**Data Collection & Preparation**

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So this section allows you to download the required dataset.

### Collect The Dataset

There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.

In this project we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

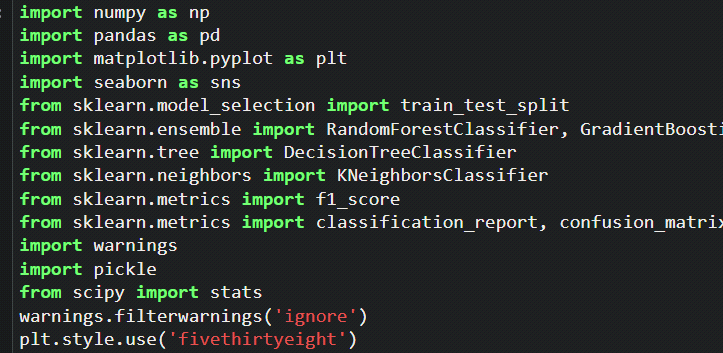
Link: <https://www.kaggle.com/code/anshigupta01/flight-price-prediction/data>

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.

**Note:** There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

**Importing The Libraries**

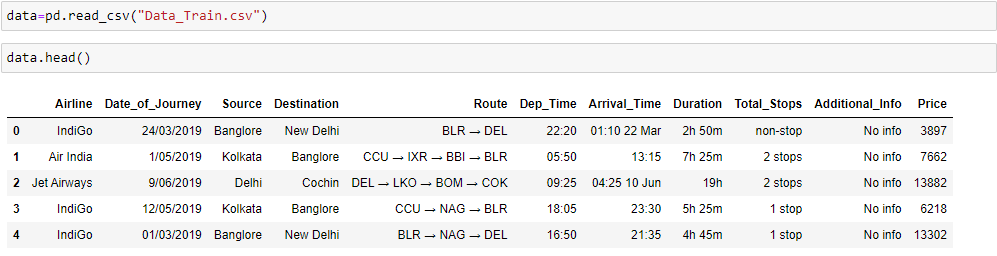
Import the necessary libraries as shown in the image. (optional) Here we have used the visualization style as FiveThirtyEight.



**Read The Dataset**

Our dataset format might be in .csv, excel files,.txt,.json, etc. We can read the dataset with the help of pandas.

In pandas we have a function called read\_csv() to read the dataset. As a parameter we have to give the directory of csv file.



**Data Preparation**

As we have understood how the data is let’s pre-process the collected data.

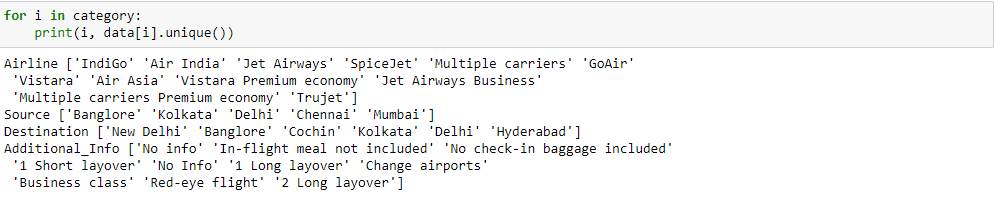
The download data set is not suitable for training the machine learning model as it might have so much of randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

* Handling missing values
* Handling categorical data
* Handling outliers
* Scaling Techniques
* Splitting dataset into training and test set

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

We have 1 missing value in Route column, and 1 missing value in Total stops column. We will meaningfully replace the missing values going further.

We now start exploring the columns available in our dataset. The first thing we do is to create a list of categorical columns, and check the unique values present in these columns.



