INTRODUCTION TO DATA SCIENCE

Applying ML classification algorithms on Bike Sharing Dataset

Under the guidance of :

Dr. Sudheer Kumar Sharma
Dr. Aloke Datta
Dr. Indra Deep Mastan

GROUP 37 :

19UCC023 - Mohit Akhouri
19UCC026 - Divyansh Rastogi
19UCC119 - Abhinav Raj Upman
19UCS024 - Hardik Satish Bhati

Department of Computer Science and Engineering
The LNM Institute of Information Technology, Jaipur

ACKNOWLEDGEMENTS

We express our sincere thanks to our professors **Dr. Sudheer Kumar Sharma** , **Dr. Aloke Datta** and **Dr. Indra Deep Mastan** for giving us the opportunity to design a project based on ML Classification on bike sharing dataset . We are thankful to our professors for their timely advice , sharing their technical experiences and their scientific approach which have helped us in completing this project.
We are overwhelmed and indebted to everyone who has helped us develop our ideas and researching regarding this topic. We express our gratitude to our **parents** who are an inspiration for us. This project has helped us in increasing our skills and knowledge and expand the spectrum of our knowledge further more.

TABLE OF CONTENTS

PROBLEM DESCRIPTION

The problem statement given to us is as follows :

" Applying ML Classification algorithms on the data set and getting inferences from the data using appropriate ML algorithms. "

We will collect the dataset "Bike sharing dataset" from the **UCI Machine Learning Repository**. We will do the data cleaning, data preprocessing and analysis. We will later apply the ML Classification algorithms on the dataset to perform the capital BikeShare Predictive data analysis.

About Bike Sharing Systems :

Bike sharing systems are a great way to lead into the **efficient renting** of bikes. These systems have made the renting of bikes efficient and quick with memberships , multiple bike locations and easy rental and return process. One of the challenges of these bike sharing systems is the raucous weather conditions. Though this report , we are able to predict how different weather conditions affects the bike usage. In the dataset , there is a log of bikes rented in different weather conditions , different seasons and whether it was a weekday or holiday.

Our Aim :

We will apply regression analysis and classification algorithms to our dataset to make relevant predictions on bike sharing systems under harsh weather conditions. This would help the bike sharing systems deal with extreme conditions like very few bikes were rented or all of the bikes were rented.

INTRODUCTION TO THE DATASET

Bike Sharing Dataset

Link to our dataset : http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset#

This dataset contains the hourly and daily count of rental bikes between years 2011 and 2012 in Capital bikeshare system with the corresponding weather and seasonal information. Some of the information about our dataset is as follows :

Data Set Characteristics	Univariate
Attribute Characteristics	Integer, Real
Associated Tasks	Regression
Number of instances	17389
Number of Attributes	16
Missing Values	No
Area	Social
Number of web Hits	633424
Date donated	2013-12-20

More information regarding the dataset like - csv files , attribute information and processing done on the dataset is given in the subsequent pages.

For working on the dataset , we have downloaded the **Bike-Sharing-Dataset.zip** file containing the following subfiles :

- Readme.txt : This file contains all the information about the dataset.
- hour.csv : This csv file contains the bike sharing counts aggregated on hourly basis. This csv file contains **17379** records.
- day.csv : This csv file contains the bike sharing counts aggregated on daily basis. This csv file contains **731** records.

Attribute information :

(1) instant : tells about the unique sequential record index

(2) dteday : tells about the date the bike was rented

(3) season : tells about the season (1 for winter , 2 for spring , 3 for summer , 4 for fall) in which bike was rented.

(4) yr : tells about the year (0 for 2011 , 1 for 2012) in which the bike was rented.

(5) mnth : tells about the month (1 to 12) in which bike was rented.

(6) hr : tells about the hour (0 to 23) for which the bike was rented.

(7) holiday : information about whether the day was holiday or not.

(8) weekday : information about the day of the week.

(9) workingday : information about the working day (day is neither weekend nor holiday then workingday = 1 otherwise workingday = 0).

(10) <u>weathersit</u> : tells about the weather conditions on a particular day.

- Clear , Few clouds , Partly cloudy , Partly cloudy
- Mist + cloudy , Mist + broken clouds , Mist + Few clouds , Mist
- Light snow , Light rain + thunderstorm + Scattered clouds , Light rain + Scattered clouds
- Heavy rain + Ice Pallets + Thunderstorm + Mist , Snow + Fog

(11) <u>temp</u> : information about the normalized temperature in Celsius. The values are derived via the following equation :

$$(t - t_{min}) / (t_{max} - t_{min})$$

In the above equation t_{min} = -8 and t_{max} = +39 (both in hourly scale).

(12) <u>atemp</u> : information about the normalized feeling temperature in Celcius. The values are derived via the following equation :

$$(t - t_{min}) / (t_{max} - t_{min})$$

In the above equation t_{min} = -16 and t_{max} = +50 (both in hourly scale).

(13) <u>hum</u> : information about the normalized humidity. The values are divided to a maximum value of 100.

(14) <u>windspeed</u> : information about the normalized windspeed. The values are divided to a maximum value of 67.

(15) <u>casual</u> : information about the count of casual users.

(16) <u>registered</u> : information about the count of registered users.

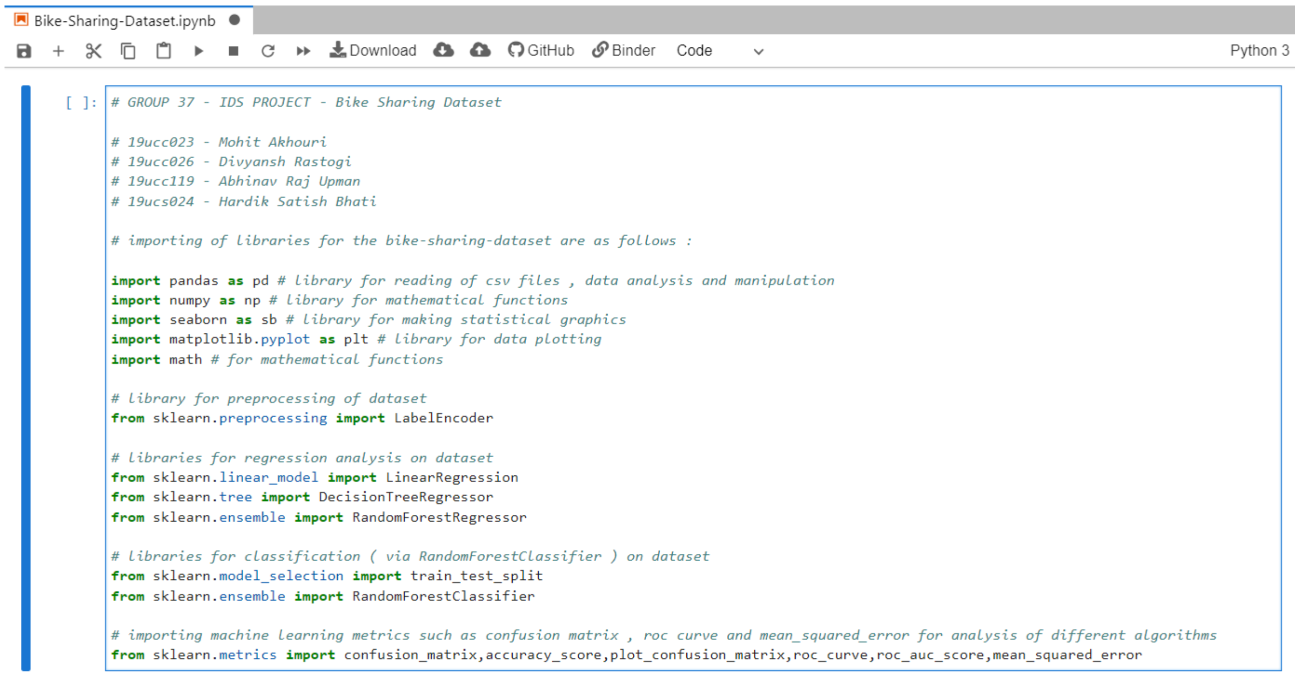
(17) <u>cnt</u> : information about the count of total rental bikes including both casual and registered users.

LNMIIT
The LNM Institute of Information Technology

TABULAR ANALYSIS OF DATASET

We will download the **Bike-Sharing-Dataset.zip** file and open it in Jupyter Notebook. Next we will write code for importing of libraries and display of the hour.csv and day.csv files.

We will **import the following python libraries** for the analysis , preprocessing , regression and classification of dataset :



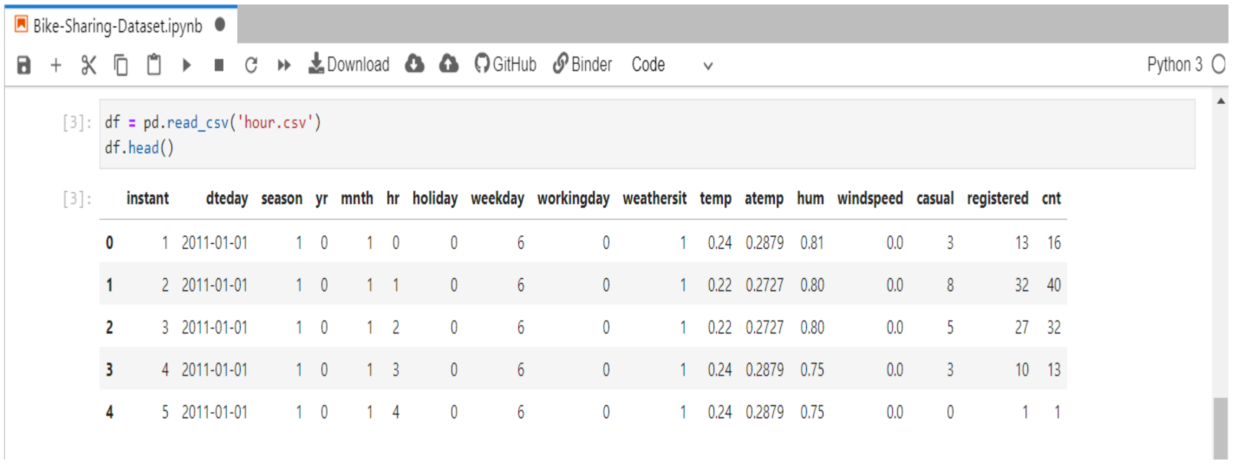
```
# GROUP 37 - IDS PROJECT - Bike Sharing Dataset

# 19ucc023 - Mohit Akhouri
# 19ucc026 - Divyansh Rastogi
# 19ucc119 - Abhinav Raj Upman
# 19ucs024 - Hardik Satish Bhati

# importing of libraries for the bike-sharing-dataset are as follows :

import pandas as pd # library for reading of csv files , data analysis and manipulation
import numpy as np # library for mathematical functions
import seaborn as sb # library for making statistical graphics
import matplotlib.pyplot as plt # library for data plotting
import math # for mathematical functions

# library for preprocessing of dataset
from sklearn.preprocessing import LabelEncoder

# libraries for regression analysis on dataset
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

# libraries for classification ( via RandomForestClassifier ) on dataset
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

# importing machine learning metrics such as confusion matrix , roc curve and mean_squared_error for analysis of different algorithms
from sklearn.metrics import confusion_matrix,accuracy_score,plot_confusion_matrix,roc_curve,roc_auc_score,mean_squared_error
```

The reason for importing the libraries are as follows :

After carefully analyzing the dataset , we came to the conclusion that since the dataset consists of **numerical values** and consists of labeled data , we will go for **regression analysis** on the dataset for prediction of bike rentals, so we have imported three libraries for regression analysis which are : **LinearRegression , DecisionTreeRegressor** and **RandomForestRegressor.**

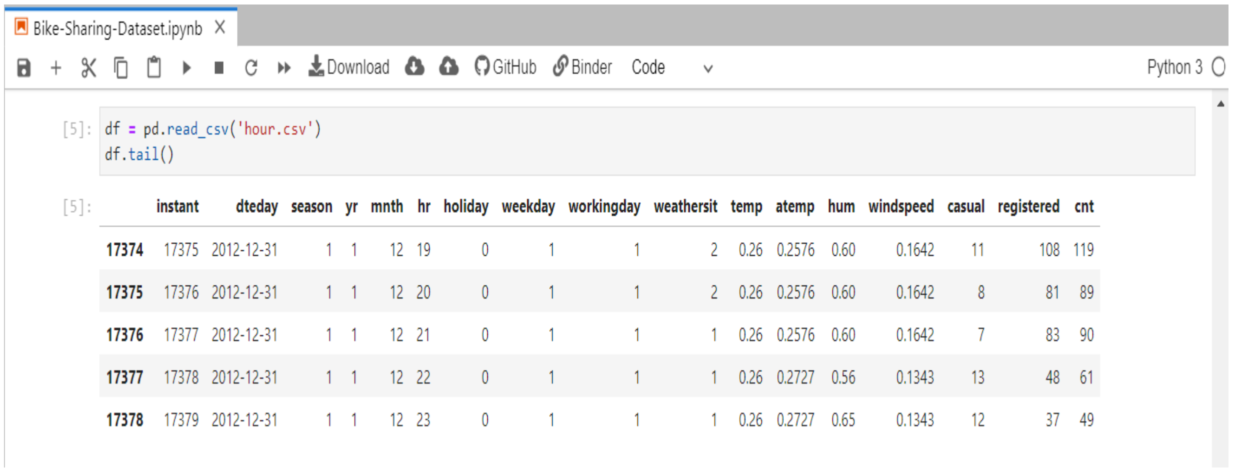
However, we could also use **RandomForestClassifier** which works on both labeled and numerical dataset. Hence we have also imported the libraries for classification of dataset which are : **train_test_split** and **RandomForestClassifier.**

Now we will further explore the data , first we will obtain a plot of the data contained in **hour.csv** and **day.csv** file with the help of pandas library.



```
[3]: df = pd.read_csv('hour.csv')
     df.head()

[3]:    instant     dteday  season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp   atemp   hum  windspeed  casual  registered  cnt
     0        1  2011-01-01       1   0     1   0        0        6           0           1  0.24  0.2879  0.81        0.0       3          13   16
     1        2  2011-01-01       1   0     1   1        0        6           0           1  0.22  0.2727  0.80        0.0       8          32   40
     2        3  2011-01-01       1   0     1   2        0        6           0           1  0.22  0.2727  0.80        0.0       5          27   32
     3        4  2011-01-01       1   0     1   3        0        6           0           1  0.24  0.2879  0.75        0.0       3          10   13
     4        5  2011-01-01       1   0     1   4        0        6           0           1  0.24  0.2879  0.75        0.0       0           1    1
```

In the above code , we have read the **hour.csv** file and with the help of **df.head()**, we obtained the first 5 records of the csv file. Similarly we can obtain the last 5 records of the csv file using the **df.tail()** which is done below.

