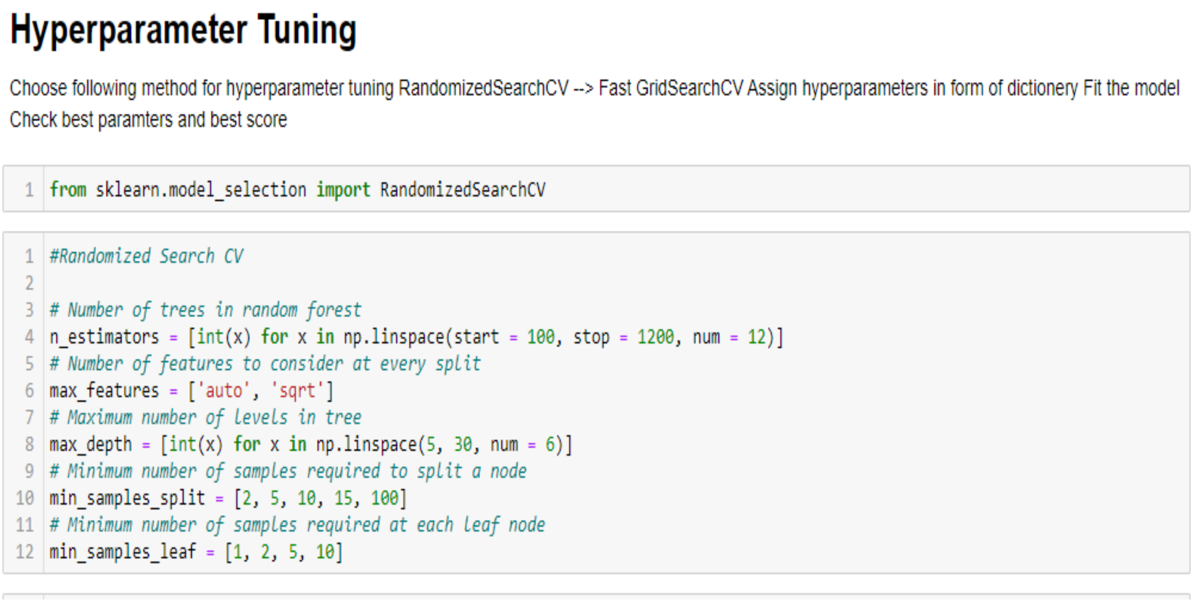


```
sns.distplot(y_test-y_pred)
plt.show()
```



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

For tuning our model we used Hyperparameter :



Hyperparameter Tuning

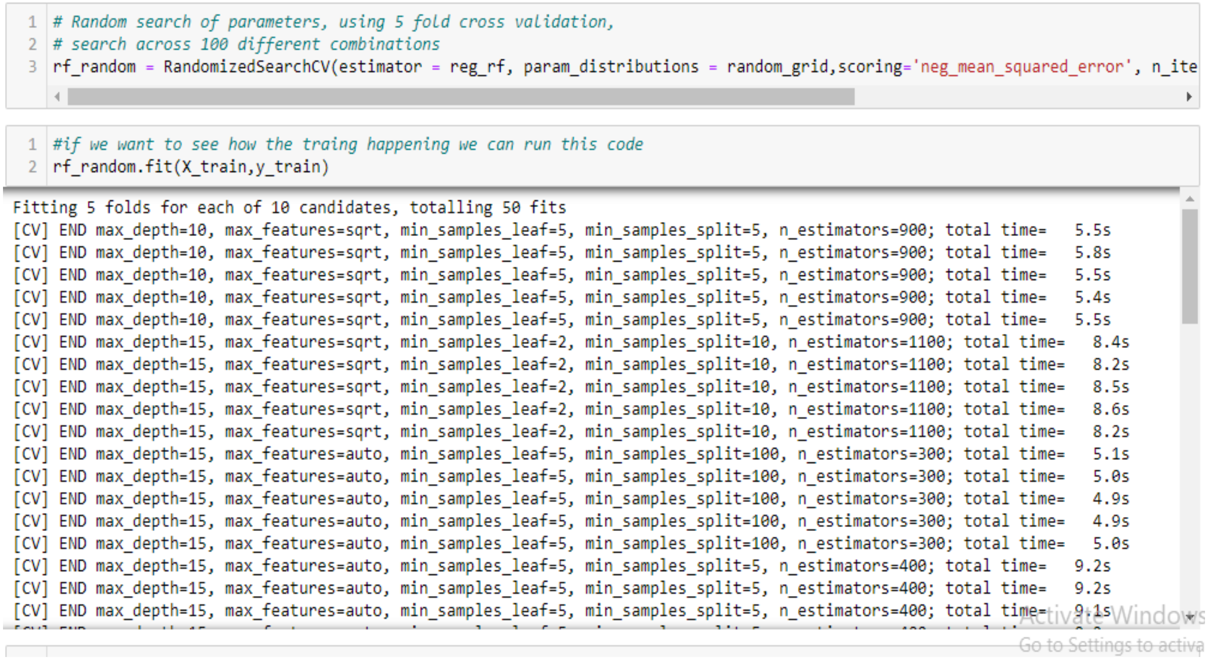
Choose following method for hyperparameter tuning RandomizedSearchCV --> Fast GridSearchCV Assign hyperparameters in form of dictionery Fit the model Check best paramters and best score

```
from sklearn.model_selection import RandomizedSearchCV
```

```
#Randomized Search CV

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

Random search of parameters, using 5 fold cross validation,

search across 100 different combinations:



```
# 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_error', n_ite
```

```
#if we want to see how the traing happening we can run this code
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 time=   5.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.5s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.4s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   5.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.4s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.2s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.5s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.6s
[CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   8.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.1s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   4.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   4.9s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=100, n_estimators=300; total time=   5.0s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   9.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   9.2s
[CV] END max_depth=15, max_features=auto, min_samples_leaf=5, min_samples_split=5, n_estimators=400; total time=   9.1s
```

CONCLUSION

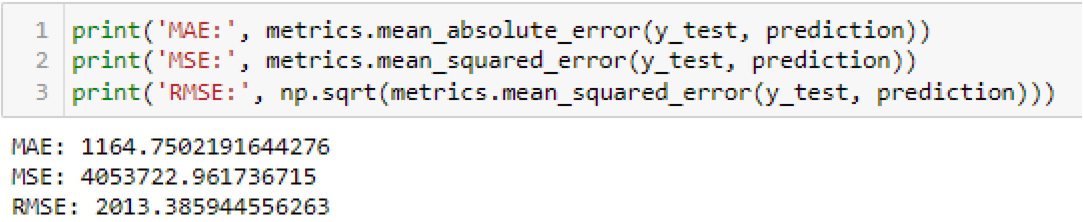
- Key Findings and Conclusions of the Study

- I find out the price of Flight price with the available independent variables
- I find out which variables are important to predict the price of variable by plotting countplot
- I fined which variables are important to predict the price of variable
- I fined how these variables describe the price of the flight

- Learning Outcomes of the Study in respect of Data Science

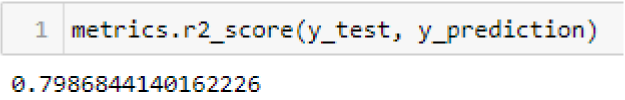
- First I define the question and done EDA(Exploratory data analysis)
- I collected all necessary dataset and drop all unnecessary data.
- Cleaning the data
- Analysing the data
    - I done comparison between Price & total stop Identifies the price depending on total stops or not.
- Sharing your results
    - I predict the price of flight tickets with the available independent variables
    - I find out which variables are important to predict the price of variable by plotting countplot.
    - These variables describe the price of the flight.

- Limitations of this work and Scope for Future Work



```
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: 1164.7502191644276
MSE: 4053722.961736715
RMSE: 2013.385944556263
```



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

```
0.7986844140162226
```

We can observe the **R2 score;** it is a very important metric that is used to evaluate the performance of a regression-based machine learning model. It is pronounced as R squared and is also known as the coefficient of determination. It works by measuring the amount of variance in the predictions explained by the dataset.

We can improve the results by removing more outliers and regularization (making the resulting more regular).