FLIP ROBO

FLIGHT PRICE PREDICTION

Submitted by:

Vijay Ashley Rodrigues K.

ACKNOWLEDGMENT

- I would like to thank FlipRobo Technologies for providing me this opportunity and guidance throughout the project and all the steps that are implemented.
- I have primarily referred to various articles scattered across various websites for the purpose of getting an idea on how prices are influenced for flights.
- I would like to thank the technical support team also for helping me out and reaching out to me on clearing all my doubts as early as possible
- I would like to thank my project SME Keshav Bansal for providing the flexibility in time and also for giving us guidance in creating the project.
- The following are some of the articles I referred to in this project.

- https://www.kayak.co.in/?ispredir=true
- https://en.wikipedia.org/wiki/List_of_low-cost_airlines
- https://www.intermiles.com/blog/top-factors-that-determine-flight-ticket-prices

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

1. Data Collection

- You have to scrape at least 1500 rows of data. You can scrape more data as well, it's up to you, More the data the better the model. In this section you have to scrape the data of flights from different websites (yatra.com, skyscanner.com, official websites of airlines, etc). The number of columns for data doesn't have a limit, it's up to you and your creativity. Generally, these columns are airline name, date of journey, source, destination, route, departure time, arrival time, duration, total stops and the target variable price. You can make changes to it, you can add or you can remove some columns, it completely depends on the website from which you are fetching the data.

2. Data Analysis

- After cleaning the data, you have to do some analysis on the data. Do airfares change frequently? Do they move in small increments or in large jumps? Do they tend to go up or down over time? What is the best time to buy so that the consumer can save the most by taking the least risk? Does the price increase as we get near to the departure date? Is Indigo cheaper than Jet Airways? Are morning flights expensive?

3. Model Building

- After collecting the data, you need to build a machine learning model. Before model building do all data pre-processing steps. Try different models with different hyper parameters and select the best model. Follow the complete life cycle of data science.
  Include all the steps like
  1. Data Cleaning
  2. Exploratory Data Analysis
  3. Data Pre-processing
  4. Model Building
  5. Model Evaluation
  6. Selecting the best model

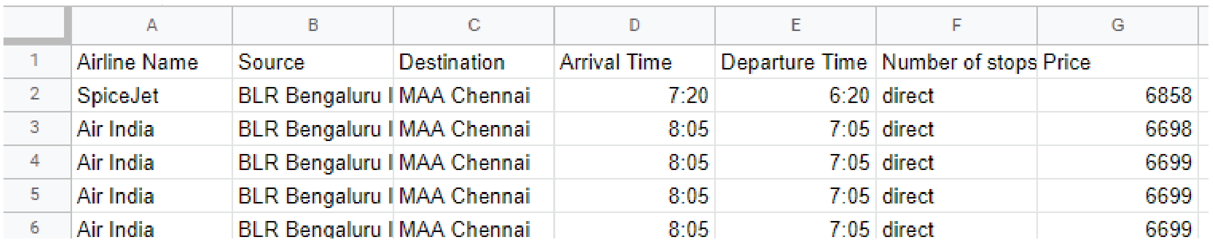
Motivation for the Problem Undertaken

- Do airfares fluctuate a lot? Are they moving in modest steps or in big leaps? Do they have a tendency to rise or fall over time? When is the optimum time to buy in order for the consumer to save the most money while assuming the least amount of risk? Is the price going to go up as we approach our departure date? Is Indigo Airlines less expensive than Jet Airways? Is it pricey to fly in the morning?

Analytical Problem Framing

Data Sources and their formats

- I have web scrapped the available flight details from Kayak.com using Selenium Webdriver.
- The dataset contains the Flight related data with 1981 records (rows) and 7 features (columns).
- The dataframe is then converted to a csv file which is used for further analysis.



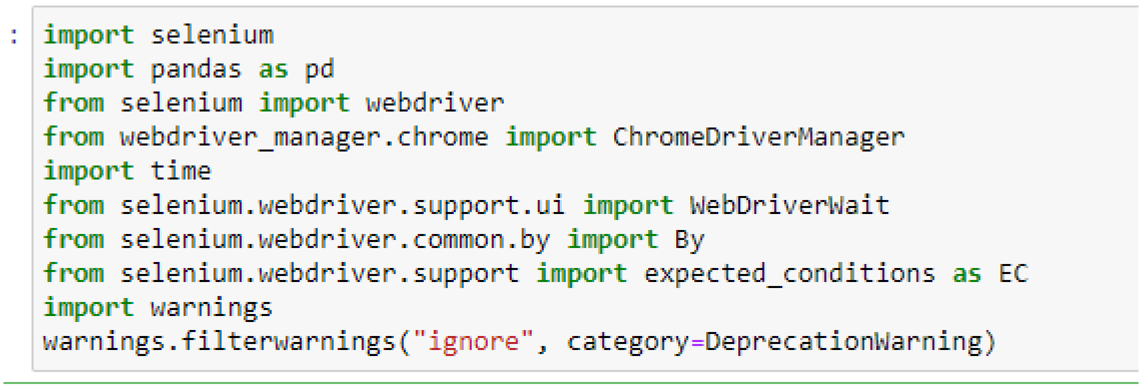
	A	B	C	D	E	F	G
1	Airline Name	Source	Destination	Arrival Time	Departure Time	Number of stops	Price
2	SpiceJet	BLR Bengaluru I	MAA Chennai	7:20	6:20	direct	6858
3	Air India	BLR Bengaluru I	MAA Chennai	8:05	7:05	direct	6698
4	Air India	BLR Bengaluru I	MAA Chennai	8:05	7:05	direct	6699
5	Air India	BLR Bengaluru I	MAA Chennai	8:05	7:05	direct	6699
6	Air India	BLR Bengaluru I	MAA Chennai	8:05	7:05	direct	6699

- Data Preprocessing Done

1. Acquire the data and converting to dataset

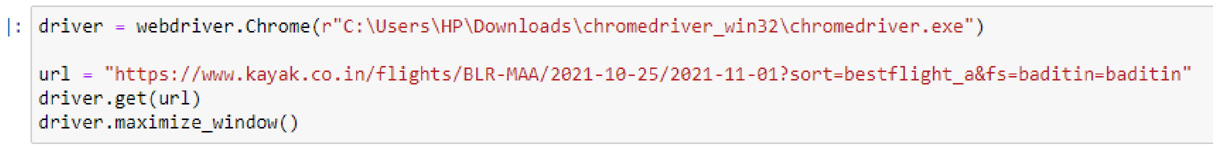
    - I have considered kayak.com travel website to scrape flight details using Selenium Webdriver.

A) Import the libraries:



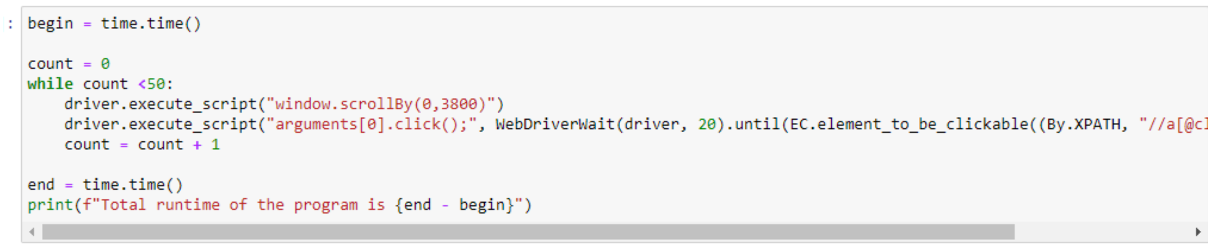
```
import selenium
import pandas as pd
from selenium import webdriver
from webdriver_manager.chrome import ChromeDriverManager
import time
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.common.by import By
from selenium.webdriver.support import expected_conditions as EC
import warnings
warnings.filterwarnings("ignore", category=DeprecationWarning)
```

B) Establish connection to webdriver and fetch URL



```
driver = webdriver.Chrome(r"C:\Users\HP\Downloads\chromedriver_win32\chromedriver.exe")

url = "https://www.kayak.co.in/flights/BLR-MAA/2021-10-25/2021-11-01?sort=bestflight_a&fs=baditin=baditin"
driver.get(url)
driver.maximize_window()
```

C) Scroll the page and select the button several times



```
begin = time.time()

count = 0
while count <50:
    driver.execute_script("window.scrollBy(0,3800)")
    driver.execute_script("arguments[0].click();", WebDriverWait(driver, 20).until(EC.element_to_be_clickable((By.XPATH, "//a[@cl
    count = count + 1

end = time.time()
print(f"Total runtime of the program is {end - begin}")
```

D) Data Extraction

All the necessary data is extracted and stored in respective variables for later use.

```python
begin = time.time()

airline = []
airport_from = []
airport_time = []
airport_dep = []
airline_stops =[]
price = []


air_from = driver.find_elements_by_xpath("(//span[@class='airport-name'])")
for airfrom in air_from:
    airport_from.append(airfrom.text)
    air_from = airport_from[::-4]
    air_to = airport_from[1:len(airport_from):4]

air_time_arr = driver.find_elements_by_xpath("(//span[@class='arrival-time base-time'])")
for airtime in air_time_arr:
    airport_time.append(airtime.text)
    air_arrival = airport_time[0:len(air_time_arr):2]

air_time_dep = driver.find_elements_by_xpath("(//span[@class='depart-time base-time'])")
for airdep in air_time_dep:
    airport_dep.append(airdep.text)
    air_dep = airport_dep[0:len(air_time_dep):2]

air_stop = driver.find_elements_by_xpath("(//div[@class='top'])")
for airstop in air_stop:
    airline_stops.append(airstop.text)
    air_stop_new = airline_stops[1:len(air_stop):3]


end = time.time()
print(f"Total runtime of the program is {end - begin}")
```

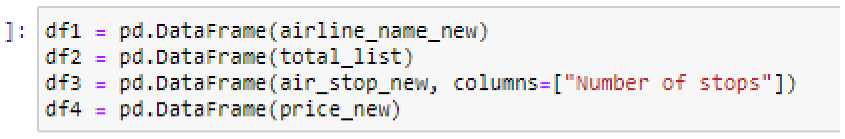
```
Total runtime of the program is 454.8920774459839
```

```python
airline_name = driver.find_elements_by_xpath("(//span[@class='codeshares-airline-names'])")
for air in airline_name:
    airline.append(air.text)

airline_name_new =pd.DataFrame(airline, columns =["Split col airline"])
airline_name_new = airline_name_new["Split col airline"].str.split(",", expand=True)
airline_name_new.drop(columns=[1], inplace=True)
airline_name_new.reset_index(drop=True)
airline_name_new.columns =["Airline Name"]
```