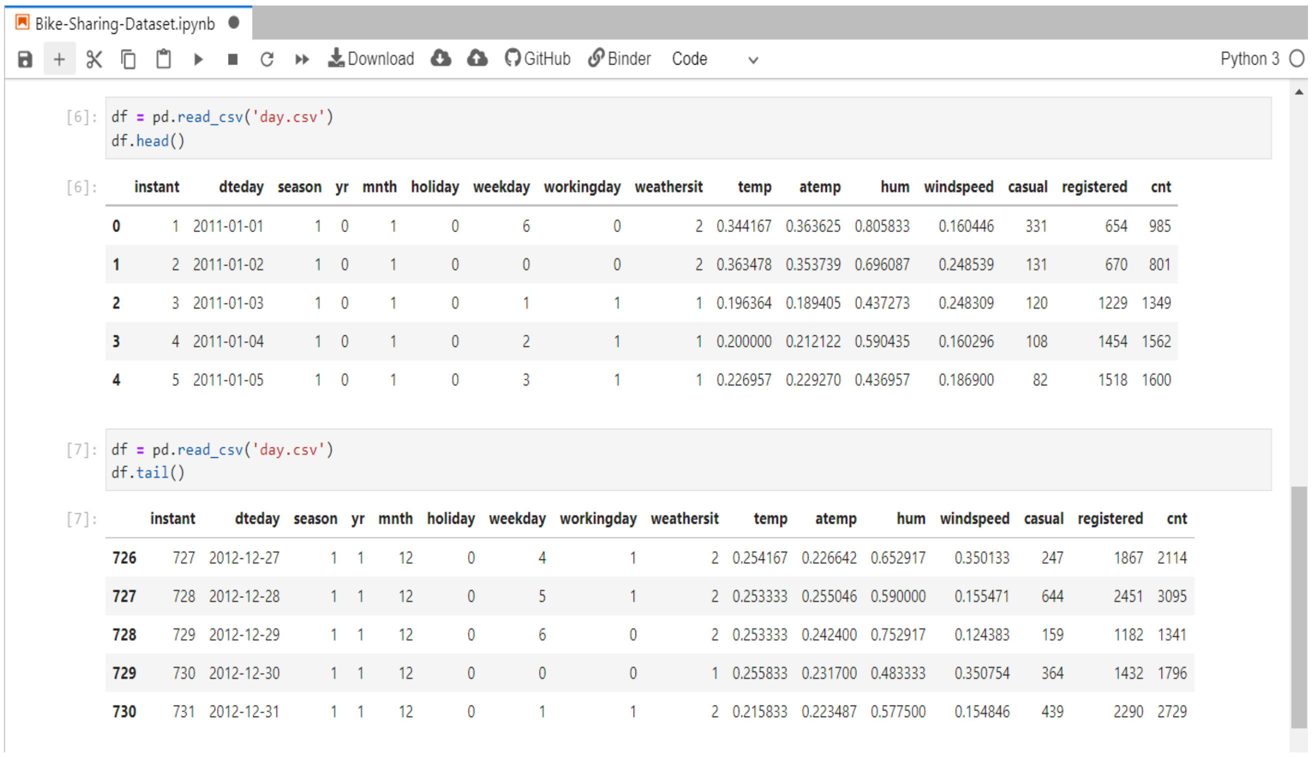


```
[5]: df = pd.read_csv('hour.csv')
     df.tail()

[5]:        instant     dteday  season  yr  mnth  hr  holiday  weekday  workingday  weathersit  temp   atemp   hum  windspeed  casual  registered  cnt
     17374    17375  2012-12-31       1   1    12  19        0        1           1           2  0.26  0.2576  0.60     0.1642      11         108  119
     17375    17376  2012-12-31       1   1    12  20        0        1           1           2  0.26  0.2576  0.60     0.1642       8          81   89
     17376    17377  2012-12-31       1   1    12  21        0        1           1           1  0.26  0.2576  0.60     0.1642       7          83   90
     17377    17378  2012-12-31       1   1    12  22        0        1           1           1  0.26  0.2727  0.56     0.1343      13          48   61
     17378    17379  2012-12-31       1   1    12  23        0        1           1           1  0.26  0.2727  0.65     0.1343      12          37   49
```

Using the pandas library we obtain the following inference about the hour.csv file :

Number of rows	17379
Number of columns	17

Now we will print the first 5 and last 5 records of day.csv file which are as follows :



```
[6]: df = pd.read_csv('day.csv')
     df.head()
```

```
[6]:   instant     dteday  season  yr  mnth  holiday  weekday  workingday  weathersit      temp     atemp       hum  windspeed  casual  registered   cnt
    0        1 2011-01-01       1   0     1        0        6           0           2  0.344167  0.363625  0.805833   0.160446     331         654   985
    1        2 2011-01-02       1   0     1        0        0           0           2  0.363478  0.353739  0.696087   0.248539     131         670   801
    2        3 2011-01-03       1   0     1        0        1           1           1  0.196364  0.189405  0.437273   0.248309     120        1229  1349
    3        4 2011-01-04       1   0     1        0        2           1           1  0.200000  0.212122  0.590435   0.160296     108        1454  1562
    4        5 2011-01-05       1   0     1        0        3           1           1  0.226957  0.229270  0.436957   0.186900      82        1518  1600
```

```
[7]: df = pd.read_csv('day.csv')
     df.tail()
```

```
[7]:     instant     dteday  season  yr  mnth  holiday  weekday  workingday  weathersit      temp     atemp       hum  windspeed  casual  registered   cnt
    726      727 2012-12-27       1   1    12        0        4           1           2  0.254167  0.226642  0.652917   0.350133     247        1867  2114
    727      728 2012-12-28       1   1    12        0        5           1           2  0.253333  0.255046  0.590000   0.155471     644        2451  3095
    728      729 2012-12-29       1   1    12        0        6           0           2  0.253333  0.242400  0.752917   0.124383     159        1182  1341
    729      730 2012-12-30       1   1    12        0        0           0           1  0.255833  0.231700  0.483333   0.350754     364        1432  1796
    730      731 2012-12-31       1   1    12        0        1           1           2  0.215833  0.223487  0.577500   0.154846     439        2290  2729
```

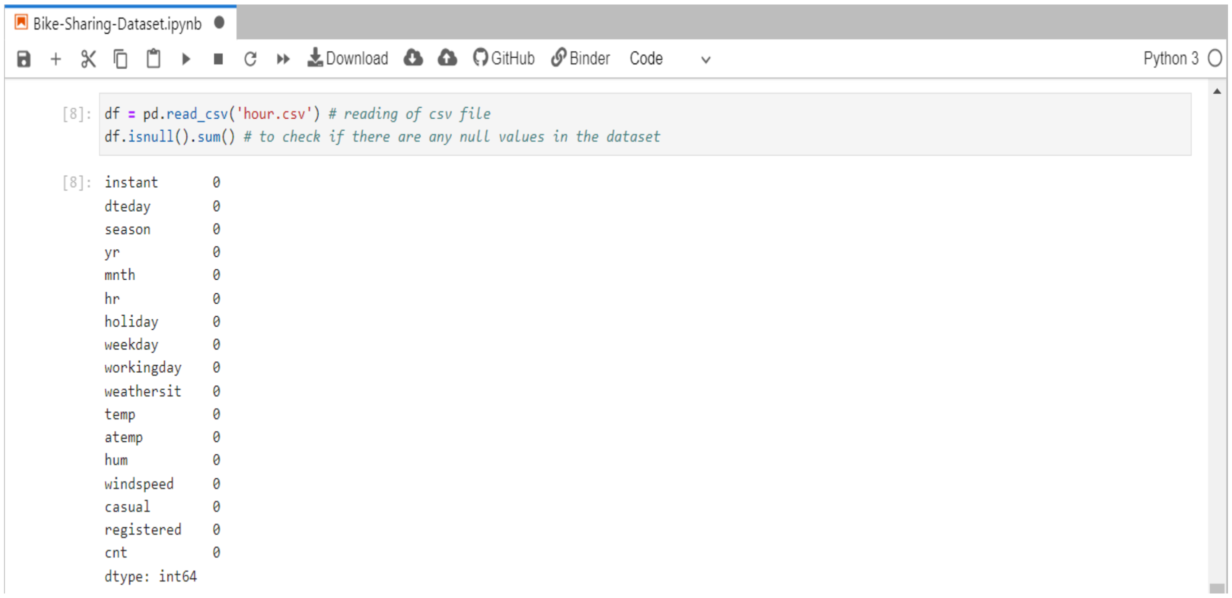
Using the pandas library we obtain the following inference about the day.csv file :

Number of rows	731
Number of columns	16

Now we will take help of python functions to derive inference about two important things about the dataset which are as follows :

- Checking NULL values in both hour.csv and day.csv files
- Column attributes of both csv files
- Basic statistical details about the dataset like Mean, SD and minimum-maximum values.

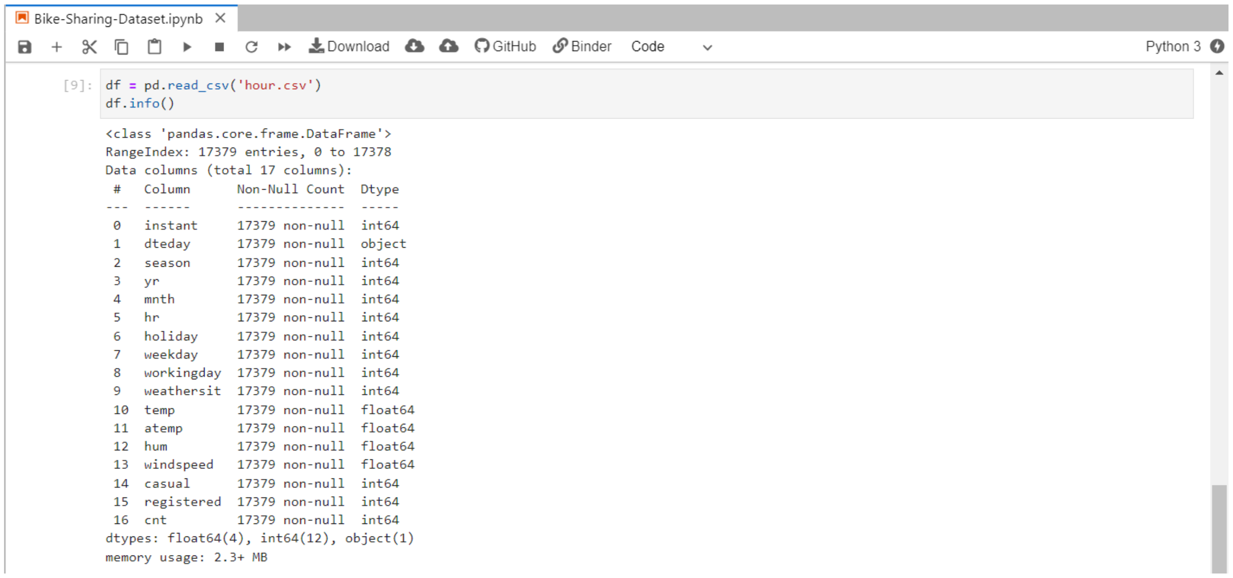
Analysis of hour.csv file :



```
[8]: df = pd.read_csv('hour.csv') # reading of csv file
     df.isnull().sum() # to check if there are any null values in the dataset

[8]: instant       0
     dteday        0
     season        0
     yr            0
     mnth          0
     hr            0
     holiday       0
     weekday       0
     workingday    0
     weathersit    0
     temp          0
     atemp         0
     hum           0
     windspeed     0
     casual        0
     registered    0
     cnt           0
     dtype: int64
```

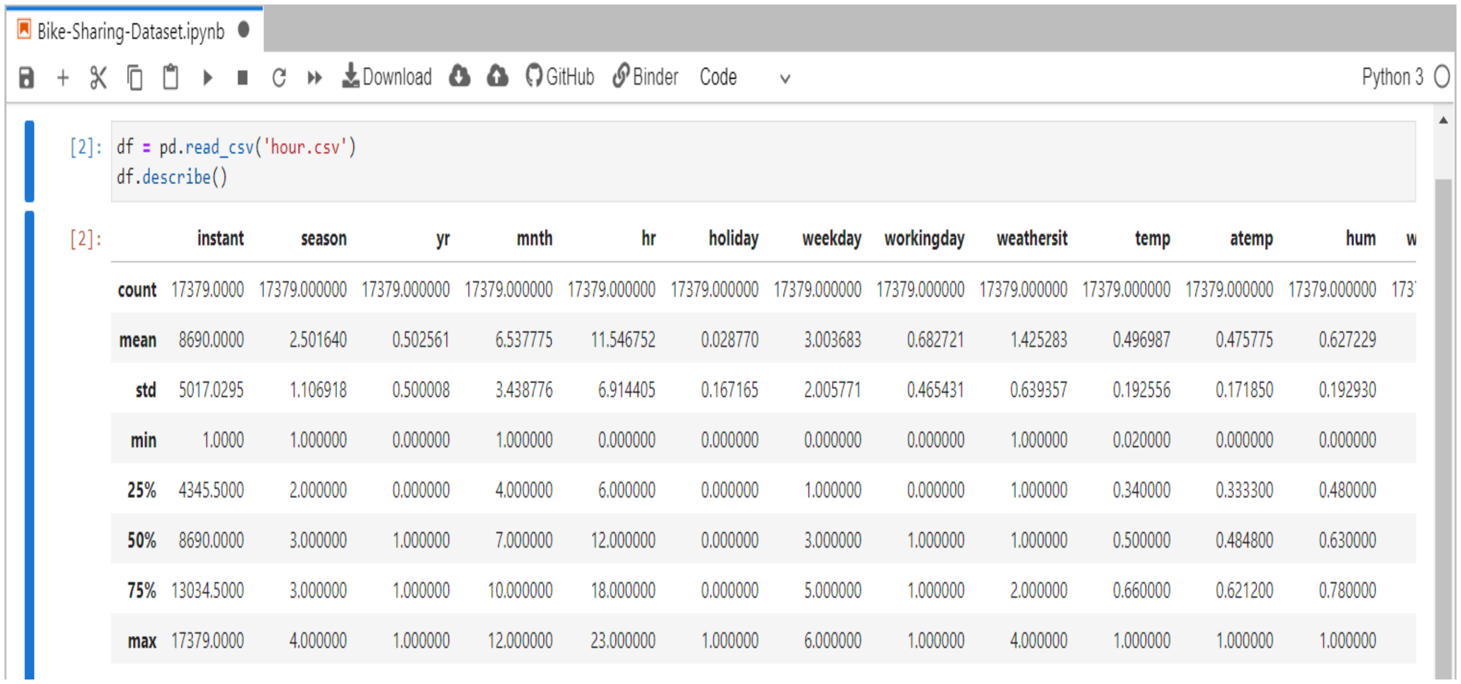
After analysis of the results obtained , we come to the conclusion that there are **no null values** in the dataset “hour.csv”. Now we will do the same analysis for the column attributes and basic statistical details about the “hour.csv” file.