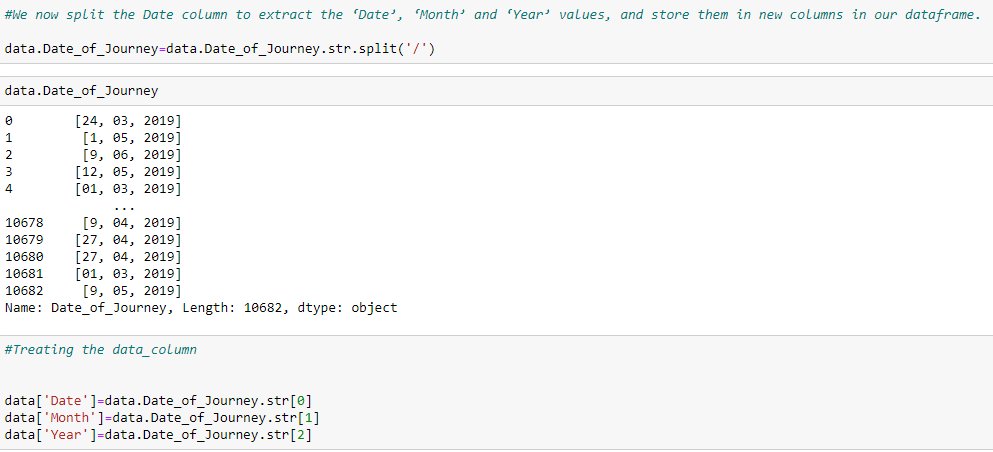
* 1. Airline column has 12 unique values - 'IndiGo' , 'Air India', 'Jet Airways' , 'SpiceJet' , 'Multiple carriers' , 'GoAir', 'Vistara', 'Air Asia', 'Vistara Premium economy' , 'Jet Airways Business', 'Multiple carriers Premium economy', 'Trujet'.