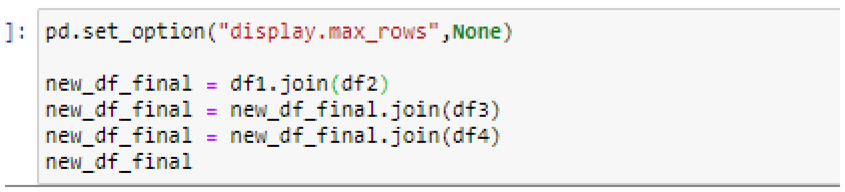
```python
air_price = driver.find_elements_by_xpath("(//div[@class='col-price result-column js-no-dtog'])")
for airprice in air_price:
    price.append(airprice.text.strip().replace("₹ ","").replace(",",""))

price_new =pd.DataFrame(price, columns =["Split price col"])
price_new = price_new["Split price col"].str.split("\n", expand=True)
price_new.drop(columns=[3,2,1], inplace=True)
price_new.reset_index(drop=True)
price_new.columns =["Price"]
```

```python
total_list = {
            "Source":air_from,
            "Destination":air_to,
            "Arrival Time":air_arrival,
            "Departure Time":air_dep}
```

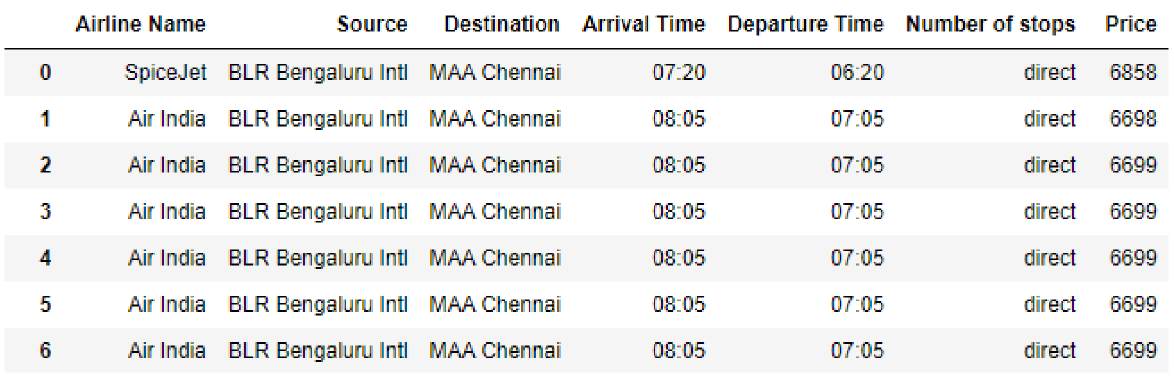
E) Conversion of the extracted data into a DataFrame



```
df1 = pd.DataFrame(airline_name_new)
df2 = pd.DataFrame(total_list)
df3 = pd.DataFrame(air_stop_new, columns=["Number of stops"])
df4 = pd.DataFrame(price_new)
```



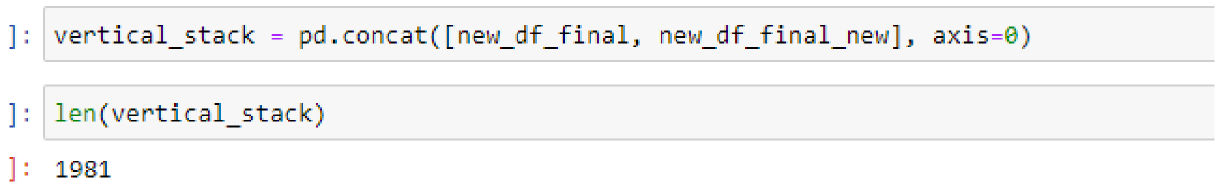
```
pd.set_option("display.max_rows",None)

new_df_final = df1.join(df2)
new_df_final = new_df_final.join(df3)
new_df_final = new_df_final.join(df4)
new_df_final
```



	Airline Name	Source	Destination	Arrival Time	Departure Time	Number of stops	Price
0	SpiceJet	BLR Bengaluru Intl	MAA Chennai	07:20	06:20	direct	6858
1	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6698
2	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6699
3	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6699
4	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6699
5	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6699
6	Air India	BLR Bengaluru Intl	MAA Chennai	08:05	07:05	direct	6699

F) Combination of DataFrames

I have extracted 2 separate data from same website and combined both the dataframes into one by stacked one above another as follows:

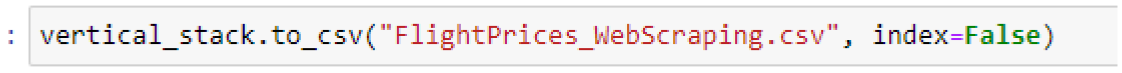


```
vertical_stack = pd.concat([new_df_final, new_df_final_new], axis=0)
```

```
len(vertical_stack)
```

```
1981
```

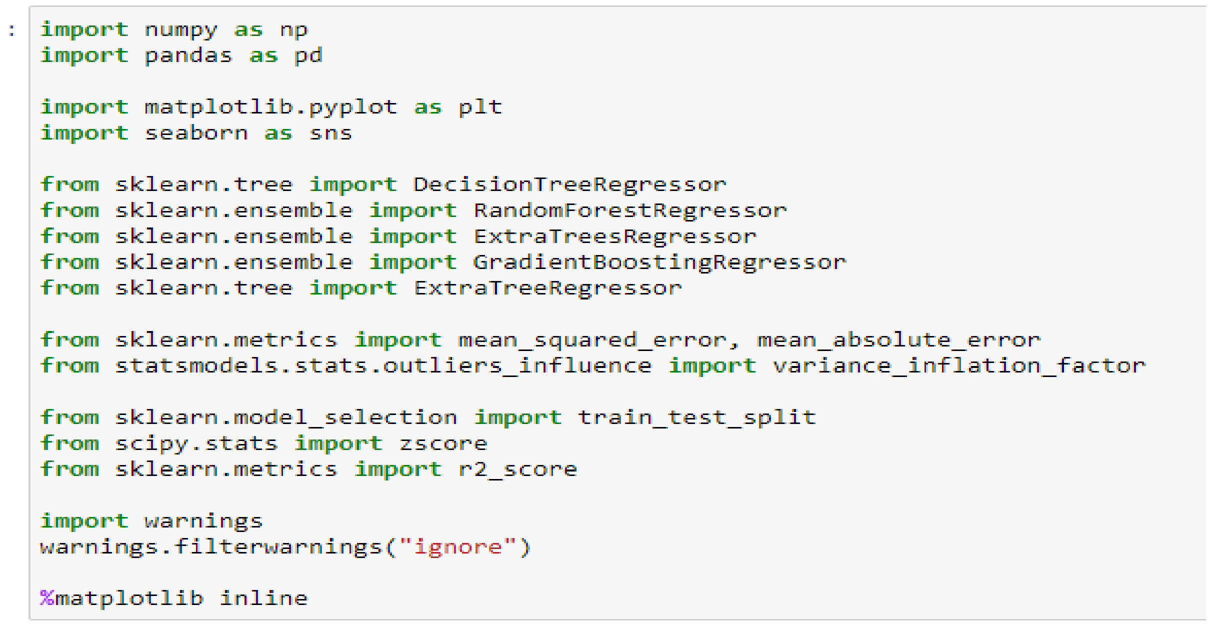
G) Convert the final dataframe into a CSV file (index removed)



```
vertical_stack.to_csv("FlightPrices_WebScraping.csv", index=False)
```

2. Import all the crucial libraries

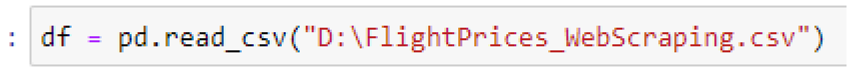
- For this project I have used the following major libraries like Pandas-profiling, Pandas, Matplotlib and Seaborn that are used for EDA or any other pre-processing done in this scenario.



```
:  import numpy as np
   import pandas as pd

   import matplotlib.pyplot as plt
   import seaborn as sns

   from sklearn.tree import DecisionTreeRegressor
   from sklearn.ensemble import RandomForestRegressor
   from sklearn.ensemble import ExtraTreesRegressor
   from sklearn.ensemble import GradientBoostingRegressor
   from sklearn.tree import ExtraTreeRegressor

   from sklearn.metrics import mean_squared_error, mean_absolute_error
   from statsmodels.stats.outliers_influence import variance_inflation_factor

   from sklearn.model_selection import train_test_split
   from scipy.stats import zscore
   from sklearn.metrics import r2_score

   import warnings
   warnings.filterwarnings("ignore")

   %matplotlib inline
```

3. Import the dataset

- The dataset is in now in CSV format and it is imported using Pandas library in Jupyter Notebook.

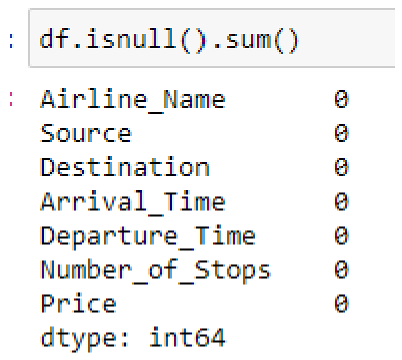


```
:  df = pd.read_csv("D:\FlightPrices_WebScraping.csv")
```

- The statement **pd. set_option("display.max_columns", None)** simply allows us to physically see all the feature columns.
- The statement **pd. set_option("display.max_rows", None)** simply allows us to physically see all the feature rows.
- By default, Jupyter Notebook doesn’t display all the rows and columns at the same time and only selected portion from starting and ending of dataset are displayed.