We will use the **Dataframe.describe()** and **Dataframe.info()** python functions to perform the above mentioned tasks. The code and the results are described below :



```
[9]: df = pd.read_csv('hour.csv')
     df.info()

     <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 17379 entries, 0 to 17378
     Data columns (total 17 columns):
      #   Column      Non-Null Count  Dtype
     ---  ------      --------------  -----
      0   instant     17379 non-null  int64
      1   dteday      17379 non-null  object
      2   season      17379 non-null  int64
      3   yr          17379 non-null  int64
      4   mnth        17379 non-null  int64
      5   hr          17379 non-null  int64
      6   holiday     17379 non-null  int64
      7   weekday     17379 non-null  int64
      8   workingday  17379 non-null  int64
      9   weathersit  17379 non-null  int64
      10  temp        17379 non-null  float64
      11  atemp       17379 non-null  float64
      12  hum         17379 non-null  float64
      13  windspeed   17379 non-null  float64
      14  casual      17379 non-null  int64
      15  registered  17379 non-null  int64
      16  cnt         17379 non-null  int64
     dtypes: float64(4), int64(12), object(1)
     memory usage: 2.3+ MB
```

After carefully analyzing the results of **Dataframe.info()** , we came to the conclusion that the data in the csv file is of one the three types - int64 , float64 or object. The results obtained after executing function **Dataframe.describe()** is as follows :

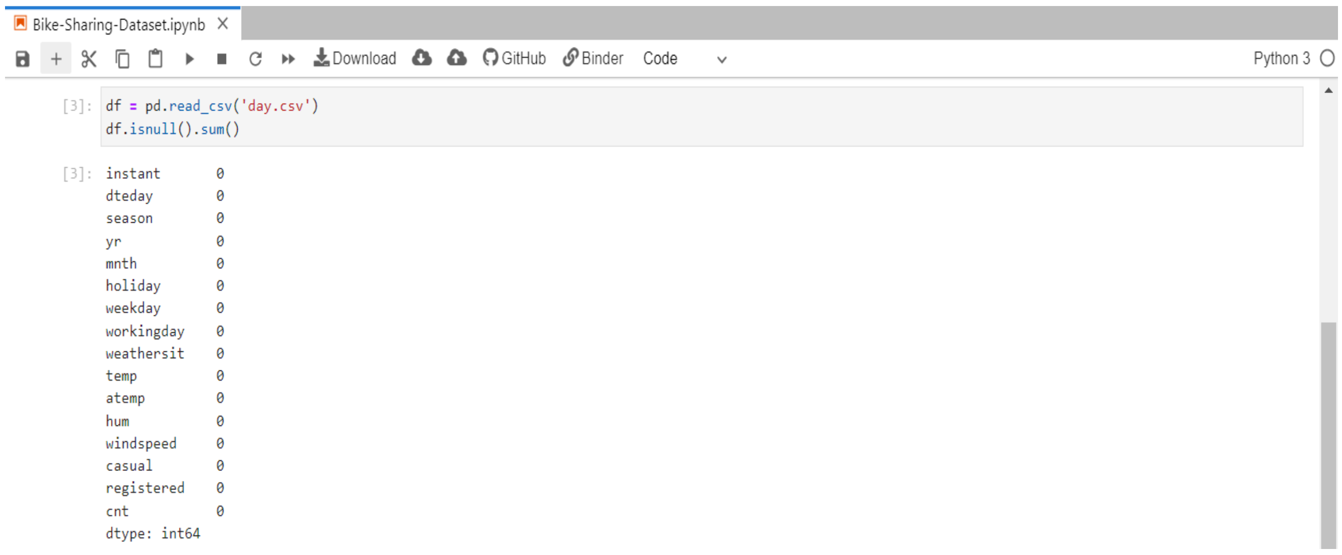


```
[2]: df = pd.read_csv('hour.csv')
     df.describe()
```

[2]:

	instant	season	yr	mnth	hr	holiday	weekday	workingday	weathersit	temp	atemp	hum	w
count	17379.0000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	17379.000000	173
mean	8690.0000	2.501640	0.502561	6.537775	11.546752	0.028770	3.003683	0.682721	1.425283	0.496987	0.475775	0.627229	
std	5017.0295	1.106918	0.500008	3.438776	6.914405	0.167165	2.005771	0.465431	0.639357	0.192556	0.171850	0.192930	
min	1.0000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	1.000000	0.020000	0.000000	0.000000	
25%	4345.5000	2.000000	0.000000	4.000000	6.000000	0.000000	1.000000	0.000000	1.000000	0.340000	0.333300	0.480000	
50%	8690.0000	3.000000	1.000000	7.000000	12.000000	0.000000	3.000000	1.000000	1.000000	0.500000	0.484800	0.630000	
75%	13034.5000	3.000000	1.000000	10.000000	18.000000	0.000000	5.000000	1.000000	2.000000	0.660000	0.621200	0.780000	
max	17379.0000	4.000000	1.000000	12.000000	23.000000	1.000000	6.000000	1.000000	4.000000	1.000000	1.000000	1.000000	

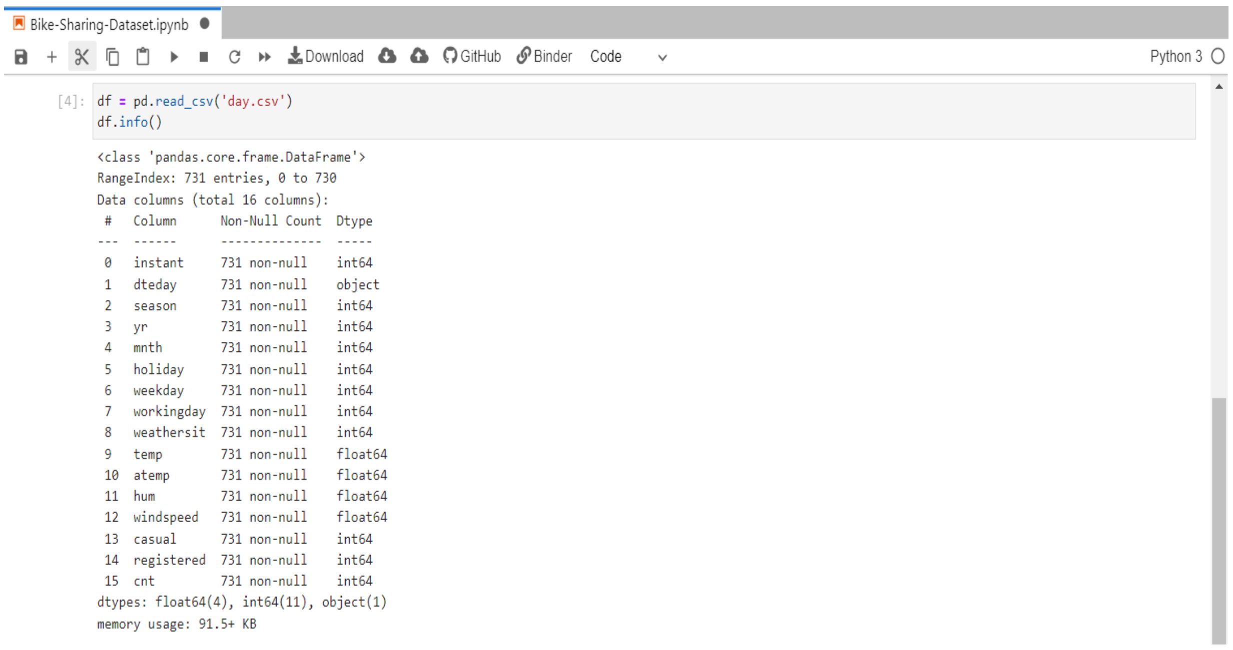
Analysis of day.csv file :



Bike-Sharing-Dataset.ipynb

```
[3]: df = pd.read_csv('day.csv')
     df.isnull().sum()

[3]: instant       0
     dteday        0
     season        0
     yr            0
     mnth          0
     holiday       0
     weekday       0
     workingday    0
     weathersit    0
     temp          0
     atemp         0
     hum           0
     windspeed     0
     casual        0
     registered    0
     cnt           0
     dtype: int64
```

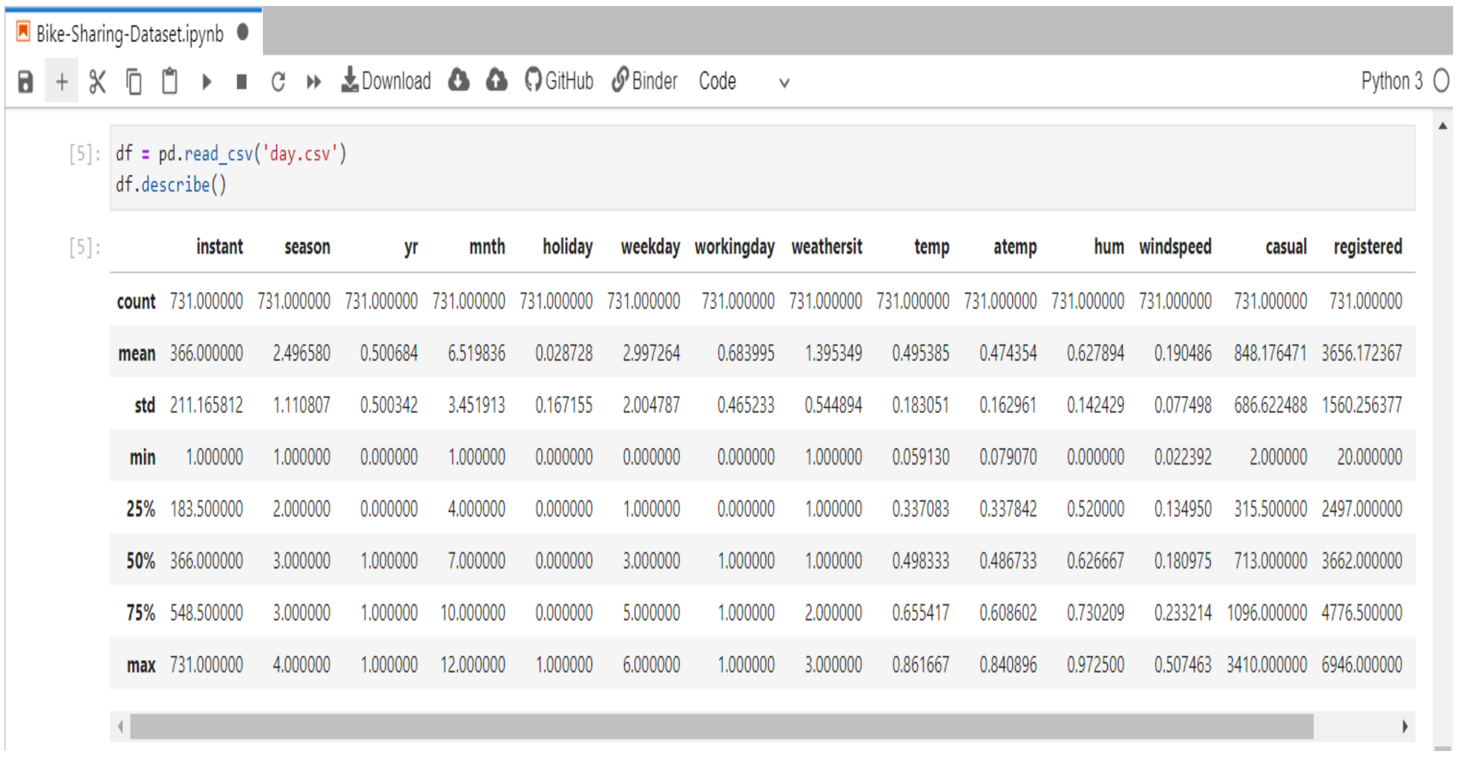
After analysis of the results obtained , we come to the conclusion that there are **no null values** in the dataset “day.csv”. Now we will do the same analysis for the column attributes and basic statistical details about the “hour.csv” file. The code and results are as follows :



Bike-Sharing-Dataset.ipynb

```
[4]: df = pd.read_csv('day.csv')
     df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 731 entries, 0 to 730
Data columns (total 16 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   instant     731 non-null    int64
 1   dteday      731 non-null    object
 2   season      731 non-null    int64
 3   yr          731 non-null    int64
 4   mnth        731 non-null    int64
 5   holiday     731 non-null    int64
 6   weekday     731 non-null    int64
 7   workingday  731 non-null    int64
 8   weathersit  731 non-null    int64
 9   temp        731 non-null    float64
 10  atemp       731 non-null    float64
 11  hum         731 non-null    float64
 12  windspeed   731 non-null    float64
 13  casual      731 non-null    int64
 14  registered  731 non-null    int64
 15  cnt         731 non-null    int64
dtypes: float64(4), int64(11), object(1)
memory usage: 91.5+ KB
```

After carefully analyzing the results of **Dataframe.info()** , we came to the conclusion that the data in the csv file is of one the three types - int64 , float64 or object. The results obtained after executing function **Dataframe.describe()** is as follows :



```
[5]: df = pd.read_csv('day.csv')
     df.describe()
```

[5]:

	instant	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	366.000000	2.496580	0.500684	6.519836	0.028728	2.997264	0.683995	1.395349	0.495385	0.474354	0.627894	0.190486	848.176471	3656.172367
std	211.165812	1.110807	0.500342	3.451913	0.167155	2.004787	0.465233	0.544894	0.183051	0.162961	0.142429	0.077498	686.622488	1560.256377
min	1.000000	1.000000	0.000000	1.000000	0.000000	0.000000	0.000000	1.000000	0.059130	0.079070	0.000000	0.022392	2.000000	20.000000
25%	183.500000	2.000000	0.000000	4.000000	0.000000	1.000000	0.000000	1.000000	0.337083	0.337842	0.520000	0.134950	315.500000	2497.000000
50%	366.000000	3.000000	1.000000	7.000000	0.000000	3.000000	1.000000	1.000000	0.498333	0.486733	0.626667	0.180975	713.000000	3662.000000
75%	548.500000	3.000000	1.000000	10.000000	0.000000	5.000000	1.000000	2.000000	0.655417	0.608602	0.730209	0.233214	1096.000000	4776.500000
max	731.000000	4.000000	1.000000	12.000000	1.000000	6.000000	1.000000	3.000000	0.861667	0.840896	0.972500	0.507463	3410.000000	6946.000000

Now since the analysis of datasets hour.csv and day.csv , we will do some plotting of the data obtained with the help of following functions :

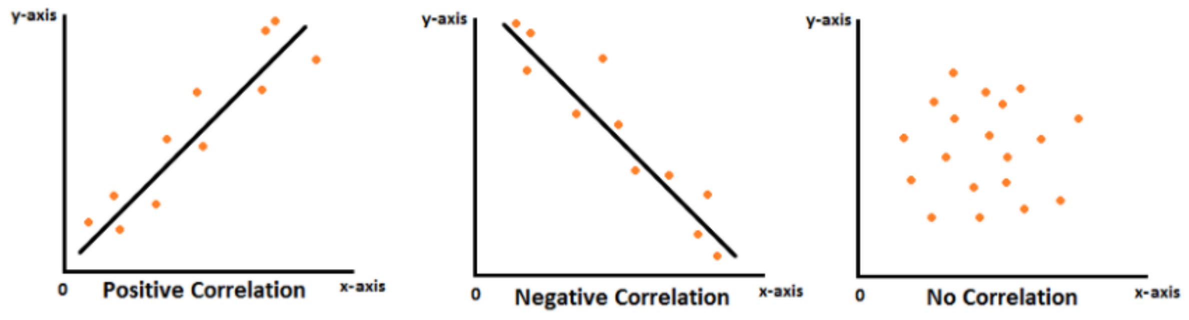
- Matplotlib library
- Seaborn library

After plotting of data is done , we will go for **Feature engineering** and finally the deployment of machine learning algorithms for the prediction of data. Now let us analyse the results obtained via the above functions graphically.

GRAPHICAL ANALYSIS OF DATASET

(1) About Correlation of attributes :

We can understand the concept of correlation with the help of the following plots.