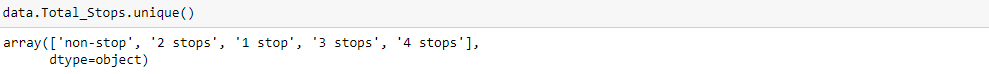
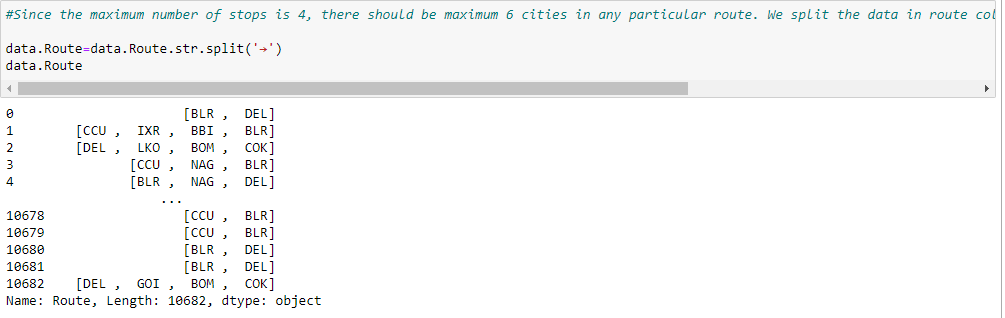
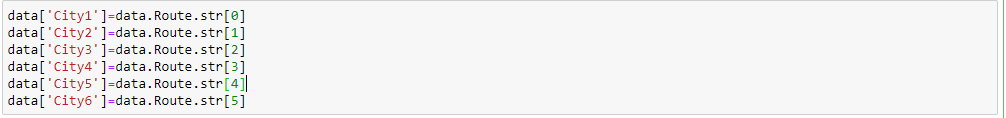
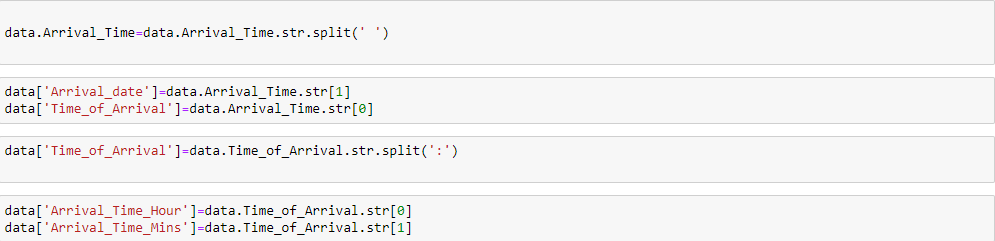
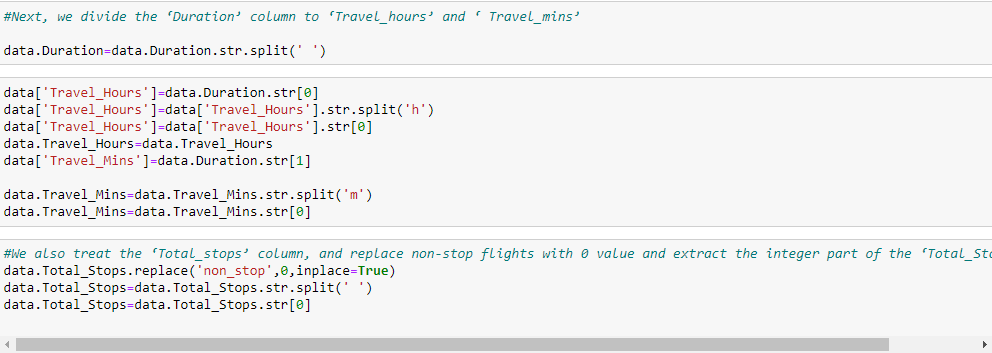
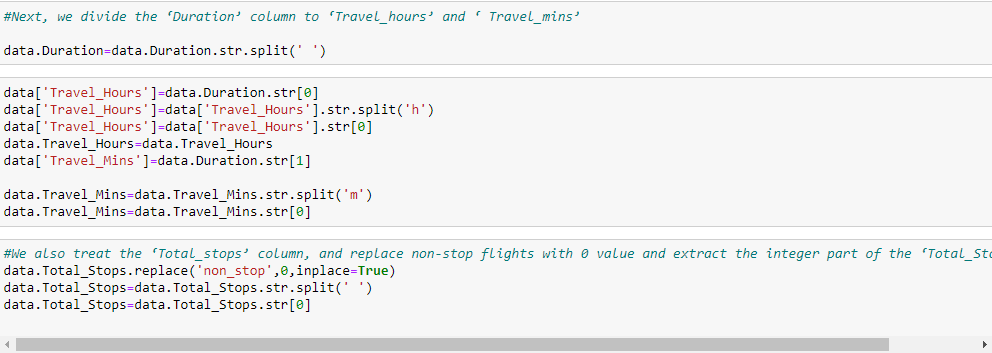
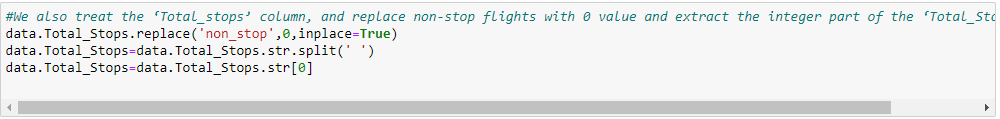
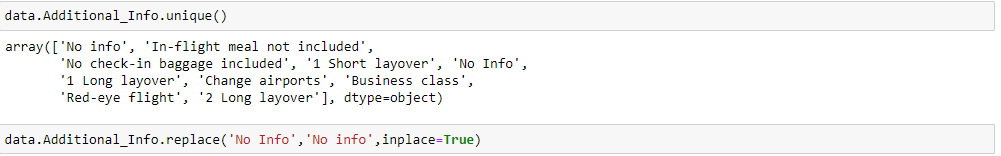
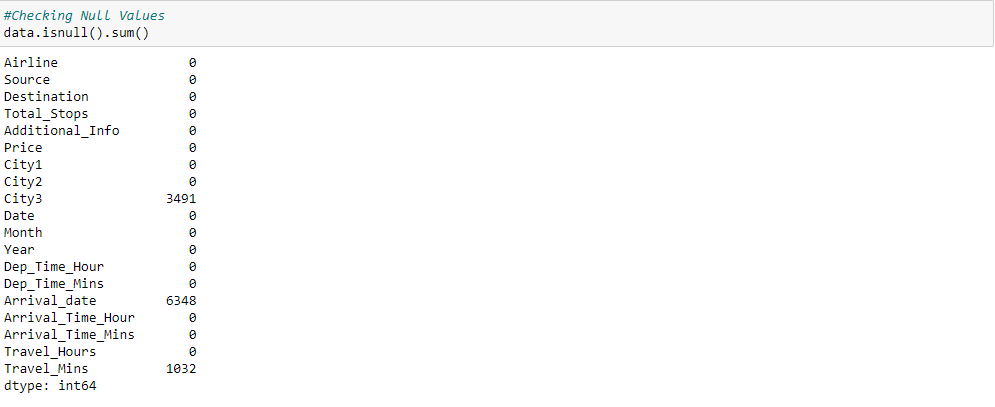
2. Source column has 5 unique values – ‘Bangalore’, ‘Kolkata’, ‘Chennai’, ‘Delhi’ and ‘Mumbai’.