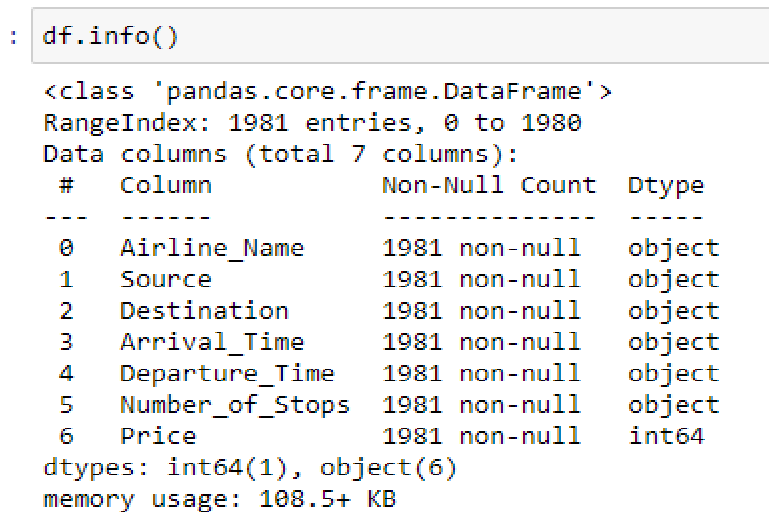
4. Identifying and handling the missing values

- There are no missing values hence no actions have been taken



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

```
Airline_Name         0
Source               0
Destination          0
Arrival_Time         0
Departure_Time       0
Number_of_Stops      0
Price                0
dtype: int64
```

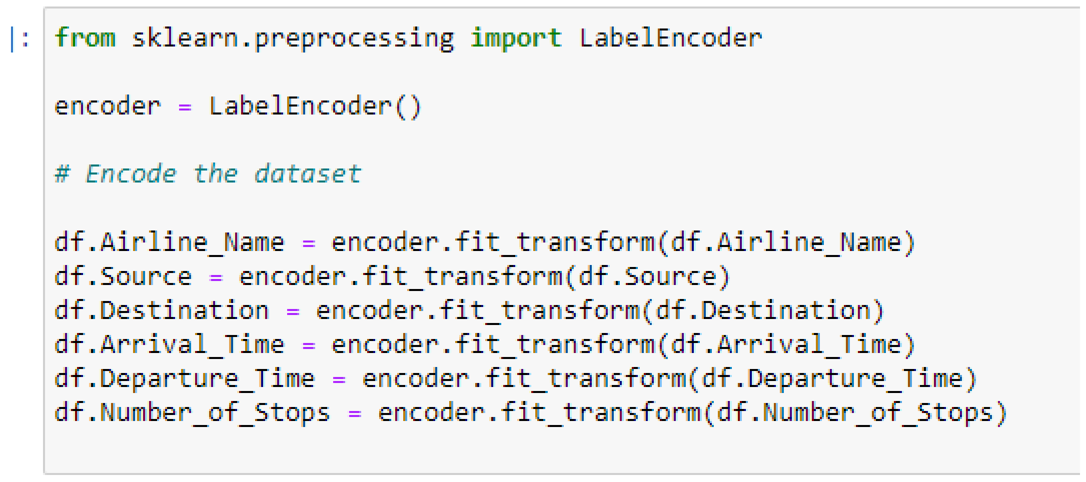


```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1981 entries, 0 to 1980
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   Airline_Name     1981 non-null   object
 1   Source           1981 non-null   object
 2   Destination      1981 non-null   object
 3   Arrival_Time     1981 non-null   object
 4   Departure_Time   1981 non-null   object
 5   Number_of_Stops  1981 non-null   object
 6   Price            1981 non-null   int64 
dtypes: int64(1), object(6)
memory usage: 108.5+ KB
```

5. Encoding the categorical data

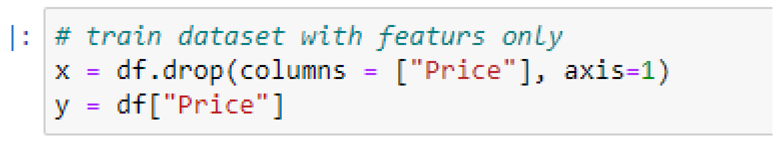
- I have used LabelEncoder as the data is categorical and is not ordinal in nature.



```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()

# Encode the dataset

df.Airline_Name = encoder.fit_transform(df.Airline_Name)
df.Source = encoder.fit_transform(df.Source)
df.Destination = encoder.fit_transform(df.Destination)
df.Arrival_Time = encoder.fit_transform(df.Arrival_Time)
df.Departure_Time = encoder.fit_transform(df.Departure_Time)
df.Number_of_Stops = encoder.fit_transform(df.Number_of_Stops)
```

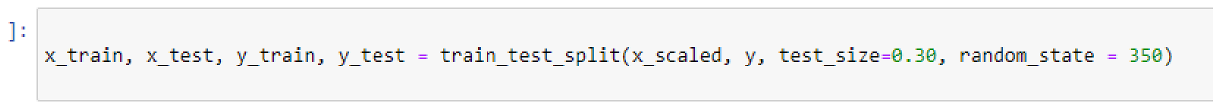
6. Splitting the dataset

- Splitting up of dataset between x (features) and y (target column)



```
# train dataset with featurs only
x = df.drop(columns = ["Price"], axis=1)
y = df["Price"]
```

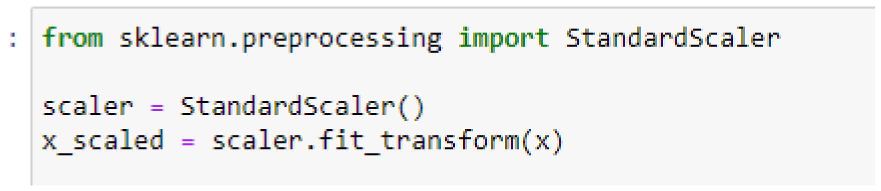
- Split the dataset into a train and test data set.
- I have chosen a random state of 200 and 30% of data is divided as a test dataset.



```
]:
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.30, random_state = 350)
```

7. Feature scaling

- Finding variance inflation factor in each scaled column
- This gives us a relationship between feature vs feature and we can drop it if necessary to avoid multicollinearity.
- From the below observation, I have not considered this step.
- Let's now Scale the data for further processing.
- we have used StandardScaler for further scaling up of data.



```
: from sklearn.preprocessing import StandardScaler

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

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

  - This dataset contains only 1981 records and has only 7 features. It depends entirely on the websites referred to.
  - Since this dataset is not possible to extract for multiple dates through web scraping, I have considered only 25th Oct 2021 as the date.

- Hardware and Software Requirements and Tools Used

Hardware / Software specifications

Windows 10 64bit

Anaconda 2021.05

Python version – Python 3.9.5

Jupyter Notebook 6.4 and Google Colab

Sweetviz – Another automated EDA package. I have used to get complete clarity of features as this has only few features.

Pandas – This is used in the data manipulation, processing and cleaning and also to get description, stats and almost everywhere in the project.

Matplotlib and **Seaborn** Majority of the data visualizations are plotted using Seaborn and to some extent only Matplotlib is used here.

LabelEncoder I have used this **Skipy** library to convert all the non-ordered categorical data into numerical data. In our case it's only one feature "pcircle".

train_test_split module from **sklearn.model_selection** to split the data into train and test and then used **StandardScaler** to bring the values to one level before imputing to model.

Warnings: I have used "ignore" to avoid the general errs that may occur and used "FutureWarning" to avoid errors that I got when running algorithms on Google Colab. To have a generic and efficient notebook file I used this as well.

Sciypy Used xgboost to import XGBClassifier and remaining algorithms including ensemble are part of Scipy.

Lime library is used to create a basic dashboard to explain the output of the algorithm.

Model/s Development and Evaluation

- Testing of Identified Approaches (Algorithms)

For the model building I have considered the following 5 algorithms for the training and testing.

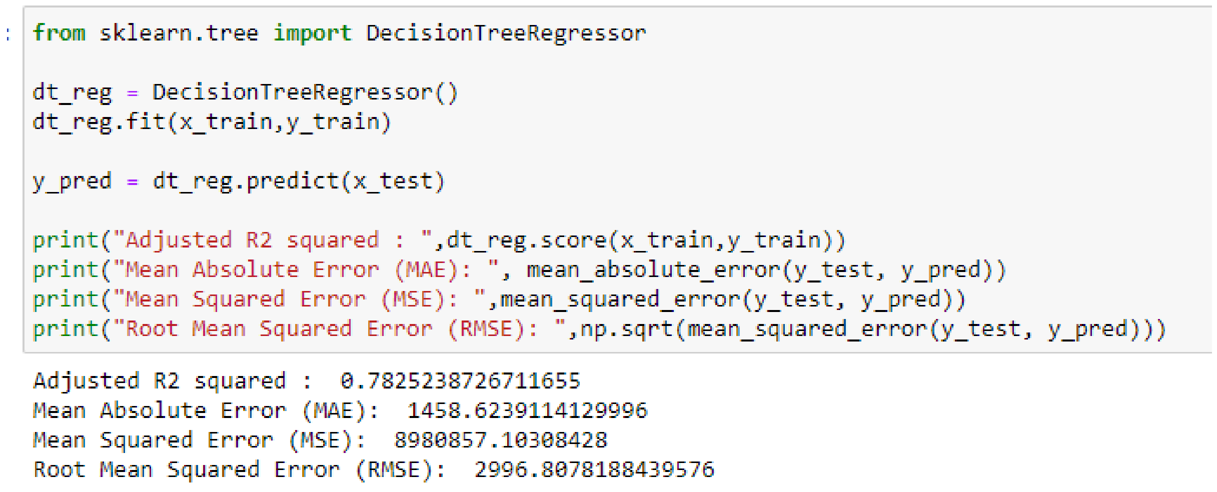
DecisionTreeRegressor
RandomForestRegressor
ExtraTreesRegressor
GradientBoostingRegressor
ExtraTreeRegressor

- Run and Evaluate selected models

  - I have used a total of 5 machine learning algorithms to find the best and suited model which also includes ensemble algorithms.
  - I have considered Adjusted R2 score for model evaluation.
  - Along with this I have also calculated Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as evaluation techniques.

1) DecisionTreeRegressor

- From the below algorithm we can see the accuracy score for DecisionTreeRegressor is approximately **78.25%**.

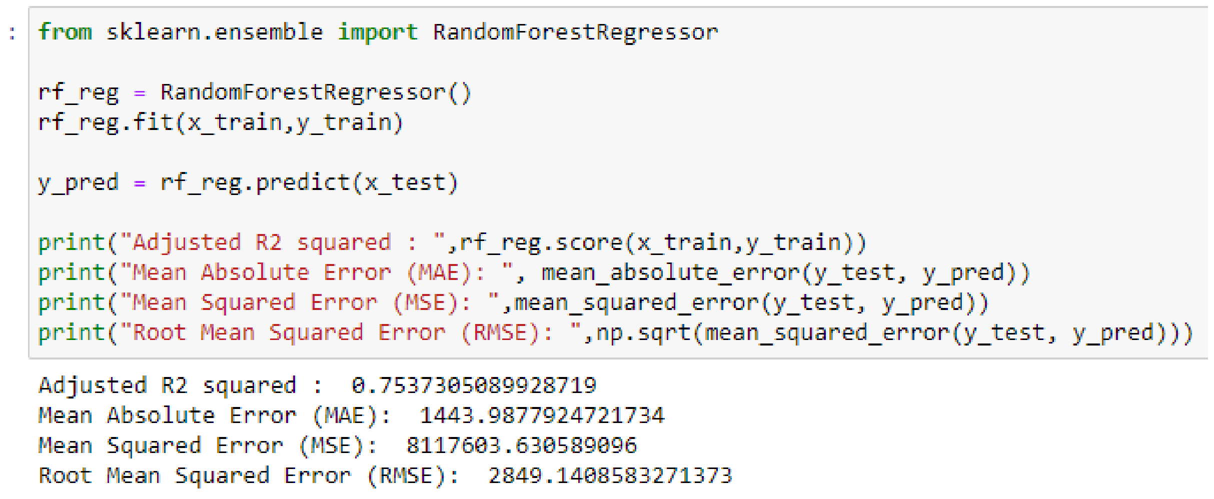