Correlation means the degree to which a pair of variables are linearly related. Correlation follows causation relationship which means that a change in one variable causes another variable to change also. Now after looking at the above figure , we can determine that there are two types of correlation which are as follows :

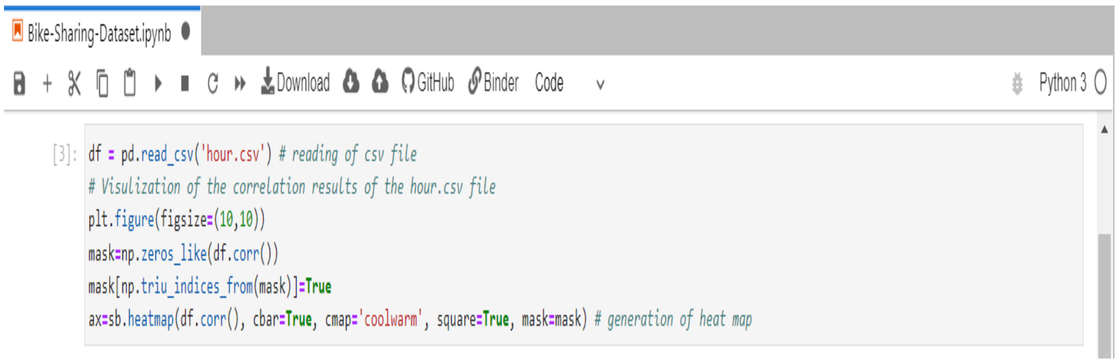
(1) Positive correlation : In positive correlation , if one variable increases then the other variable increases and same if one variable decreases then the other variable decreases. The slope of the **fitting line** in such a case would be **positive.**

(2) Negative correlation : In negative correlation , the inverse relationship is there between the correlated variables , that is if one variable increases/decreases then the other variable decreases/increases.

Now to obtain the correlation between the attributes , we have the **seaborn** library in Python. So now we will use the **heatmap** function of the seaborn library to obtain correlation between attributes of both datasets - hour.csv and day.csv

(1a) Visualization of heat map for the hour.csv dataset :

The code used for the generation of heatmap is as follows :

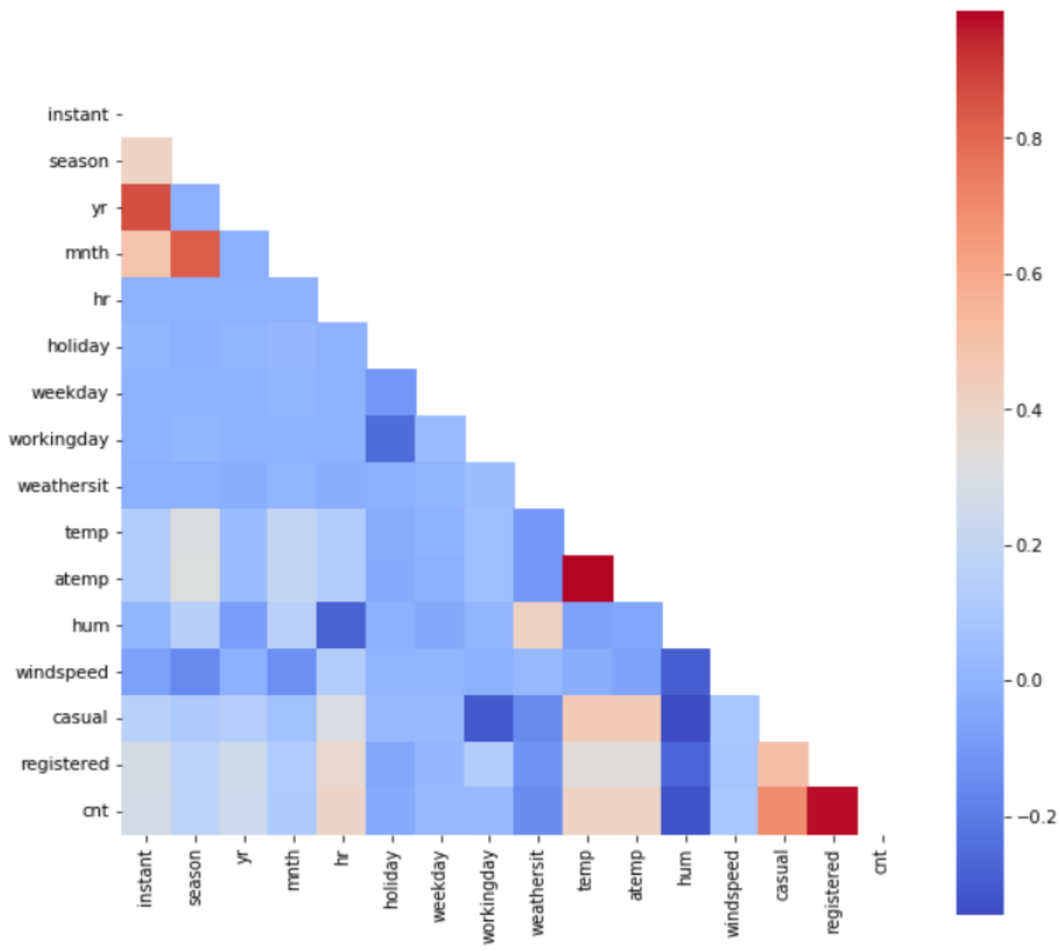


```
[3]: df = pd.read_csv('hour.csv') # reading of csv file
     # Visulization of the correlation results of the hour.csv file
     plt.figure(figsize=(10,10))
     mask=np.zeros_like(df.corr())
     mask[np.triu_indices_from(mask)]=True
     ax=sb.heatmap(df.corr(), cbar=True, cmap='coolwarm', square=True, mask=mask) # generation of heat map
```

In the above code , first we have read the csv file using the **pd.read_csv** function and then we have made use of the **heatmap** function of the **seaborn library** to obtain a heatmap between the different attributes of hour.csv

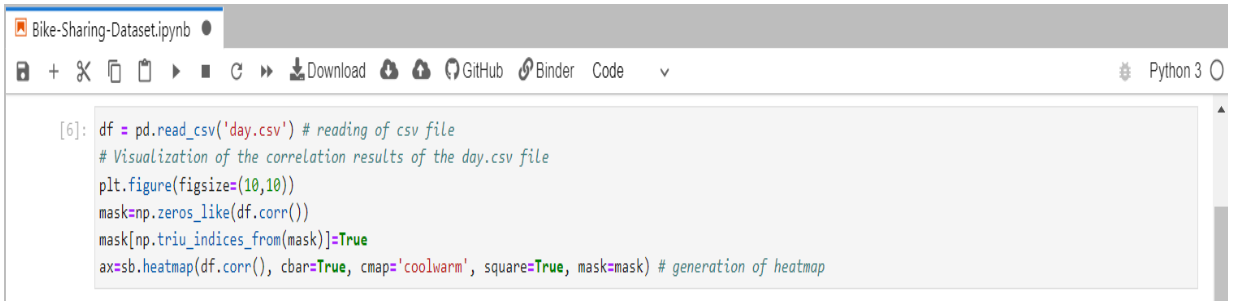
The heatmap obtained is given below. From the heatmap we can observe that the attributes **yr** and **instant** are much closely related to each other as their heat signatures are falling in the higher range. They are **much closely related** means that they have equal effect on the dataset , meaning that removal of one of these attributes may not affect the dataset. Some other attributes that are much closely related are **temp** and **atemp.**

On the other hand , attributes that are not so much closely related to each other are the ones that are in pair with the **hum** column of the dataset. Similarly , we can look for the correlation between different types of attributes.



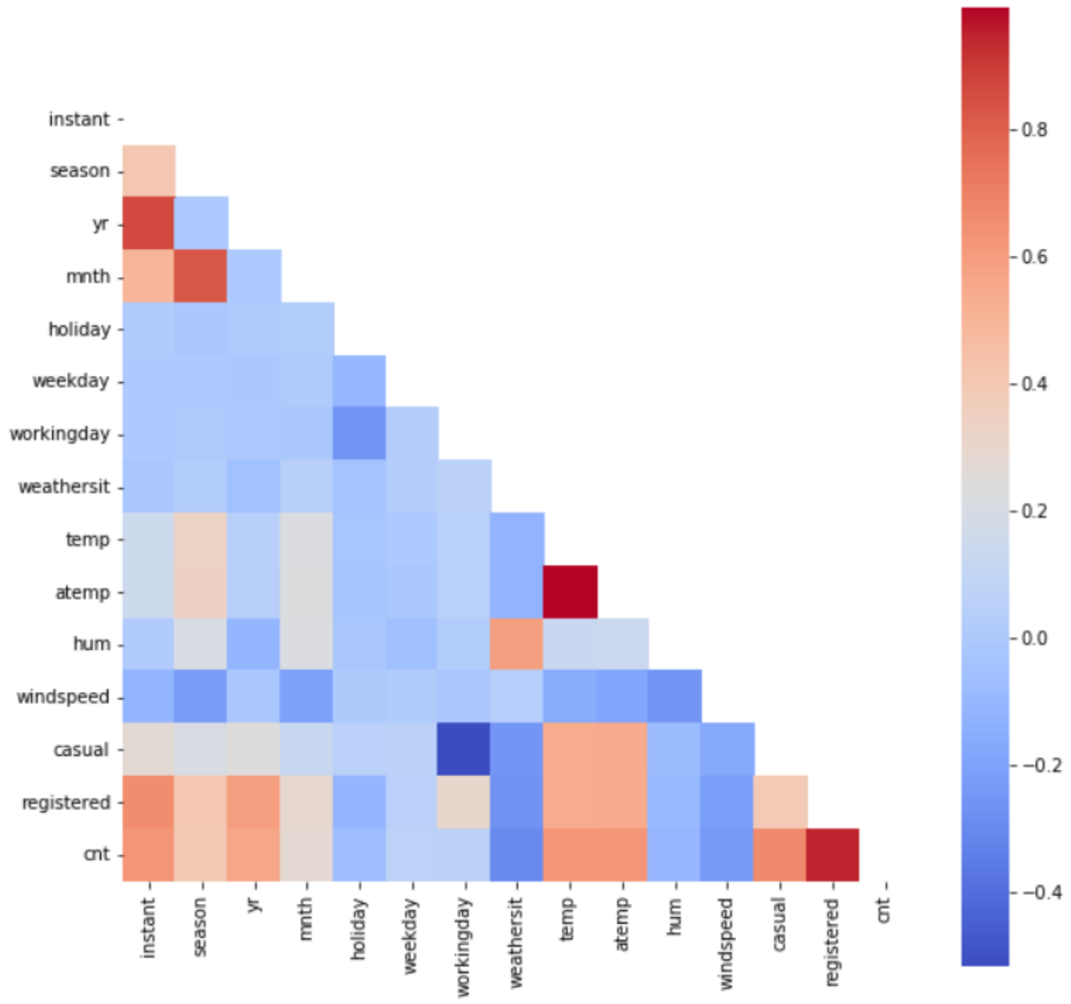
(1b) Visualization of heat map for the day.csv dataset :

The code used for the generation of heatmap is as follows :



```
[6]: df = pd.read_csv('day.csv') # reading of csv file
     # Visualization of the correlation results of the day.csv file
     plt.figure(figsize=(10,10))
     mask=np.zeros_like(df.corr())
     mask[np.triu_indices_from(mask)]=True
     ax=sb.heatmap(df.corr(), cbar=True, cmap='coolwarm', square=True, mask=mask) # generation of heatmap
```

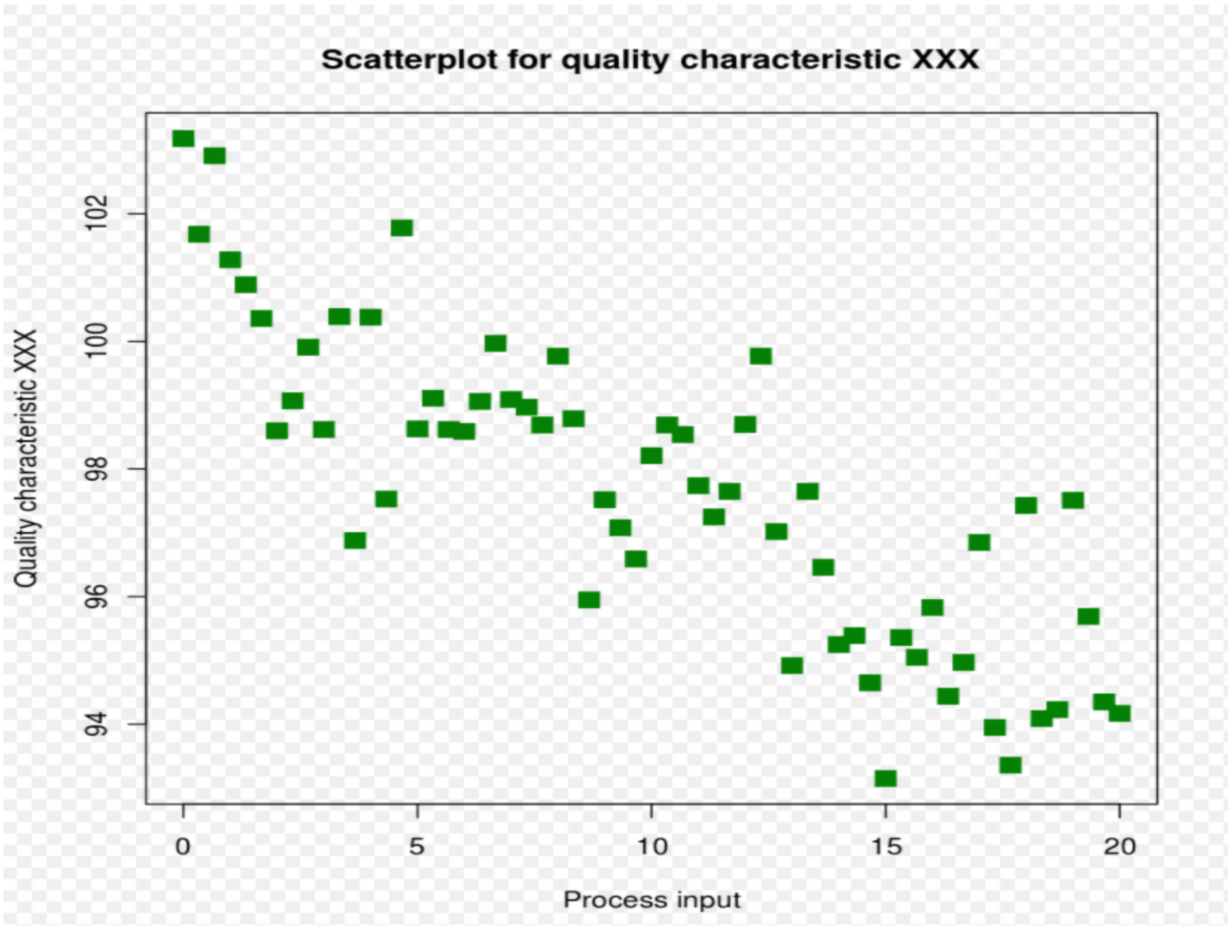
In the above code , first we have read the csv file using the **pd.read_csv** function and then we have made use of the **heatmap** function of the **seaborn library** to obtain a heatmap between the different attributes of day.csv



The heatmap obtained is given above. From the heatmap we can observe that the attributes **temp** and **atemp** are much closely related (higher correlation) , which means that both of these observations have equal effect on the dataset. It means that removal of one of these attributes does not affect the dataset.

From the heatmap , we can also observe that attributes **workingday** and **casual** have **low correlation** (not much closely related to each other).