3. Destination column has 6 unique values - 'New Delhi', 'Banglore', 'Cochin', 'Kolkata', 'Delhi' , 'Hyderabad'.

4. Additional info column has 10 unique values - 'No info', 'In-flight meal not included', 'No check-in baggage included', '1 Short layover' , 'No Info', '1 Long layover', 'Change airports' , 'Business class', 'Red-eye flight' , '2 Long layover'.

We now split the Date column to extract the ‘Date’, ‘Month’ and ‘Year’ values, and store them in new columns in our dataframe.



* Further, we split the Route column to create multiple columns with cities that the flight travels through. We check the maximum number of stops that a flight has, to confirm what should be the maximum number of cities in the longest route.
* 
* Since the maximum number of stops is 4, there should be maximum 6 cities in any particular route. We split the data in route column, and store all the city names in separate columns
* 
* 
* In the similar manner, we split the Dep\_time column, and create separate columns for departure hours and minutes.
  + 
* Further, for the arrival date and arrival time separation, we split the ‘Arrival\_Time’ column, and create ‘Arrival\_date’ column. We also split the time and divide it into ‘Arrival\_time\_hours’ and ‘Arrival\_time\_minutes’, similar to what we did with the ‘Dep\_time’ column.
  + 
* Next, we divide the ‘Duration’ column to ‘Travel\_hours’ and ‘ Travel\_mins’
  + 
  + 
* We also treat the ‘Total\_stops’ column, and replace non-stop flights with 0 value and extract the integer part of the ‘Total\_Stops’ column.
  + - 
* We proceed further to the ‘Additional\_info’ column, where we observe that there are 2 categories signifying ‘No info’, which are divided into 2 categories since ‘I’ in ‘No Info’ is capital. We replace ‘No Info’ by ‘No info’ to merge it into a single category.
  + - 
* We now drop all the columns from which we have extracted the useful information (original columns). We also drop some columns like ‘city4’,'city5' and ‘city6’, since majority of the data in these columns was NaN(null). As a result, we now obtain 20 different columns, which we will be feeding to our ML model. But first, we treat the missing values and explore the contents in the columns and its impact on the flight price, to separate a list of final set of columns.
  + - 
* After dropping some columns, here we can see the meaningful columns to predict the flight price without the NaN values.
  + - 

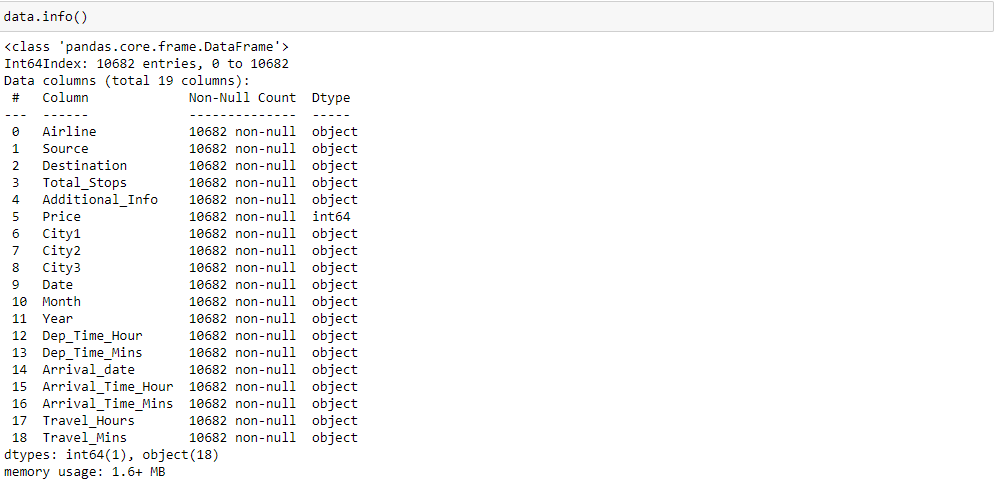
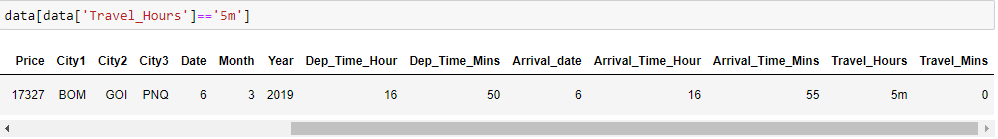
### Replacing Missing Values

We further replace ‘NaN’ values in ‘City3’ with ‘None’, since rows where ‘City3’ is missing did not have any stop, just the source and the destination.