```
from sklearn.tree import DecisionTreeRegressor

dt_reg = DecisionTreeRegressor()
dt_reg.fit(x_train,y_train)

y_pred = dt_reg.predict(x_test)

print("Adjusted R2 squared : ",dt_reg.score(x_train,y_train))
print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.7825238726711655
Mean Absolute Error (MAE):  1458.6239114129996
Mean Squared Error (MSE):  8980857.10308428
Root Mean Squared Error (RMSE):  2996.8078188439576
```

2) RandomForestRegressor

- From the below algorithm we can see the accuracy score for RandomForestRegressor is approximately **75.37 %.**

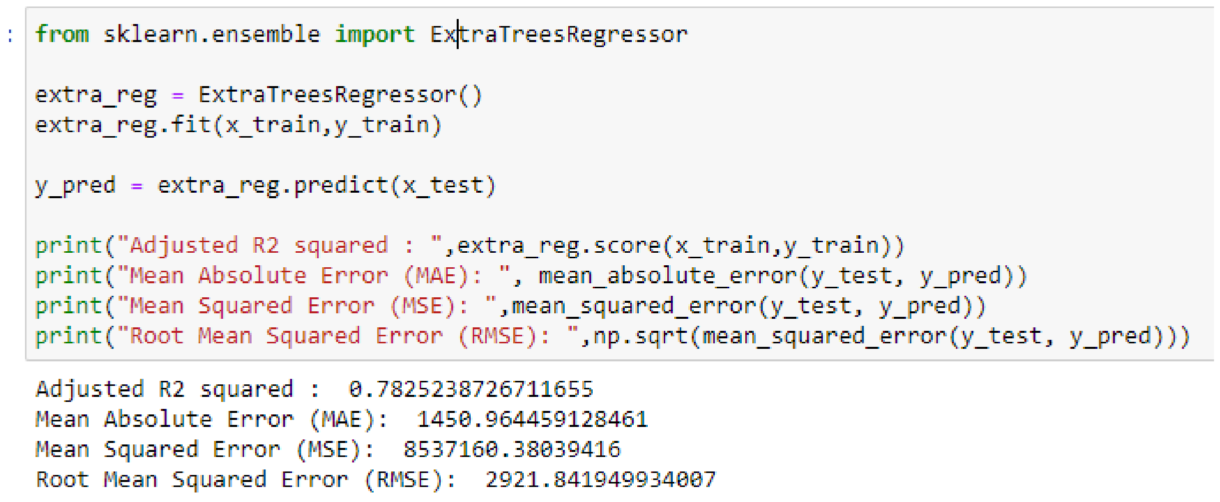


```
from sklearn.ensemble import RandomForestRegressor

rf_reg = RandomForestRegressor()
rf_reg.fit(x_train,y_train)

y_pred = rf_reg.predict(x_test)

print("Adjusted R2 squared : ",rf_reg.score(x_train,y_train))
print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.7537305089928719
Mean Absolute Error (MAE):  1443.9877924721734
Mean Squared Error (MSE):  8117603.630589096
Root Mean Squared Error (RMSE):  2849.1408583271373
```

3) ExtraTreesRegressor

- From the below algorithm we can see the accuracy score for ExtraTreesRegressor is approximately **78.25 %.**

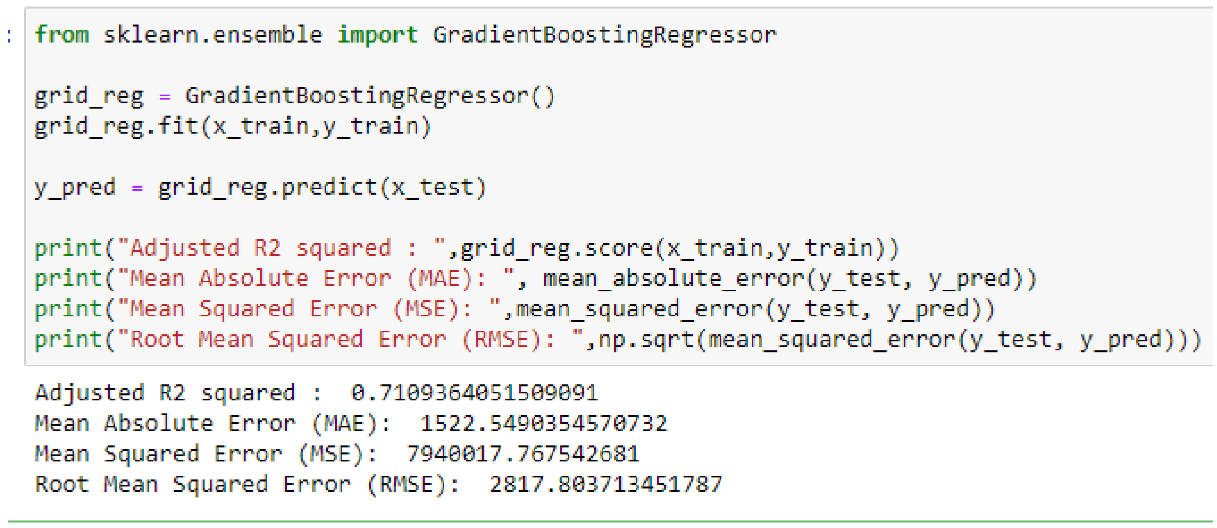


```
from sklearn.ensemble import ExtraTreesRegressor

extra_reg = ExtraTreesRegressor()
extra_reg.fit(x_train,y_train)

y_pred = extra_reg.predict(x_test)

print("Adjusted R2 squared : ",extra_reg.score(x_train,y_train))
print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.7825238726711655
Mean Absolute Error (MAE):  1450.964459128461
Mean Squared Error (MSE):  8537160.38039416
Root Mean Squared Error (RMSE):  2921.841949934007
```

4) GradientBoostingRegressor

- From the below algorithm we can see the accuracy score for GradientBoostingRegressor is approximately **71.09 %**.

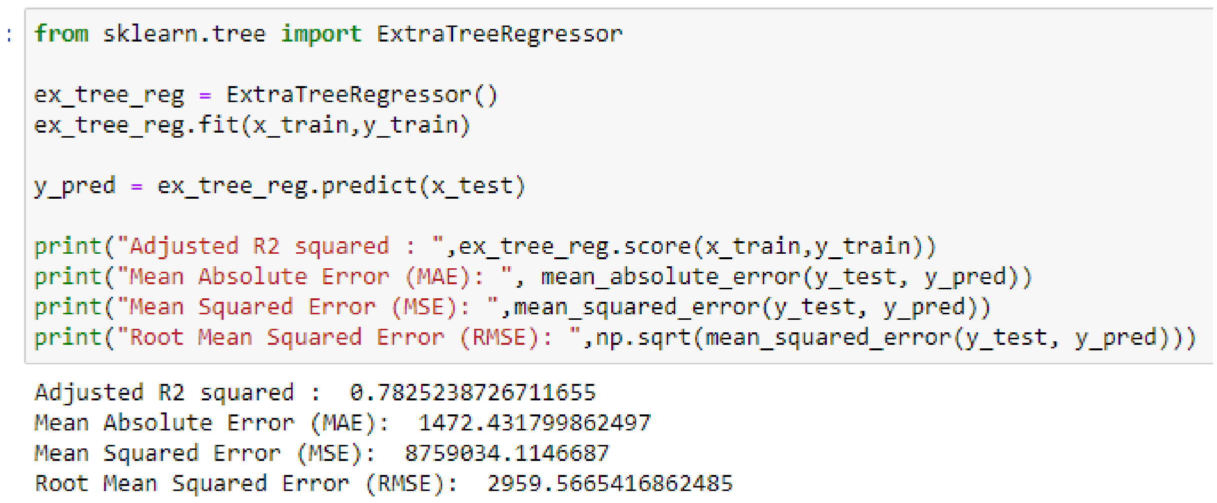


```
: from sklearn.ensemble import GradientBoostingRegressor

  grid_reg = GradientBoostingRegressor()
  grid_reg.fit(x_train,y_train)

  y_pred = grid_reg.predict(x_test)

  print("Adjusted R2 squared : ",grid_reg.score(x_train,y_train))
  print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
  print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
  print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.7109364051509091
Mean Absolute Error (MAE):  1522.5490354570732
Mean Squared Error (MSE):  7940017.767542681
Root Mean Squared Error (RMSE):  2817.803713451787
```

5) ExtraTreeRegressor (not ensemble)

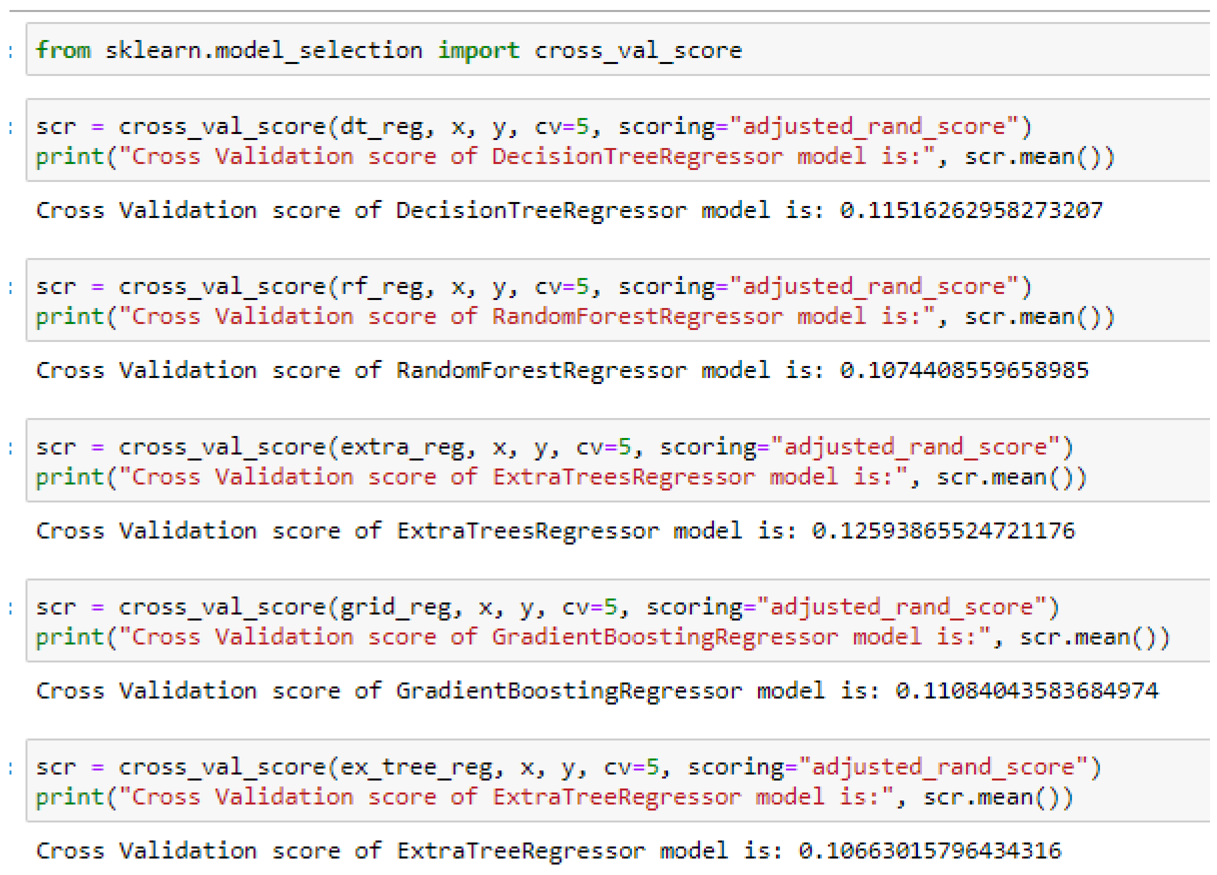
- From the below algorithm we can see the accuracy score for ExtraTreeRegressor (not ensemble) is approximately **78.25 %**.