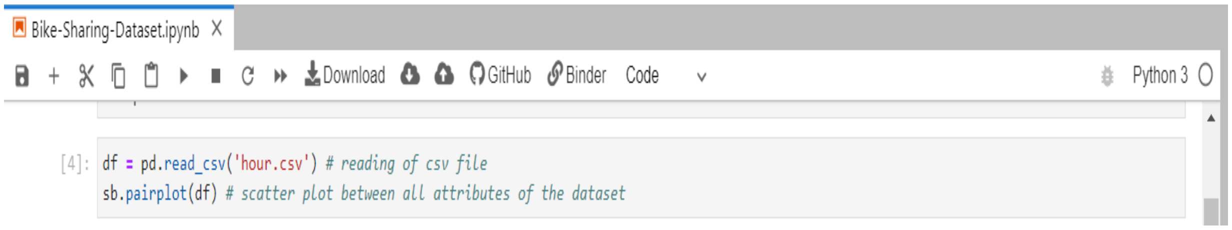
(2) About Scatter Plot :



Scatter Plot is yet another way of representation of attributes of a dataset. It is basically a type of plot or **mathematical diagram** which uses **cartesian coordinates** for the display of values. It uses two variables for a set of data , one on the X-axis and one on the Y-axis. In scatter plot , data is represented in form of points on the cartesian plane , with one variable determining the position on the horizontal axis and the other variable determining the position on the vertical axis.

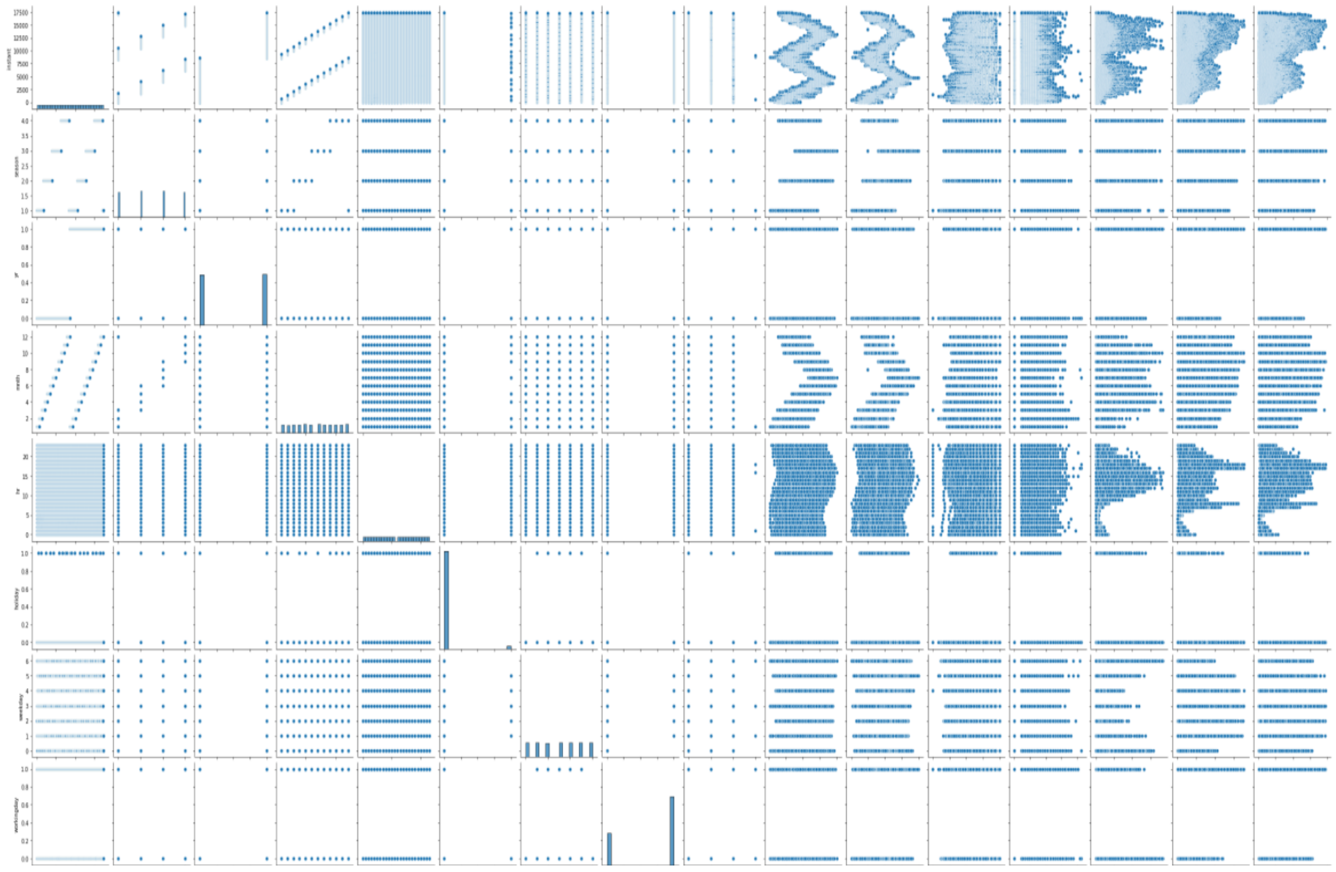
In Python , we use the **pairplot** function of the **seaborn library** to make a scatter plot. If we don't pass any attributes to the pairplot function, then it will construct a scatter plot between all attributes of the dataset.

(2a) Visualization of scatter plot between the attributes of hour.csv :

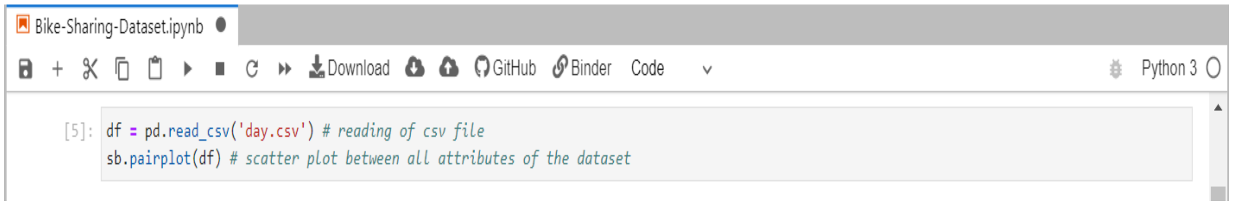


```
[4]: df = pd.read_csv('hour.csv') # reading of csv file
     sb.pairplot(df) # scatter plot between all attributes of the dataset
```

The code for the scatter plot of hour.csv file is given above. In this we have not provided any parameters in the pairplot function hence resulting in the scatter plot between all attributes of the dataset. The corresponding snapshot of the scatter plot is as follows :

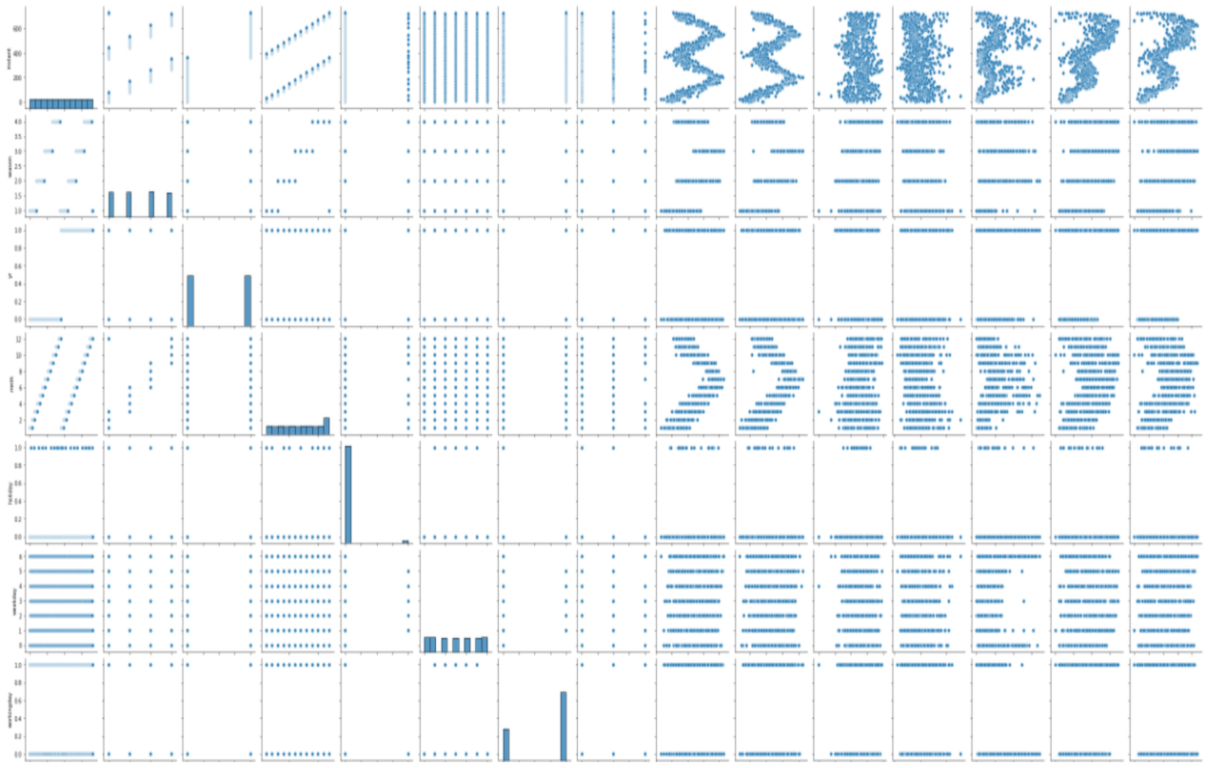


(2b) Visualization of scatter plot between the attributes of day.csv :



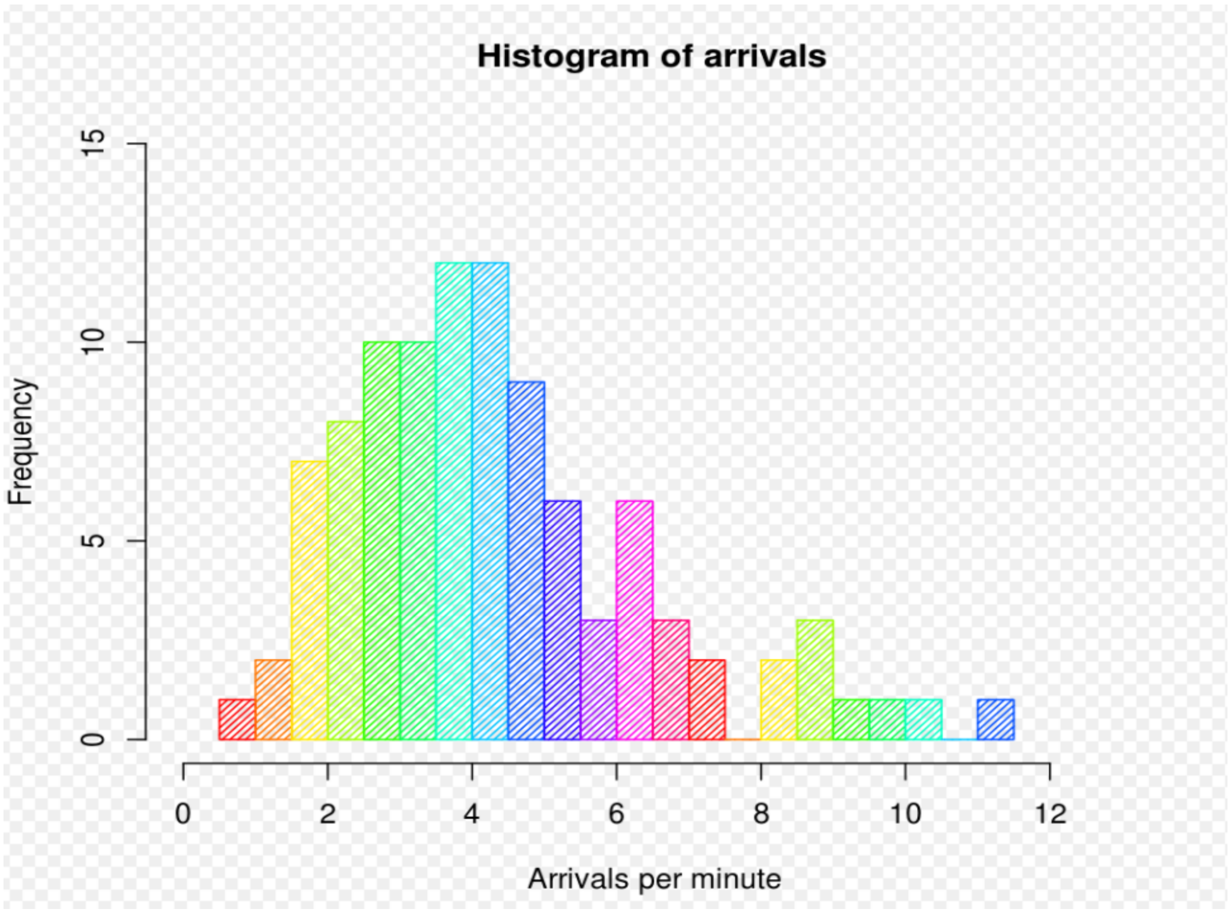
```
[5]: df = pd.read_csv('day.csv') # reading of csv file
     sb.pairplot(df) # scatter plot between all attributes of the dataset
```

The code for the scatter plot of day.csv file is given above. In this we have not provided any parameters in the pairplot function hence resulting in the scatter plot between all attributes of the dataset. The corresponding snapshot of the scatter plot is as follows :



Now for further processing of data - **prediction** and **model comparison** , we have created a new combined csv file - **bike_rental_hour.csv.**

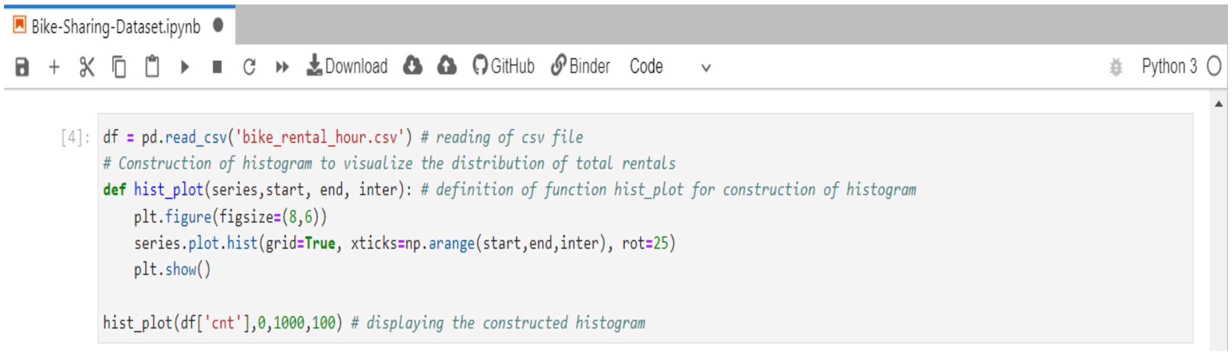
(3) About Histogram :



Histogram is yet another way of representation of **numerical data.** It is an approximate representation of numerical data distribution. Histogram was first introduced by **Karl Pearson**. Some of the steps involved in constructing histogram are as follows : **decision of bin size, count values falling in a particular bin** and creating **consecutive non-overlapping** rectangles in each bin. We can think of histogram as a simplistic kernel density estimation. It uses a kernel to smooth frequencies over the bins.