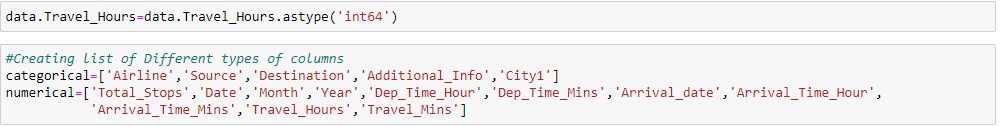
We also replace missing values in ‘Arrival\_date’ column with values in ‘Date’ column, since the missing values are those values where the flight took off and landed on the same date.

We also replace missing values in ‘Travel\_mins’ as 0, since the missing values represent that the travel time was in terms on hours only, and no additional minutes.



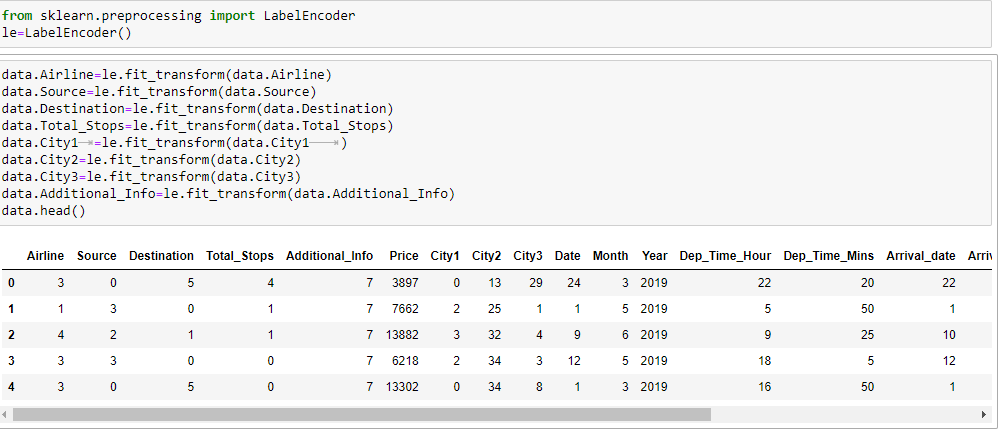
* Using the above steps, we were successfully able to treat all the missing values from our data. We again check the info in our data and find out that the dataset still has data types for multiple columns as ‘object’, where it should be ‘int’
*   
  + Hence, we try to change the datatype of the required columns
  +   
    - During this step, we face issue converting the ‘Travel\_hours’ column, saying that the column has data as ‘5m’, which is not allowing its conversion to ‘int’.
    -   
      * The data signifies that the flight time is ‘5m’, which is obviously wrong as the plane cannot fly from BOMBAY->GOA->PUNE->HYDERABAD in 5 mins! (The flight has ‘Total\_stops’ as 2)
      * https://lh6.googleusercontent.com/FtyM2oRbZYPEWih_bR-WgSSS7Ps0vTkH-liUJrXfUQ1V9um2FM2cMyHuHmu6zYNR6ll1GwTSwsCp85kd3-3O-gnV56PpVjUFAj1TeYJw_VCw_0bAKW862durEf7rZx4oUIS46Udnnu_4Jmx2UYKrg-4DccGrxehqnsiZ6KT0l-ye7uq-xgv3EABZP4gVJun_ZDXGNtY1-g  
        + We then convert the ‘Travel\_hours’ column to ‘int’ data type, and the operation happens successfully.

We now have a treated dataset with 10682 rows and 17 columns (16 independent and 1 dependent variable).



We create separate lists of categorical columns and numerical columns for plotting and analyzing the data

### Label Encoding

* Label encoding converts the data in machine readable form, but it assigns a unique number (starting from 0) to each class of data. it performs the conversion of categorical data into numeric format.
* In our dataset I have converted these variables 'Airline','Source','Destination','Total\_Stops','City1','City2','City3','Additional\_Info' into number format. So that it helps the model in better understanding of the dataset and enables the model to learn more complex structures.
* 

### Output Columns

* Initially in our dataset we have 19 features. So, in that some features are not more important to get output (Price).
* So i removed some unrelated features and I selected important features. So, it makes easy to understand. Now we have only 12 Output Columns.