```
: from sklearn.tree import ExtraTreeRegressor

  ex_tree_reg = ExtraTreeRegressor()
  ex_tree_reg.fit(x_train,y_train)

  y_pred = ex_tree_reg.predict(x_test)

  print("Adjusted R2 squared : ",ex_tree_reg.score(x_train,y_train))
  print("Mean Absolute Error (MAE): ", mean_absolute_error(y_test, y_pred))
  print("Mean Squared Error (MSE): ",mean_squared_error(y_test, y_pred))
  print("Root Mean Squared Error (RMSE): ",np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
Adjusted R2 squared :  0.7825238726711655
Mean Absolute Error (MAE):  1472.431799862497
Mean Squared Error (MSE):  8759034.1146687
Root Mean Squared Error (RMSE):  2959.5665416862485
```

- The below code shows us the cross validation performed over all the algorithms and I have used the CV values as 5.
- If you observe the below screenshot, we get the difference in the values.
- Also, I have considered “adjusted_rand_score” as the scoring method in this case for all.



```
: from sklearn.model_selection import cross_val_score
```

```
: scr = cross_val_score(dt_reg, x, y, cv=5, scoring="adjusted_rand_score")
  print("Cross Validation score of DecisionTreeRegressor model is:", scr.mean())
```

```
Cross Validation score of DecisionTreeRegressor model is: 0.11516262958273207
```

```
: scr = cross_val_score(rf_reg, x, y, cv=5, scoring="adjusted_rand_score")
  print("Cross Validation score of RandomForestRegressor model is:", scr.mean())
```

```
Cross Validation score of RandomForestRegressor model is: 0.1074408559658985
```

```
: scr = cross_val_score(extra_reg, x, y, cv=5, scoring="adjusted_rand_score")
  print("Cross Validation score of ExtraTreesRegressor model is:", scr.mean())
```

```
Cross Validation score of ExtraTreesRegressor model is: 0.12593865524721176
```

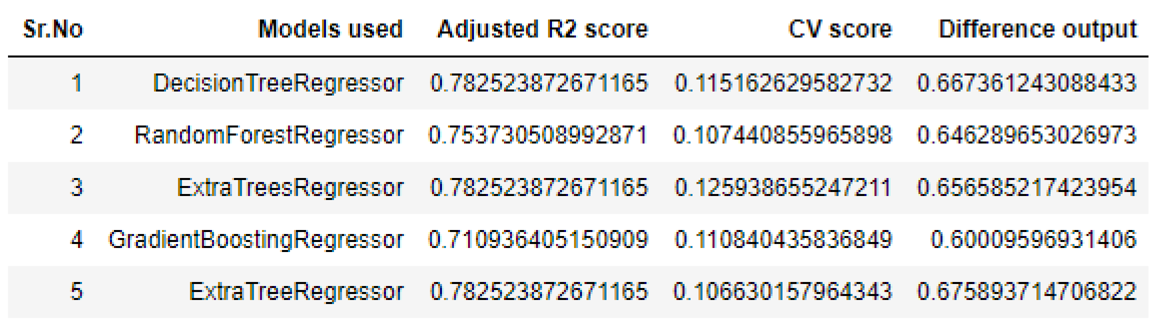
```
: scr = cross_val_score(grid_reg, x, y, cv=5, scoring="adjusted_rand_score")
  print("Cross Validation score of GradientBoostingRegressor model is:", scr.mean())
```

```
Cross Validation score of GradientBoostingRegressor model is: 0.11084043583684974
```

```
: scr = cross_val_score(ex_tree_reg, x, y, cv=5, scoring="adjusted_rand_score")
  print("Cross Validation score of ExtraTreeRegressor model is:", scr.mean())
```

```
Cross Validation score of ExtraTreeRegressor model is: 0.10663015796434316
```

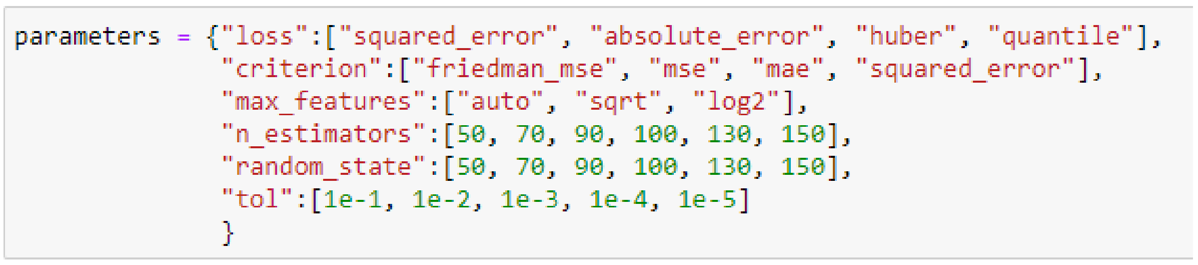
- From the above algorithms GradientBoostingRegressor seems to be an ideal algorithm in this scenario and for this type of dataset.
- The difference between the accuracy score and cross validation for this model is very less compared to other models.



Sr.No	Models used	Adjusted R2 score	CV score	Difference output
1	DecisionTreeRegressor	0.782523872671165	0.115162629582732	0.667361243088433
2	RandomForestRegressor	0.753730508992871	0.107440855965898	0.646289653026973
3	ExtraTreesRegressor	0.782523872671165	0.125938655247211	0.656585217423954
4	GradientBoostingRegressor	0.710936405150909	0.110840435836849	0.60009596931406
5	ExtraTreeRegressor	0.782523872671165	0.106630157964343	0.675893714706822

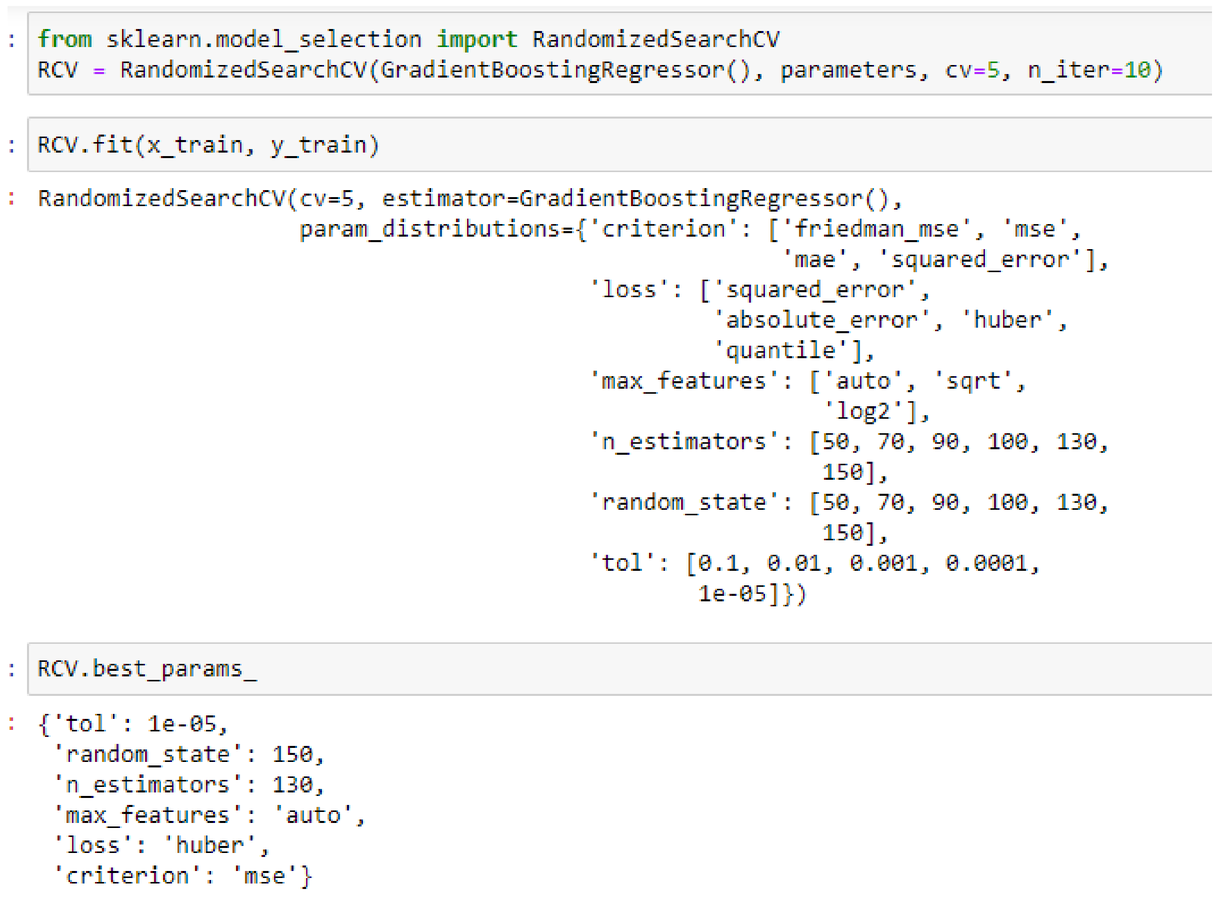
- Let us try to tune the proposed model (GradientBoostingRegressor) to get better accuracy, if possible.

- The "parameters" have been selected from the skicit library and I have considered 6 parameters.