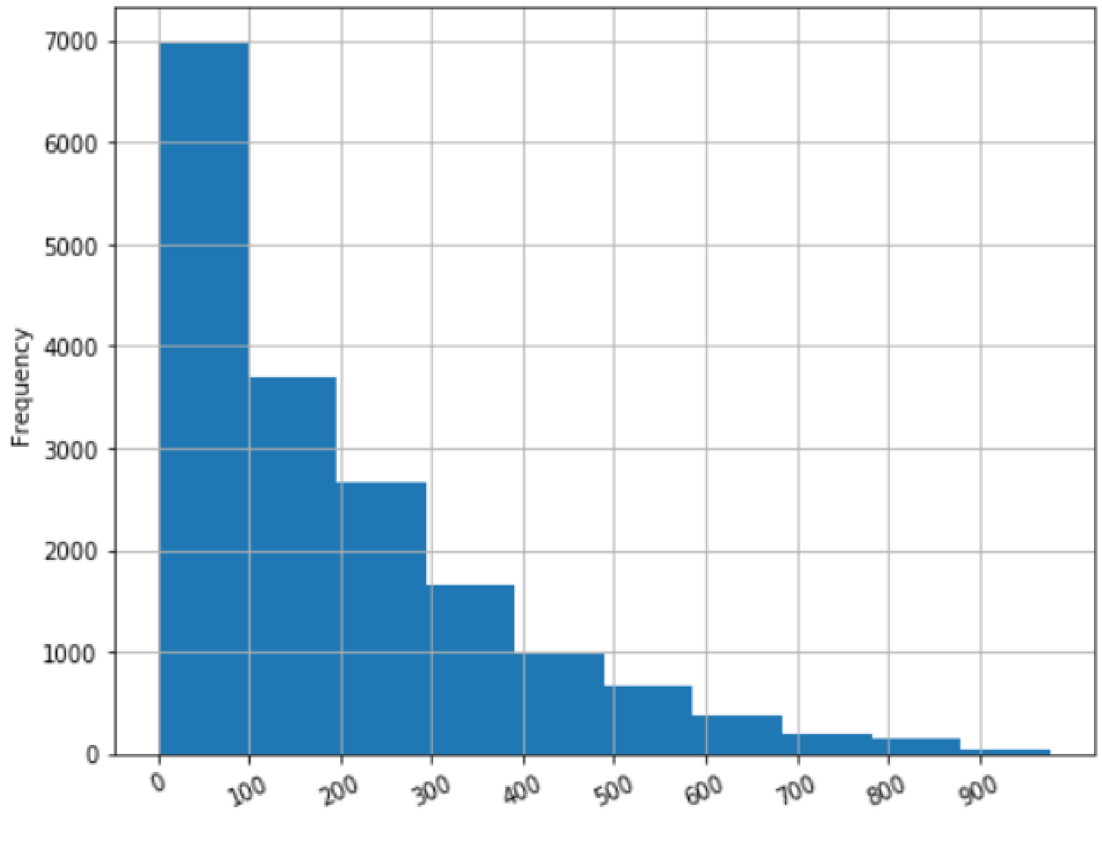
To construct a histogram in python , we can define a hist_plot function. The code and corresponding results are as follows.

(3a) Visualization of histogram to see distribution of total rentals :

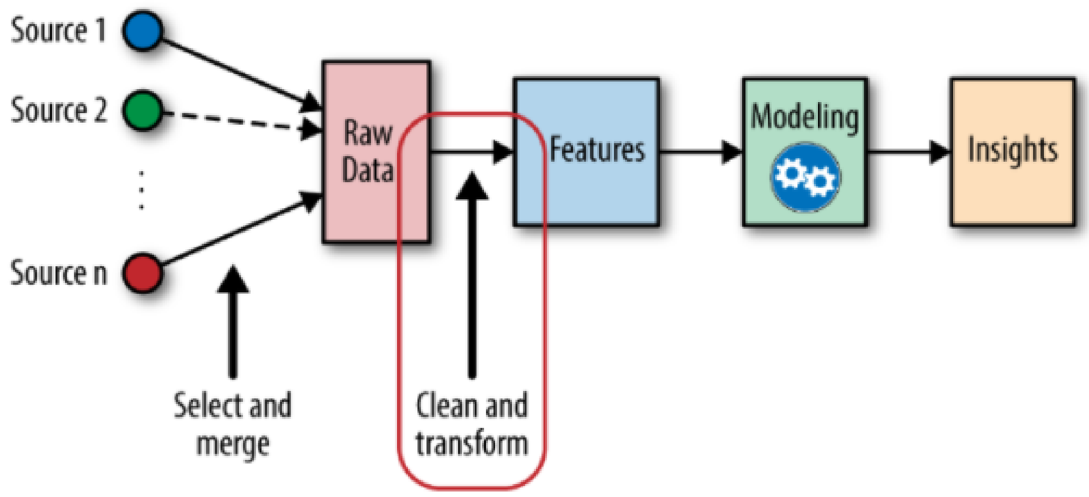


```
[4]: df = pd.read_csv('bike_rental_hour.csv') # reading of csv file
     # Construction of histogram to visualize the distribution of total rentals
     def hist_plot(series,start, end, inter): # definition of function hist_plot for construction of histogram
         plt.figure(figsize=(8,6))
         series.plot.hist(grid=True, xticks=np.arange(start,end,inter), rot=25)
         plt.show()

     hist_plot(df['cnt'],0,1000,100) # displaying the constructed histogram
```

The code for the construction of histogram is as given above. In this we are trying to visualize the distribution of total bike rentals. We are doing this because in the subsequent topics like feature engineering and model error comparison , we need to do prediction of bike rentals which required visualization of the **target feature**.



FEATURE ENGINEERING



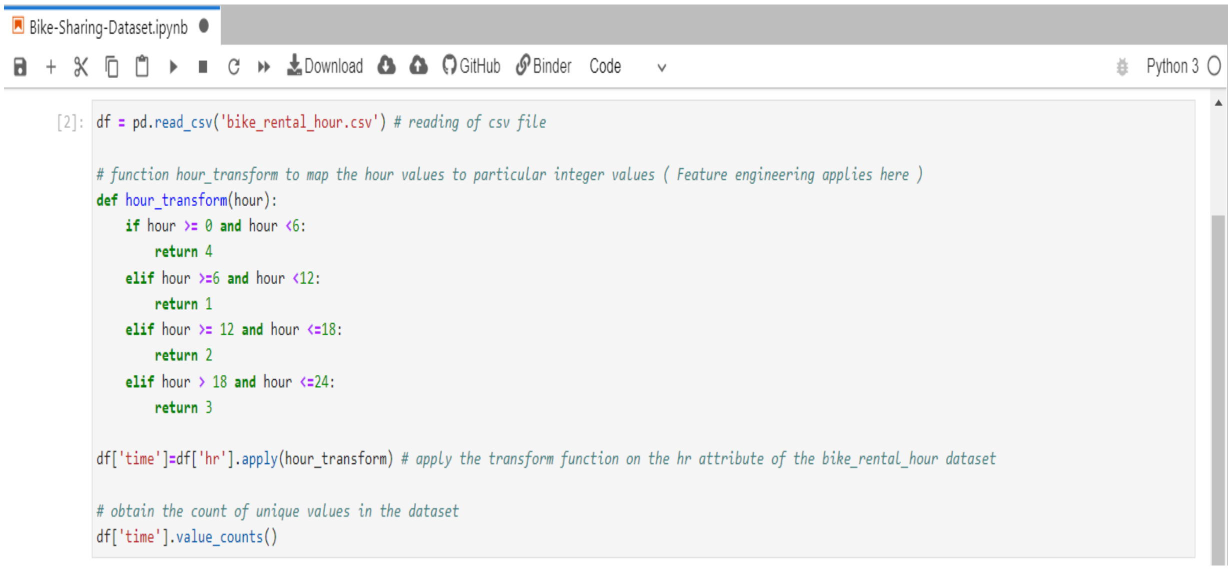
First let us discuss about the essence of feature engineering - that is what is feature engineering and why we require this in our project.

What is Feature Engineering :

If we talk about it in simple terms , it basically refers to the process of using domain knowledge to select and transform the most relevant attributes from raw data. It is used when we create a predictive model (like in our case , prediction of bike rentals) using ML or statistical modelling.
The steps involved in the feature engineering are :

- Data exploration
- Exploratory analysis
- Benchmark

Aim of Feature Engineering : It is done so as to improve the performance of the Machine Learning algorithms. This was the motivation for us in using this.

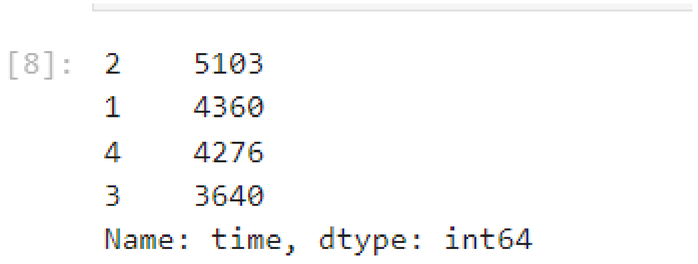


```
[2]: df = pd.read_csv('bike_rental_hour.csv') # reading of csv file

     # function hour_transform to map the hour values to particular integer values ( Feature engineering applies here )
     def hour_transform(hour):
         if hour >= 0 and hour <6:
             return 4
         elif hour >=6 and hour <12:
             return 1
         elif hour >= 12 and hour <=18:
             return 2
         elif hour > 18 and hour <=24:
             return 3

     df['time']=df['hr'].apply(hour_transform) # apply the transform function on the hr attribute of the bike_rental_hour dataset

     # obtain the count of unique values in the dataset
     df['time'].value_counts()
```

After careful analysis of the dataset , we found out that the attribute **hour** of the dataset was not contributing well enough in the efficiency of the ML algorithms which we are going to apply later. So we took the help of feature engineering concept and transformed the information contained in the hour attribute to one of the four labors :

- Morning
- Afternoon
- Evening
- Night

After counting the unique values in the dataset , the results obtained were as follows:



```
[8]: 2    5103
     1    4360
     4    4276
     3    3640
     Name: time, dtype: int64
```

Now since the tabular and graphical analysis of the dataset is done along with Feature engineering , we will move towards the ML algorithms for bike rental predictions.

MODEL ERRORS COMPARISON

Inferences about the dataset :

After the comprehensive tabular and graphical analysis of the dataset , we obtained the following results :

- Our dataset contains **Real / Integer values**
- Some of the attributes are **more significant** as compared to others
- Some of the attributes require **feature engineering** for **efficiency of ML Algorithms used for prediction**.

We have also applied Feature engineering concept in the previous section and transformed the **hour** attribute into one of the four labors of morning, afternoon, evening and night. Since our dataset consists of real/integer values we will go for regression analysis. But we can also use one classification model called the **RandomForestClassification** for prediction of bike rentals. We can go for the random forest classification since it can work on both labelled and numerical dataset.

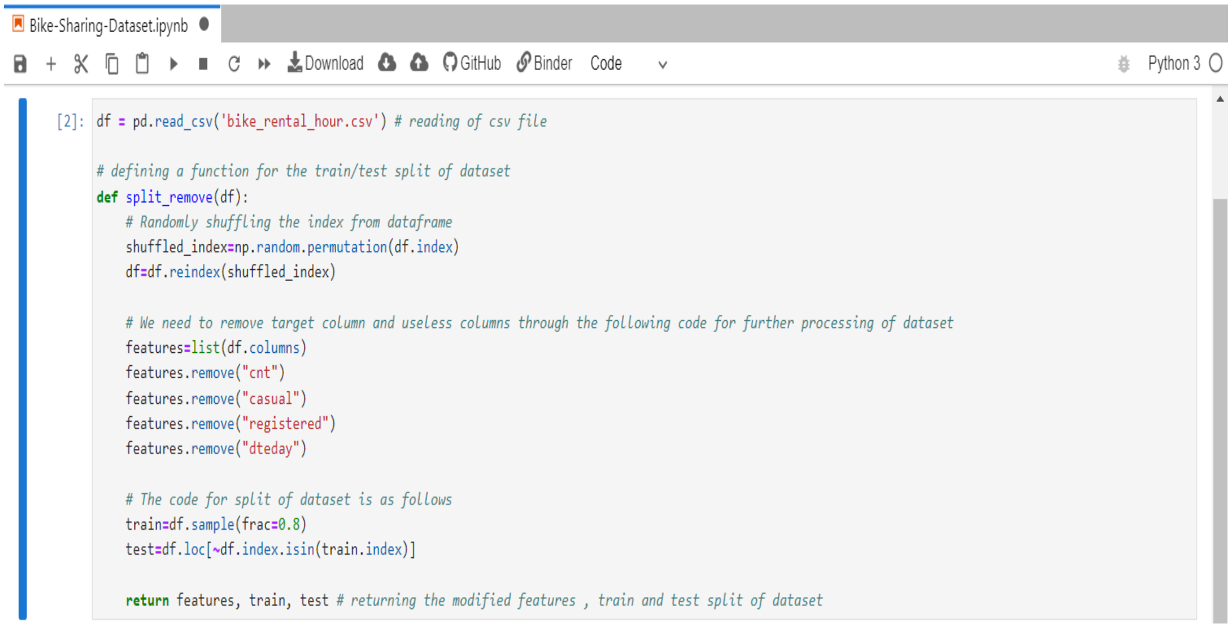
So we will proceed for the regression analysis and classification analysis in the following order :

- Split data into train/test
- Linear regression model on the dataset
- Decision Tree regression model on the dataset
- Random Forest regression model on the dataset
- Random Forest Classification on the dataset

(1) Split data into train/test :

The code for the train/test split of dataset is given below.
Here in this code , first we have read the csv file using read_csv function of pandas library. Now we will create a function called **split_remove** which will perform the required function of splitting the dataset. The steps involved in the process are as follows :

- Randomly shuffle the index from dataframe 'df' using function np.random_permutation
- Removing target column and useless columns from the dataset
- The split dataset code is written
- The new modified features and train/test split dataset is returned from the function.



```
[2]: df = pd.read_csv('bike_rental_hour.csv') # reading of csv file

     # defining a function for the train/test split of dataset
     def split_remove(df):
         # Randomly shuffling the index from dataframe
         shuffled_index=np.random.permutation(df.index)
         df=df.reindex(shuffled_index)

         # We need to remove target column and useless columns through the following code for further processing of dataset
         features=list(df.columns)
         features.remove("cnt")
         features.remove("casual")
         features.remove("registered")
         features.remove("dteday")

         # The code for split of dataset is as follows
         train=df.sample(frac=0.8)
         test=df.loc[~df.index.isin(train.index)]

         return features, train, test # returning the modified features , train and test split of dataset
```

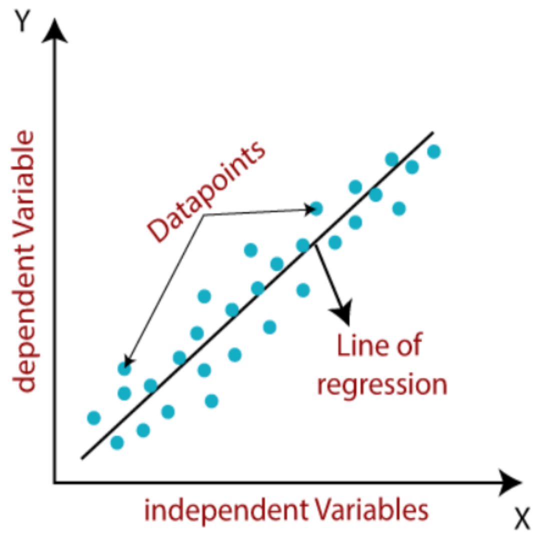
<u>Why we are doing this :</u> We are splitting the dataset into train and test so that we can pass them through different **ML algorithms** which we will proceed for in the further sections. The **training dataset** is given to the algorithm and on the basis of the training dataset , **prediction of the test dataset** takes place.