```
parameters = {"loss":["squared_error", "absolute_error", "huber", "quantile"],
              "criterion":["friedman_mse", "mse", "mae", "squared_error"],
              "max_features":["auto", "sqrt", "log2"],
              "n_estimators":[50, 70, 90, 100, 130, 150],
              "random_state":[50, 70, 90, 100, 130, 150],
              "tol":[1e-1, 1e-2, 1e-3, 1e-4, 1e-5]
              }
```

- RandomizedSearchCV is used to tune the parameters by fitting the same to the training dataset and used the best parameters after selection



```
: from sklearn.model_selection import RandomizedSearchCV
  RCV = RandomizedSearchCV(GradientBoostingRegressor(), parameters, cv=5, n_iter=10)

: RCV.fit(x_train, y_train)

: RandomizedSearchCV(cv=5, estimator=GradientBoostingRegressor(),
                     param_distributions={'criterion': ['friedman_mse', 'mse',
                                                        'mae', 'squared_error'],
                                          'loss': ['squared_error',
                                                   'absolute_error', 'huber',
                                                   'quantile'],
                                          'max_features': ['auto', 'sqrt',
                                                           'log2'],
                                          'n_estimators': [50, 70, 90, 100, 130,
                                                           150],
                                          'random_state': [50, 70, 90, 100, 130,
                                                           150],
                                          'tol': [0.1, 0.01, 0.001, 0.0001,
                                                  1e-05]})

: RCV.best_params_

: {'tol': 1e-05,
   'random_state': 150,
   'n_estimators': 130,
   'max_features': 'auto',
   'loss': 'huber',
   'criterion': 'mse'}
```

- Rebuild the model using the appropriate params we received from best_params_

```
4]: mod_grid_reg = GradientBoostingRegressor(tol= 1e-05, random_state= 150, n_estimators= 130, max_features= "auto",
                                             loss= "huber", criterion= "mse")

    mod_grid_reg.fit(x_train,y_train)
    pred = mod_grid_reg.predict(x_test)
    pred
```

```
4]: array([15880.9215685 , 32181.30966232, 14489.86738242, 20723.66722366,
           14680.65212527, 15085.59646836, 13869.87169085, 14805.54978653,
           16622.39425103, 15880.9215685 , 22102.05724731, 15033.77886747,
           20821.85297422, 14501.18159301, 19256.94039052, 13257.2058375 ,
           15880.9215685 , 14702.89220635, 14843.31836924, 15209.59147518,
```

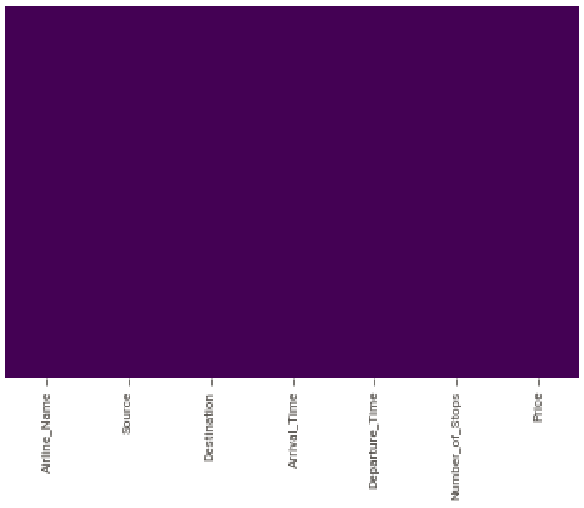
- Key Metrics for success in solving problem under consideration

  - Using sklearn.metrics I have used Mean Absolute Error (MAE), Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) as evaluation techniques and Adjusted R2 score.

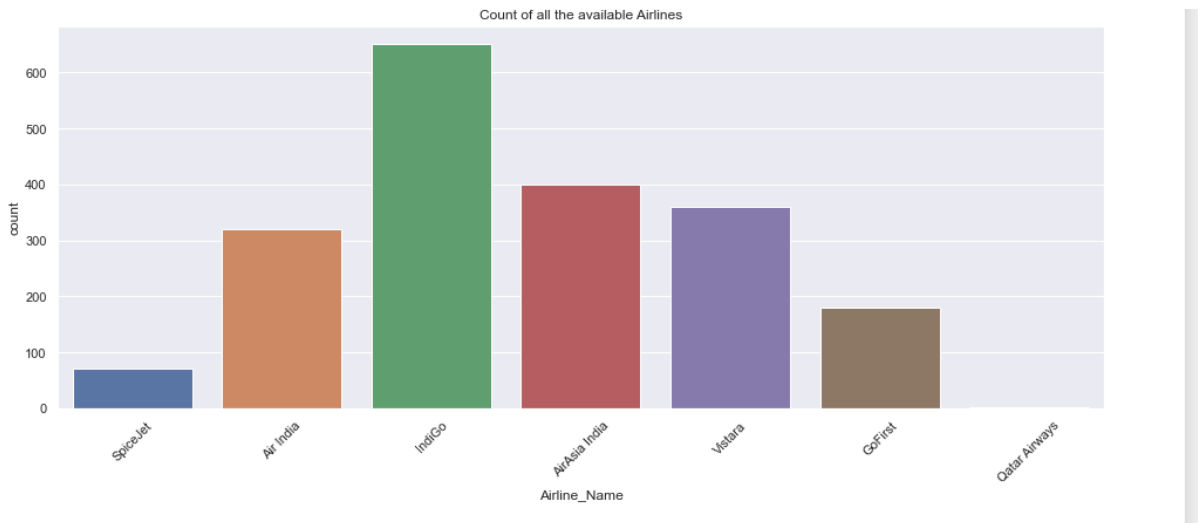
- Visualizations

  - This heat map indicates if there are any missing values. Since all the values are accounted for, there is no change in color of the graph.

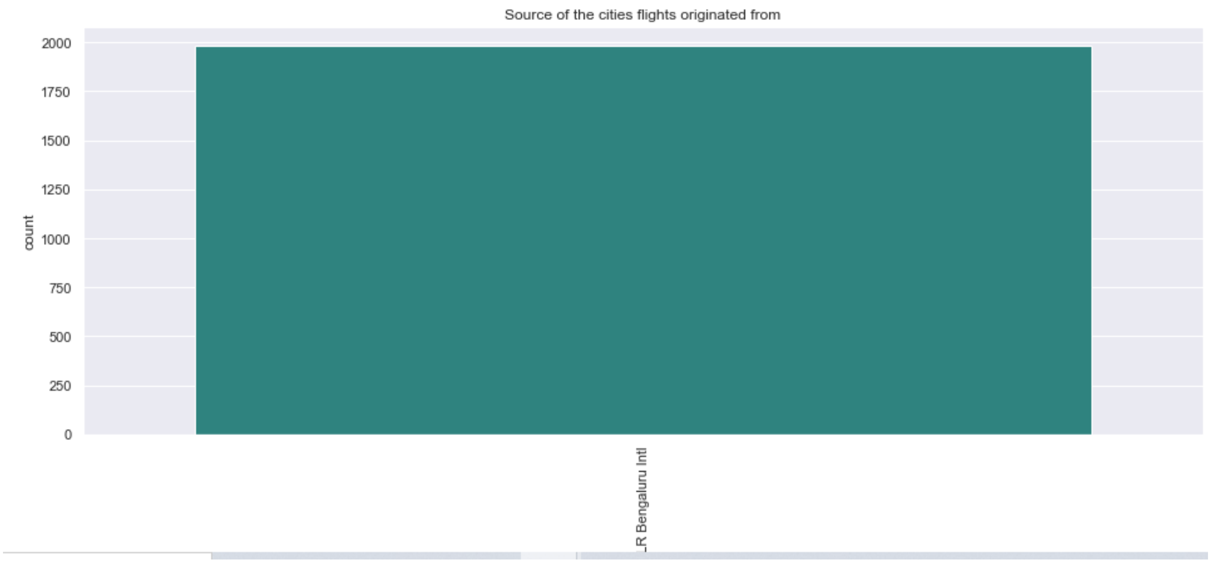
```
: sns.heatmap(df.isnull(), yticklabels=False, cbar=False, cmap="viridis")
  plt.show()
```



- From the below plot with respect to training dataset, we can see that IndiGo holds a large share of business in this sector with about 32.86 % followed by AirAsia India and Vistara.
- However this data is valid only for the date 25th Oct 2021.

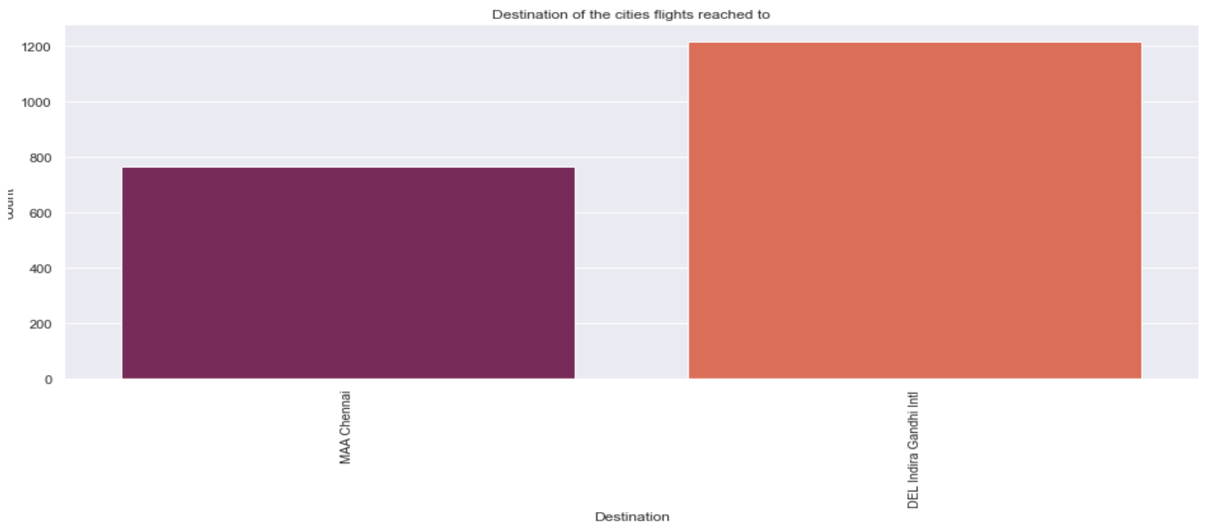


- Since this data was scraped from a website Kayak.com, the source is the same i.e. Bangalore and hence we have no other places.

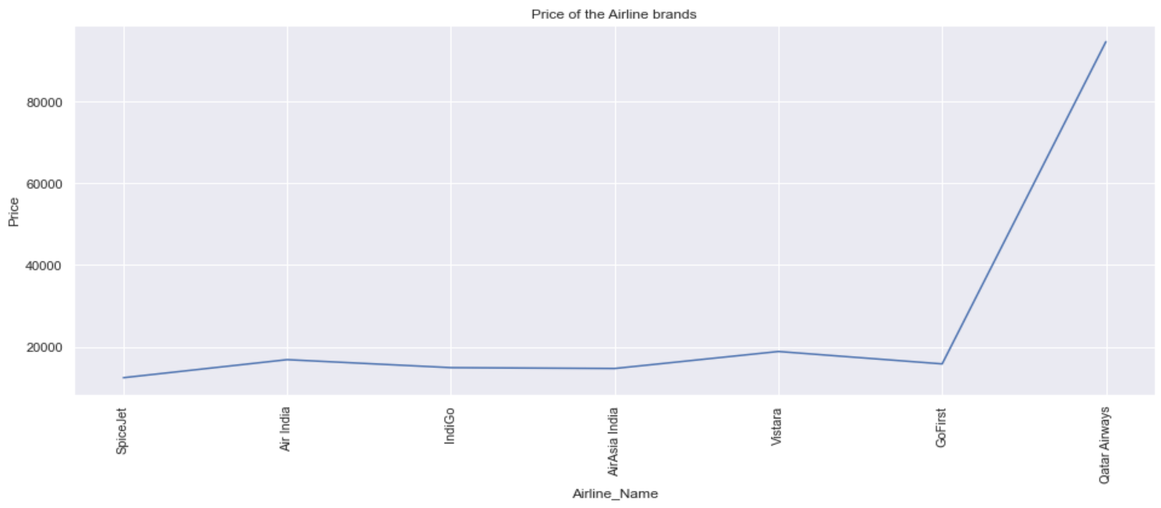


- We can see from the below plot that people live in Delhi compared to Chennai.

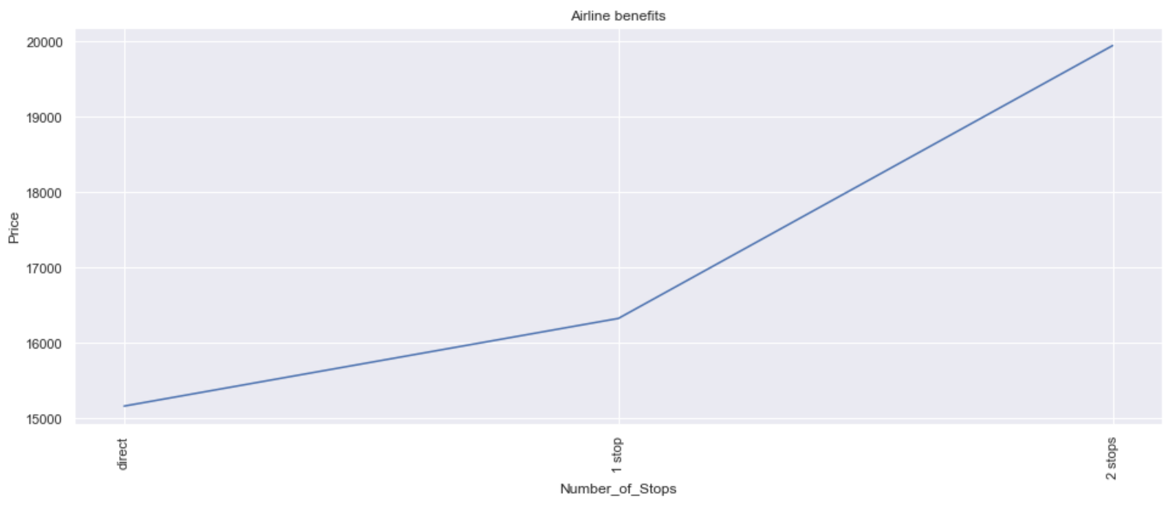
- This data cannot be generalized because these were the only destinations that appeared in the search bar.
- Also, Chennai being one of the busiest metropolitan cities, we can see why a lot of passengers seem to travel there.



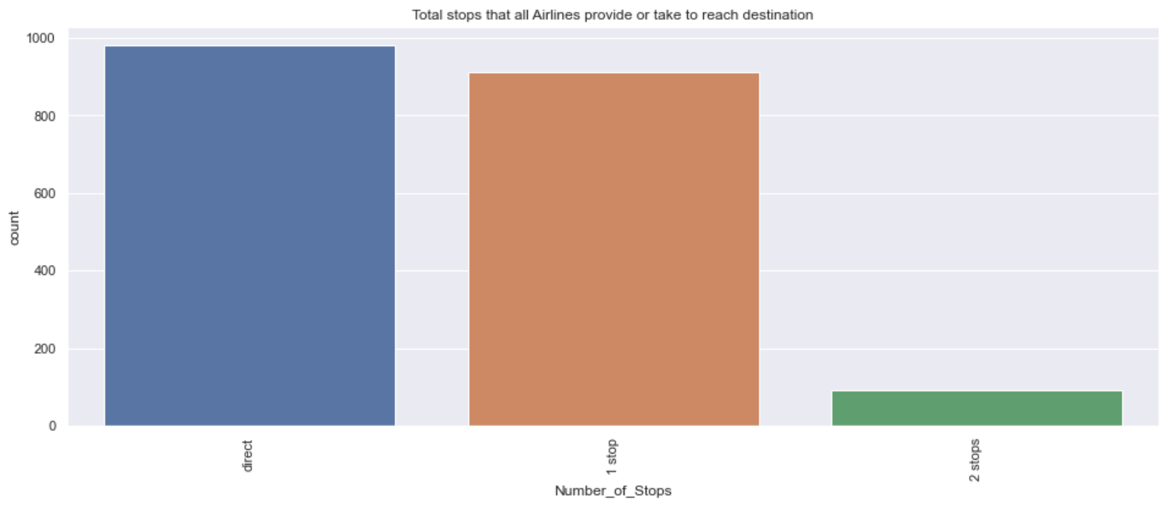
- We can see from the below plot the airline prices for all the companies. It appears Qatar Airways has the highest price for the ticket compared to other airlines.
- It could be because all the remaining flights are domestic and Qatar Airways is the only international flight operating amongst these. The tariff changes, etc may seem to affect. Moreover prices depend on the brand and it's purchase power and Qatar Airways have a good stake in airline industry.
- This could be the reason for the massive price hike for domestic travel.
- However, in general the price of the ticket should not cost more than 15,000 to 20,000 Rs. and Qatar Airways tickets are going above 80,000.



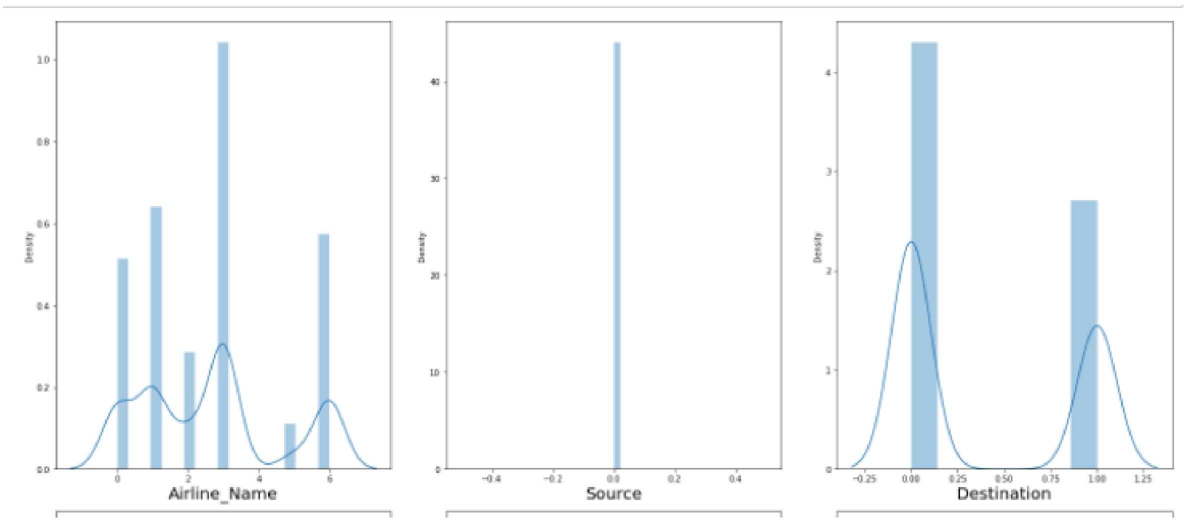
- From the below plot we can see how the total stops influence prices.
- In general sense, flights with single or no stops are cheaper than flights having multiple stops. But in this case we can see that flights with 2 stops have a higher price compared to the other flights.
- Having multiple stops means the airline is liable for hangar rent, and other taxes and unless they plan to travel far, having additional stops will add more cost to the airline companies.



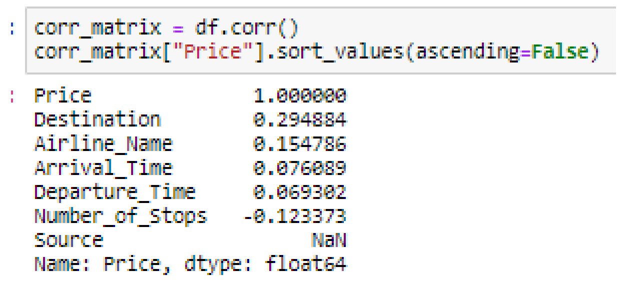
- The below plot shows us the number of stops given by flights
- We can see the majority of flights have "one stop" and this shows that flights have reached directly to the destination with more time for halt.
- This also tells us that for domestic travel passengers do not prefer connecting flights as it's time consuming.
- However we can also see that people have opted for "non-stop" flights. These are the same as "one-stop" but here there is essentially no stop at all. Flight may take time to refuel and that's it.
- This is common for all economy flights or airlines that have cheap tickets because stopping at an airport means the airline company has to pay parking fees that could be huge even if it means staying only for sometime.



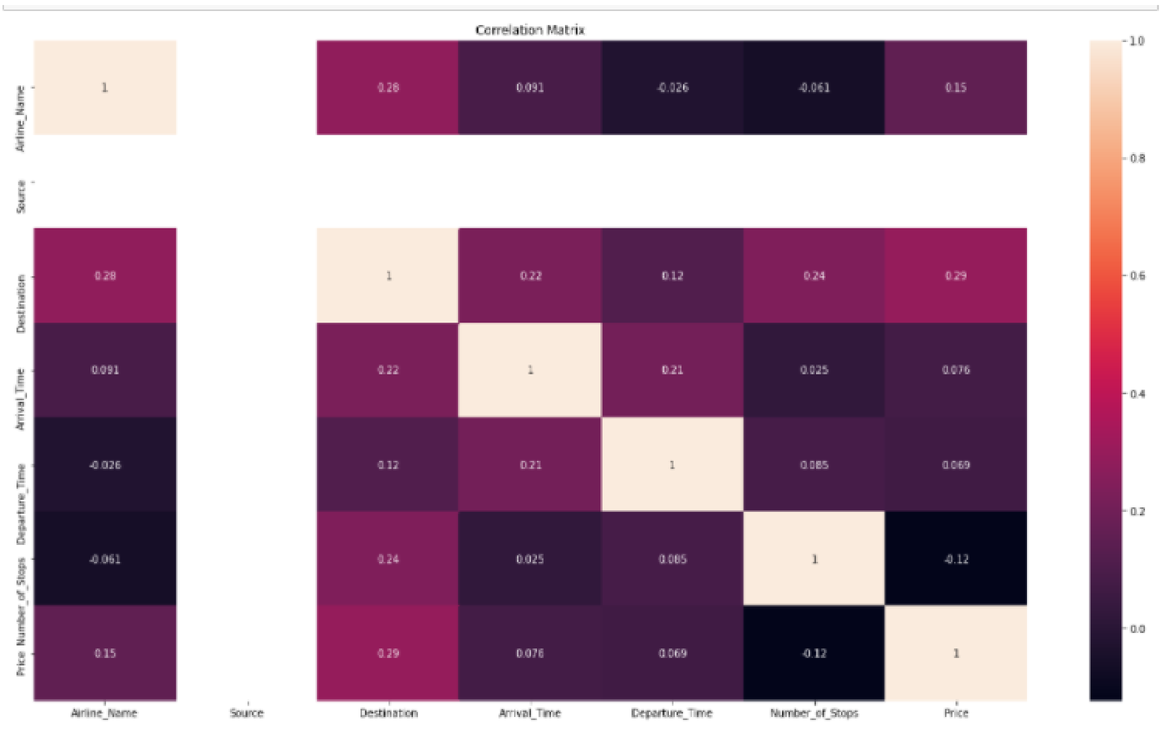
- The below plot shows us the distribution of the dataset. Since all are categorical variables and the only numerical variable "Price" is a target variable, we are not making any changes.



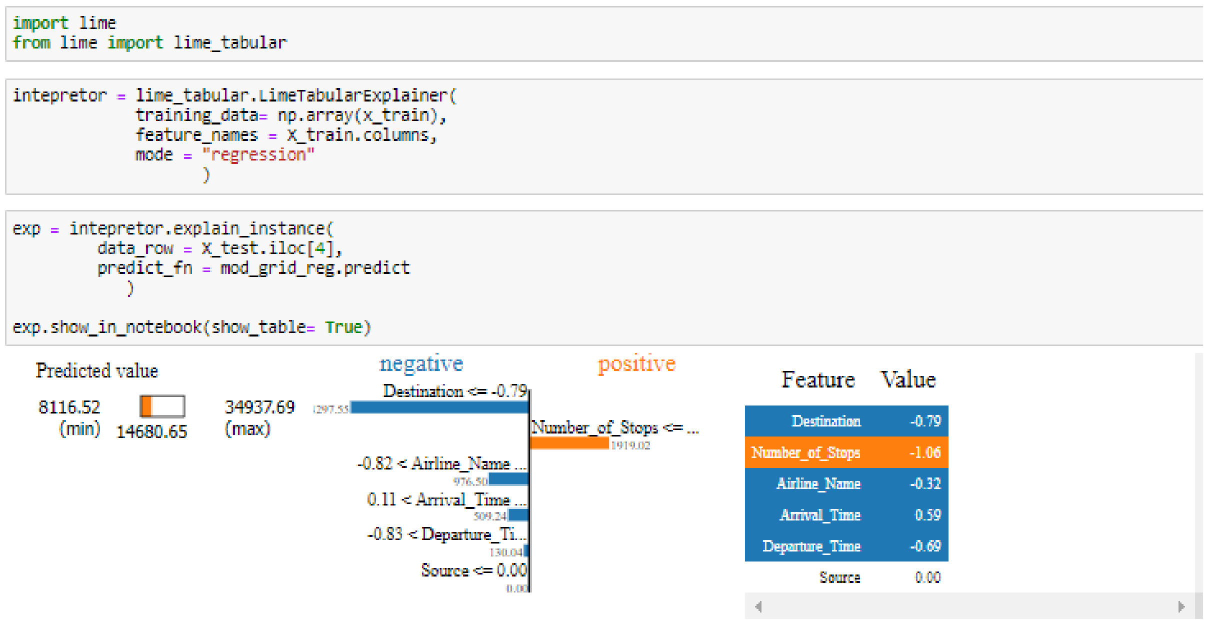
- Let us now examine correlation using a "heatmap" for further clarification.