(2) Linear Regression Model applied on the dataset :

Linear regression in Machine learning domain is one of the most easiest algorithms that one can go for. It is used in the **predictive analysis** of features. Linear regression is used in the predictive analysis of those datasets where the data/information is **continuous/real** or of **numeric type**. Linear regression basically shows a **linear relationship** between a dependent variable (let us say **y**) and one or more independent variables.
The mathematical representation of linear regression is :

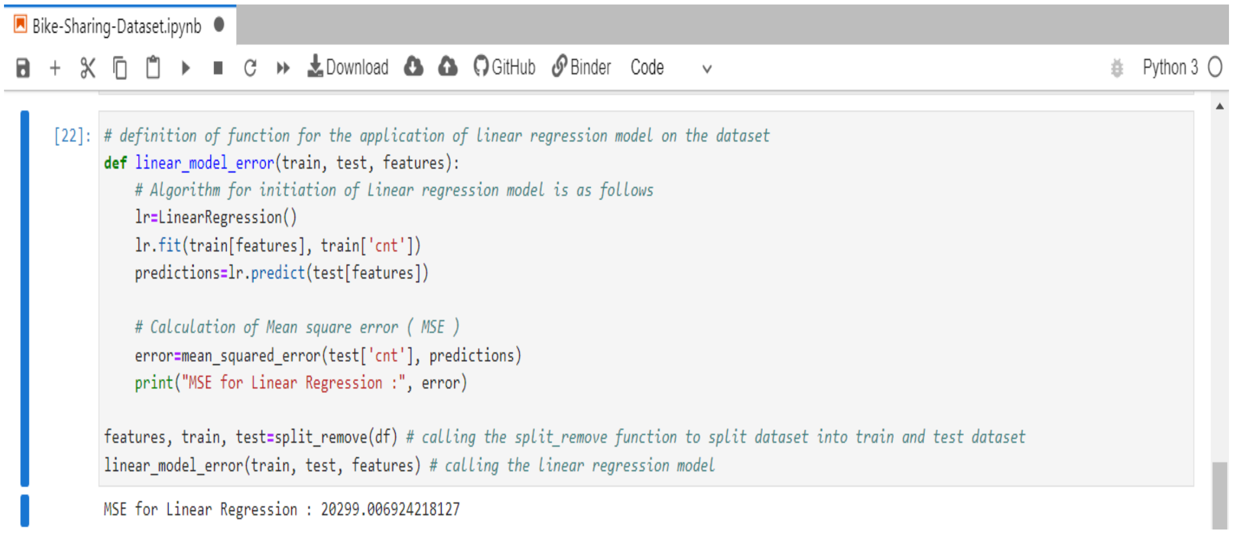
$$y = a_0 + a_1 x + \varepsilon$$

In the above equation , **y** is the dependent variable (target variable) , **x** is the independent variable (predictor variable), $\mathbf{a_0}$ is the intercept of the line (additional degree of freedom) , $\mathbf{a_1}$ is the linear regression coefficient and **ε** is the random error introduced in the model.
The graphical representation of linear regression is as follows :



Application of Linear Regression model on the dataset :

Now let us apply **linear regression model** on the dataset and observe the results. The code and corresponding results are as given below :



```
[22]: # definition of function for the application of linear regression model on the dataset
def linear_model_error(train, test, features):
    # Algorithm for initiation of Linear regression model is as follows
    lr=LinearRegression()
    lr.fit(train[features], train['cnt'])
    predictions=lr.predict(test[features])

    # Calculation of Mean square error ( MSE )
    error=mean_squared_error(test['cnt'], predictions)
    print("MSE for Linear Regression :", error)

features, train, test=split_remove(df) # calling the split_remove function to split dataset into train and test dataset
linear_model_error(train, test, features) # calling the linear regression model

MSE for Linear Regression : 20299.006924218127
```

Inference about the Linear Regression Model :

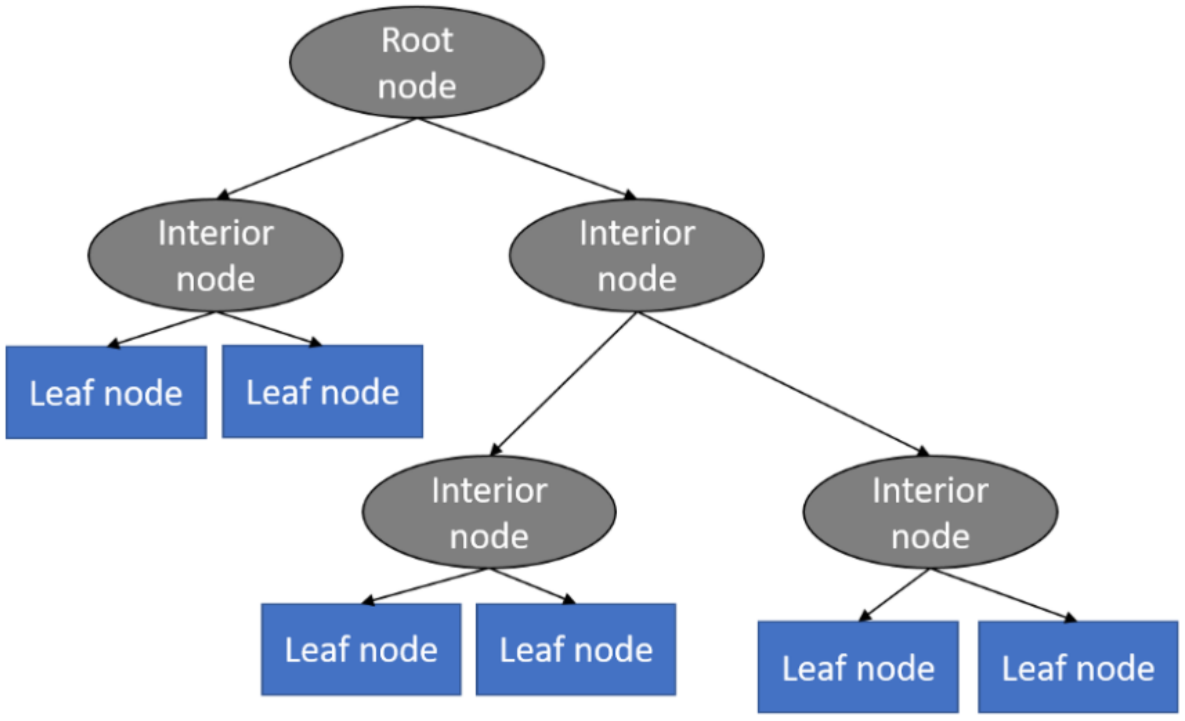
The metric used for the efficiency of ML algorithms is Mean square error (MSE). From the results obtained above , we see that the MSE for linear regression is very much high. The reasons for the same could be that if there is a possibility that if data has a few high rental counts , there are larger errors which are penalized more with MSE. This leads to a higher total error.

Now let us move forward toward calculation of MSE for other regression techniques.

(3) Decision Tree Regression Model applied on the dataset :

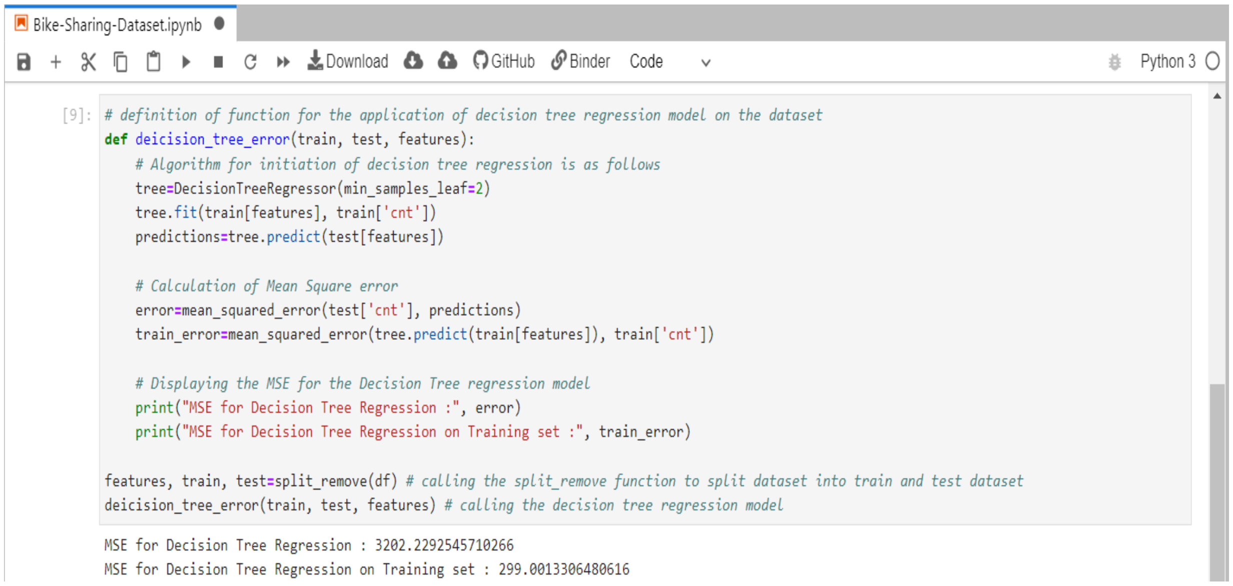
Decision Tree is also the most commonly used approached that are used for **supervised learning**. One of the benefits of Decision Tree is that it can solve both **regression** and **classification** tasks.
The structure of the decision tree is such that it has three types of nodes.

- Root Node : It is the starting/initial node of the tree. It represents the entire sample and it further gets split into child nodes (interior nodes).
- Interior Nodes : The interior nodes represent the features of the dataset and its branches represent the decision rules that need to be taken.
- Leaf Nodes : It is used to represent the outcome of the Regression/Classification model.



Application of Decision Tree Regression model on the dataset :

Now let us apply **decision tree regression model** on the dataset and observe the results. The code and corresponding results are as given below :



```
[9]: # definition of function for the application of decision tree regression model on the dataset
def deicision_tree_error(train, test, features):
    # Algorithm for initiation of decision tree regression is as follows
    tree=DecisionTreeRegressor(min_samples_leaf=2)
    tree.fit(train[features], train['cnt'])
    predictions=tree.predict(test[features])

    # Calculation of Mean Square error
    error=mean_squared_error(test['cnt'], predictions)
    train_error=mean_squared_error(tree.predict(train[features]), train['cnt'])

    # Displaying the MSE for the Decision Tree regression model
    print("MSE for Decision Tree Regression :", error)
    print("MSE for Decision Tree Regression on Training set :", train_error)

features, train, test=split_remove(df) # calling the split_remove function to split dataset into train and test dataset
deicision_tree_error(train, test, features) # calling the decision tree regression model
```

```
MSE for Decision Tree Regression : 3202.2292545710266
MSE for Decision Tree Regression on Training set : 299.0013306480616
```

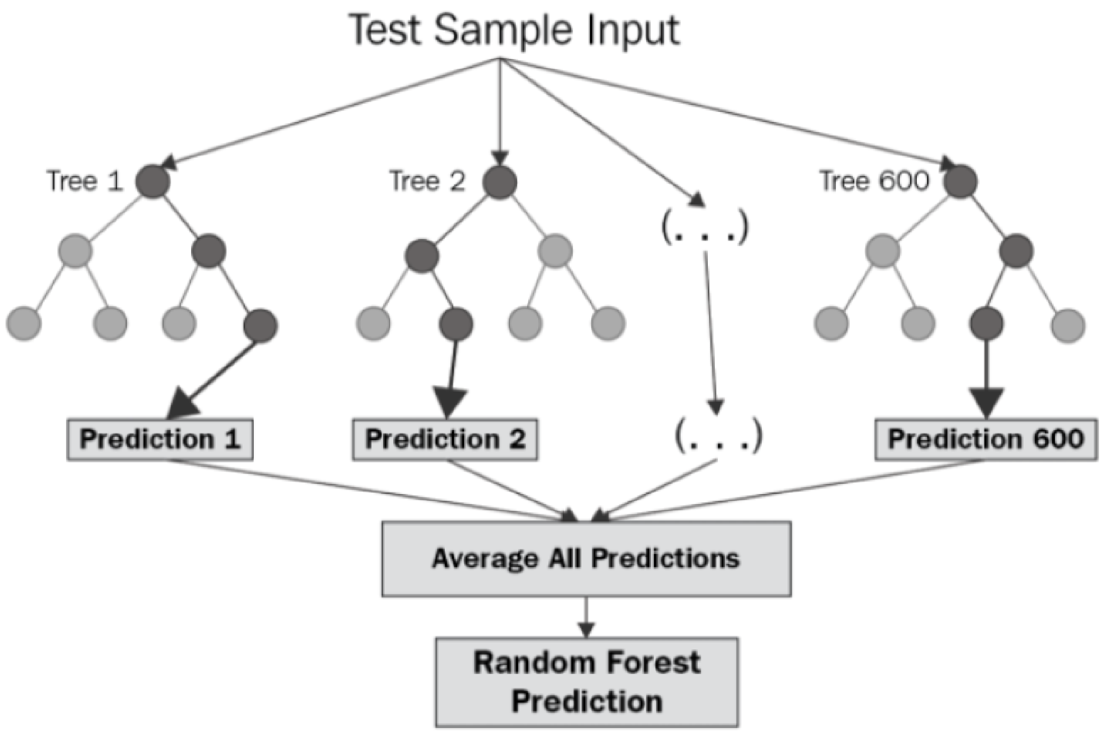
Inference about the Decision Tree Regression Model :

The metric used for the efficiency of decision tree model is Mean Square error. After carefully analyzing the results of regression model , we came to the conclusion that decision tree **performs much better** than the linear regression model. The only catch is that since there is a huge difference between the MSE for the training dataset and MSE for the test dataset, one of the issue it is pointing to is the **overfitting issues.**

Now let us move forward towards calculation of MSE for the final model that is Random Forest Regression model. After this we will go for the Random Forest Classification model analysis of dataset.