```
: corr_matrix = df.corr()
  corr_matrix["Price"].sort_values(ascending=False)

: Price              1.000000
  Destination        0.294884
  Airline_Name       0.154786
  Arrival_Time       0.076089
  Departure_Time     0.069302
  Number_of_Stops   -0.123373
  Source                  NaN
  Name: Price, dtype: float64
```



- Model dashboard using Lime library



```
import lime
from lime import lime_tabular
```

```
intepretor = lime_tabular.LimeTabularExplainer(
                training_data= np.array(x_train),
                feature_names = X_train.columns,
                mode = "regression"
                     )
```

```
exp = intepretor.explain_instance(
            data_row = X_test.iloc[4],
            predict_fn = mod_grid_reg.predict
                )

exp.show_in_notebook(show_table= True)
```

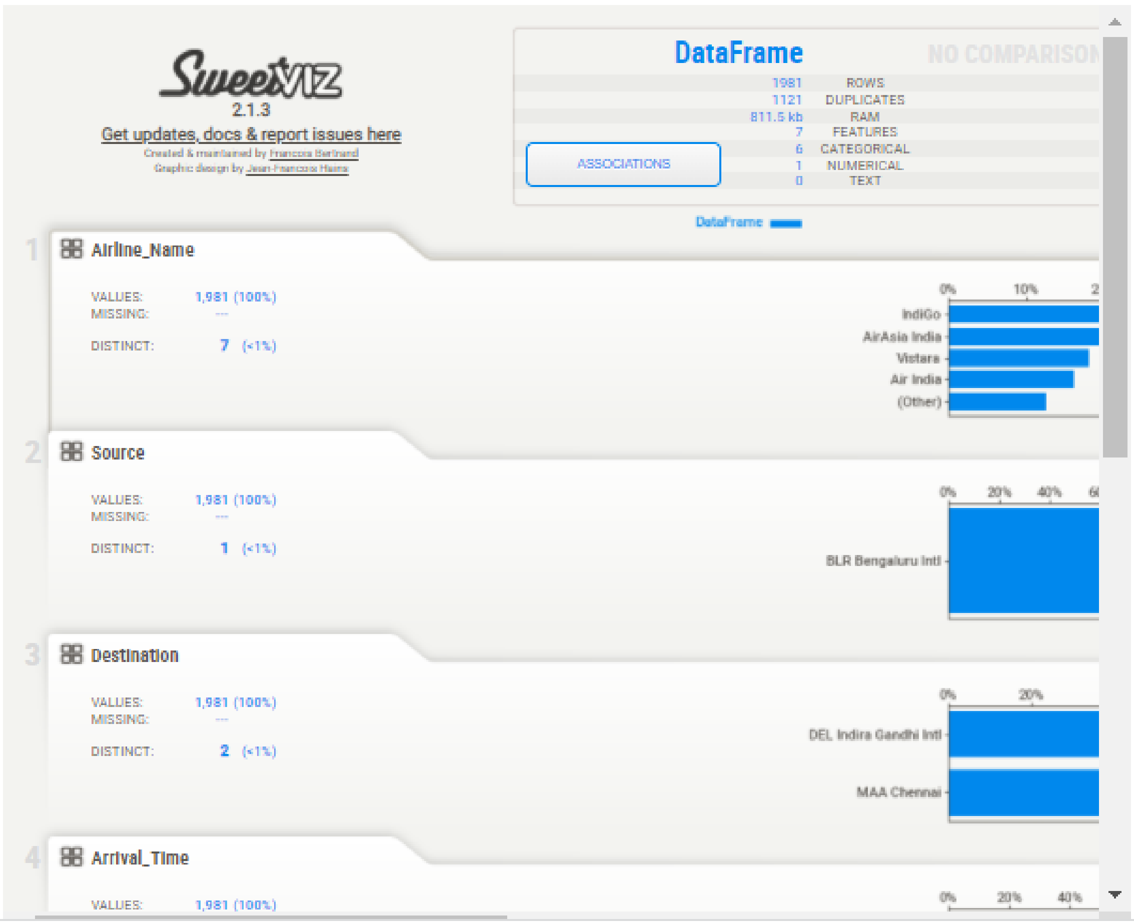
Predicted value

8116.52 (min) 14680.65 34937.69 (max)

negative positive

Feature	Value
Destination	-0.79
Number_of_Stops	-1.06
Airline_Name	-0.32
Arrival_Time	0.59
Departure_Time	-0.69
Source	0.00

- Sweetviz library dashboard that gives us results on association of various variables.



Sweetviz 2.1.3
Get updates, docs & report issues here
Created & maintained by Francois Bertrand
Graphic design by Jean-Francois Hains

DataFrame — NO COMPARISON

```
1981      ROWS
1121      DUPLICATES
811.5 kb  RAM
7         FEATURES
6         CATEGORICAL
1         NUMERICAL
0         TEXT
```

ASSOCIATIONS

1 Airline_Name
VALUES: 1,981 (100%)
MISSING: ---
DISTINCT: 7 (<1%)

2 Source
VALUES: 1,981 (100%)
MISSING: ---
DISTINCT: 1 (<1%)

3 Destination
VALUES: 1,981 (100%)
MISSING: ---
DISTINCT: 2 (<1%)

4 Arrival_Time
VALUES: 1,981 (100%)

CONCLUSION

- Key Findings and Conclusions of the Study

  - From the EDA , I could observe that pricing depends on the day and also on the brand of the airline.
  - An airline will charge more as they know customers prefer certain amenities and this also influences the mentality of a customer psychologically.

- From this dataset, I observed that flights having multiple stops charge more especially if there are a few flights for that day or flight that takes a specific route or destination.

● Learning Outcomes of the Study in respect of Data Science

- I realized how difficult it may be when it comes to combining both web scraping and analyzing as I spent more time on scraping the data.
- It also depends on the website you are referring to. Some websites have more details, some have less details. It’s best to focus on a selected few as every site has different paths and referencing it may become difficult.

● Limitation of this work and Scope for Future Work

- This dataset consists only of 1981 records and ideally it’s very less.
- There are only 7 features and all appear to be generic. When it comes to price prediction there is a possibility of various other factors like discounts, coupons, food provision etc.
- This dataset is only for one specific day and we cannot really say if prices would remain the same for other days.
- The only destinations in this dataset are Delhi and Chennai and the source is Bangalore and there is a small chance of getting different output due to lack of options.
- SInce the dataset is acquired from the scraped data, it's not necessary that we will get the same or similar data next time as dynamic websites keep changing the contents.