(4) Random Forest Regression Model applied on the dataset :

The graphical representation of the **random forest regression** model is as follows :

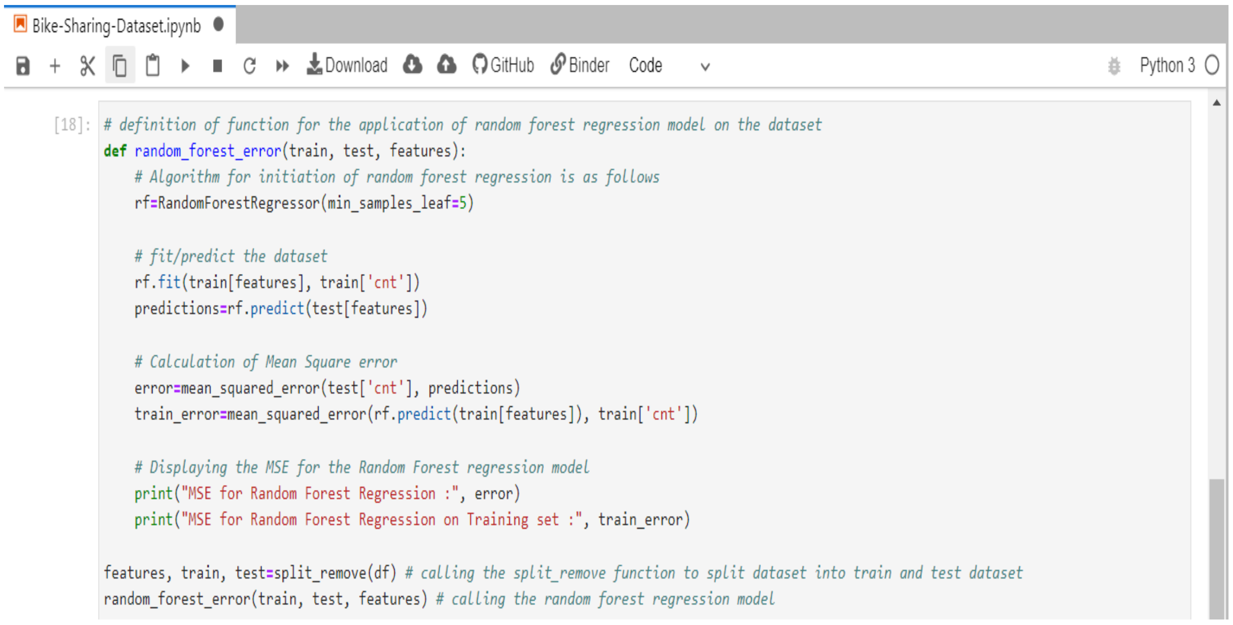


Random Forest Regression model is one of the **supervised learning** technique. It uses the concept of **ensemble learning method** for regression. Now let us discuss something about the ensemble learning method :

Ensemble learning method : In this learning method , there are multiple machine learning algorithms. Predictions from these different machine learning algorithms are combined together to make a final prediction about the dataset which is much more accurate than prediction made by a single model.

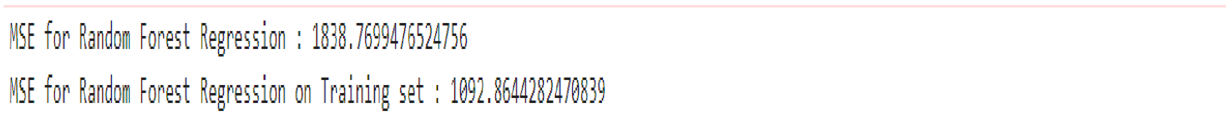
Application of Random Forest Regression model on the dataset :

Now let us apply **random forest regression model** on the dataset and observe the results. The code for the random forest regression model on the dataset is given below :



```python
# definition of function for the application of random forest regression model on the dataset
def random_forest_error(train, test, features):
    # Algorithm for initiation of random forest regression is as follows
    rf=RandomForestRegressor(min_samples_leaf=5)

    # fit/predict the dataset
    rf.fit(train[features], train['cnt'])
    predictions=rf.predict(test[features])

    # Calculation of Mean Square error
    error=mean_squared_error(test['cnt'], predictions)
    train_error=mean_squared_error(rf.predict(train[features]), train['cnt'])

    # Displaying the MSE for the Random Forest regression model
    print("MSE for Random Forest Regression :", error)
    print("MSE for Random Forest Regression on Training set :", train_error)

features, train, test=split_remove(df) # calling the split_remove function to split dataset into train and test dataset
random_forest_error(train, test, features) # calling the random forest regression model
```

The results obtained for the MSE of random forest regression model is as follows :



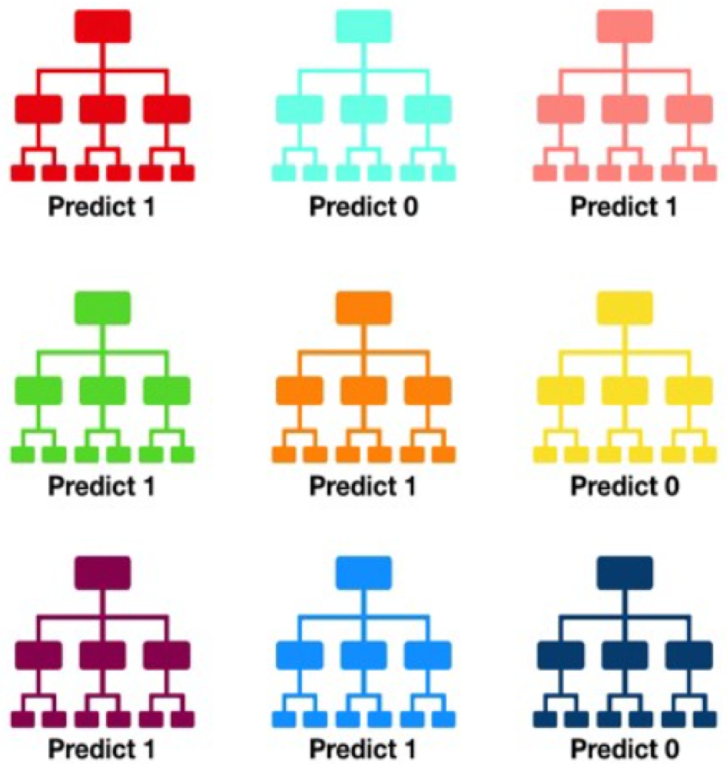
```
MSE for Random Forest Regression : 1838.7699476524756
MSE for Random Forest Regression on Training set : 1092.8644282470839
```

Inference about the Random Forest Regression Model :

After carefully analyzing the results of the Random Forest Regression model , we came to the conclusion that Random Forest Regression model outperforms the previous two regression models and it gives a more balance mean square error (MSE) on both the training and test data sets.

(5) Random Forest Classification model applied on the dataset :

Apart from the regression techniques that we discussed above , we can also apply random forest classification on the dataset. Some information about the classification technique and how to apply it on the **Bike-sharing-dataset** is described below.



Random Forest Classifier consist of a **large number of decision trees** that we can use as an ensemble. We can see this as each individual tree in the random forest gives a class prediction and on the basis of prediction with higher number of votes, that class becomes the overall model's prediction.

How to apply random forest classification to the bike sharing dataset : we can go for classification on the dataset if we have **binned dataset** available.

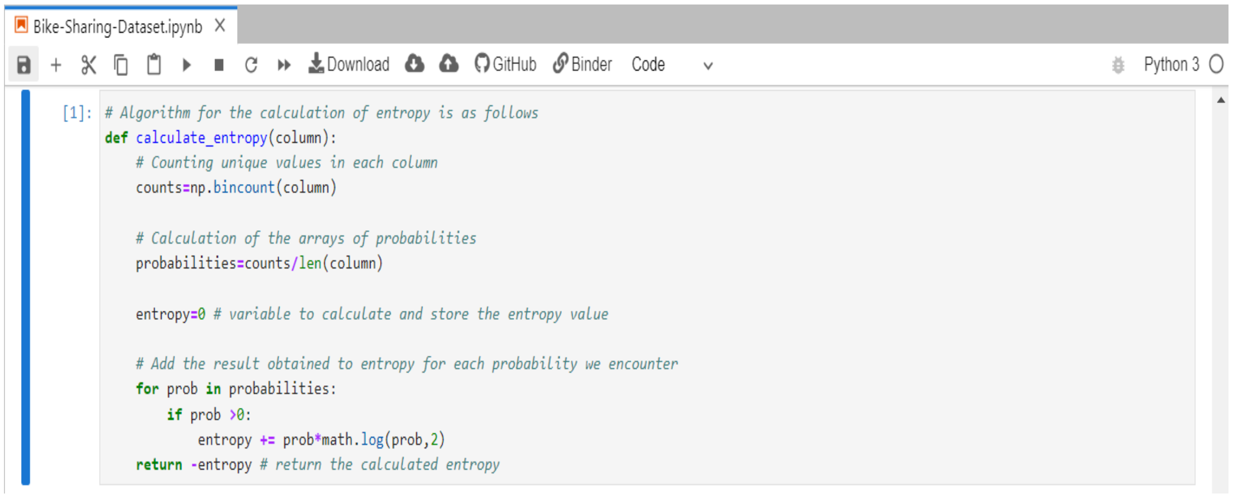
MOST SIGNIFICANT ATTRIBUTE

Now after analysis of dataset (both tabular and graphical) and prediction of target feature (bike rentals) is done let us explore further into analysis of most significant attribute that affects the dataset and whose contribution is most in the Mean square error of the ML algorithms discussed above.

Let us go step by step in exploration of most significant attribute :

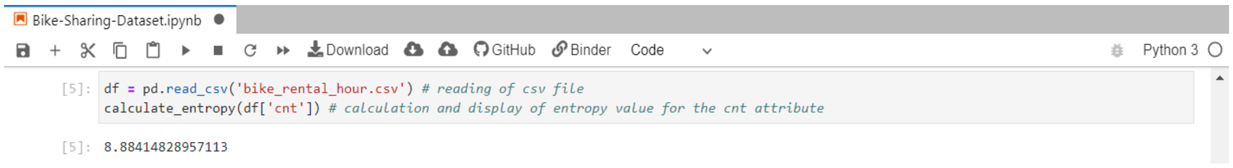
(1) Calculation of entropy :

The function code for the calculation of entropy is as follows :



```
[1]: # Algorithm for the calculation of entropy is as follows
def calculate_entropy(column):
    # Counting unique values in each column
    counts=np.bincount(column)

    # Calculation of the arrays of probabilities
    probabilities=counts/len(column)

    entropy=0 # variable to calculate and store the entropy value

    # Add the result obtained to entropy for each probability we encounter
    for prob in probabilities:
        if prob >0:
            entropy += prob*math.log(prob,2)
    return -entropy # return the calculated entropy
```

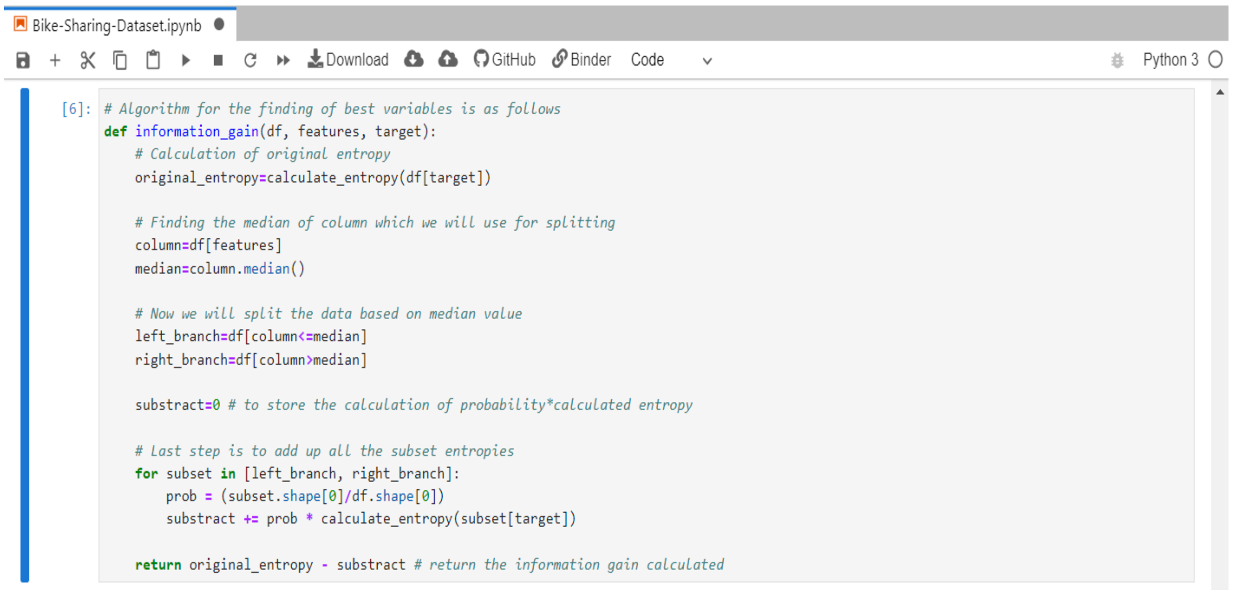
Now read the csv file using pandas library and apply the entropy function on the **cnt** column of the dataset.



```
[5]: df = pd.read_csv('bike_rental_hour.csv') # reading of csv file
calculate_entropy(df['cnt']) # calculation and display of entropy value for the cnt attribute

[5]: 8.88414828957113
```

The entropy value calculated for the **cnt** columns comes out to be **8.884148**
The next step in the calculation of most significant attribute is to **find the best variables** from the dataset.

<u>(2) Finding the best variables from the dataset :</u>



```
# Algorithm for the finding of best variables is as follows
def information_gain(df, features, target):
    # Calculation of original entropy
    original_entropy=calculate_entropy(df[target])

    # Finding the median of column which we will use for splitting
    column=df[features]
    median=column.median()

    # Now we will split the data based on median value
    left_branch=df[column<=median]
    right_branch=df[column>median]

    substract=0 # to store the calculation of probability*calculated entropy

    # Last step is to add up all the subset entropies
    for subset in [left_branch, right_branch]:
        prob = (subset.shape[0]/df.shape[0])
        substract += prob * calculate_entropy(subset[target])

    return original_entropy - substract # return the information gain calculated
```

Through the above algorithm , we have obtained the best variables from the dataset. The steps involved in doing so are as follows :

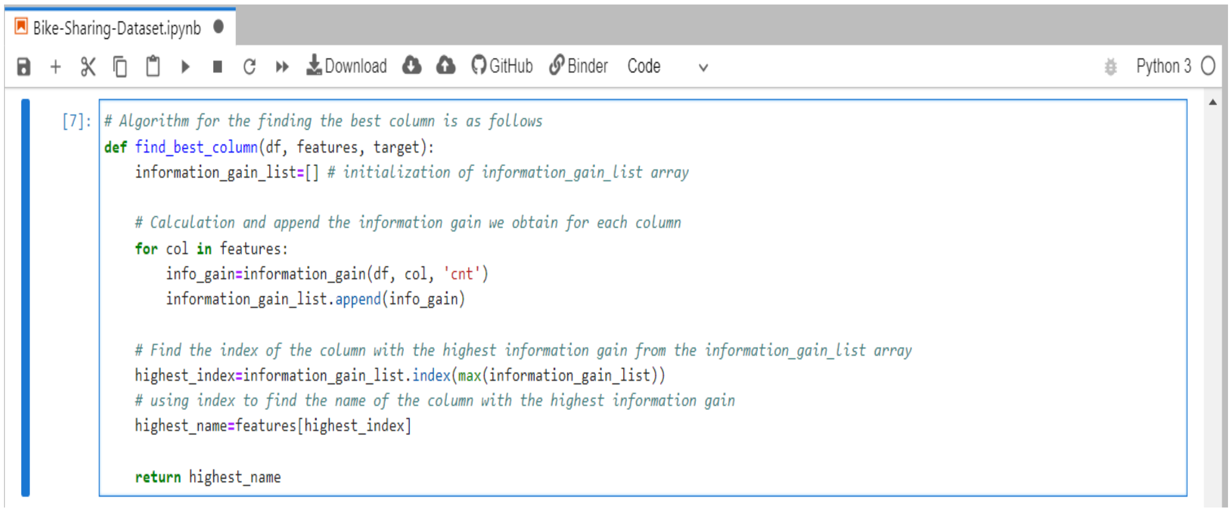
- Calculation of original entropy value
- Finding the median of the column of dataset which is used for splitting the data
- Splitting the data on the basis of median value calculated in the previous step
- Add up all the subset entropies
- Return the **information gain** calculated with the help of original entropy and subtracted entropy.

Now the last step in the analysis of most significant attribute is finding the best column from the dataset.

(3) Finding the best column from the dataset :

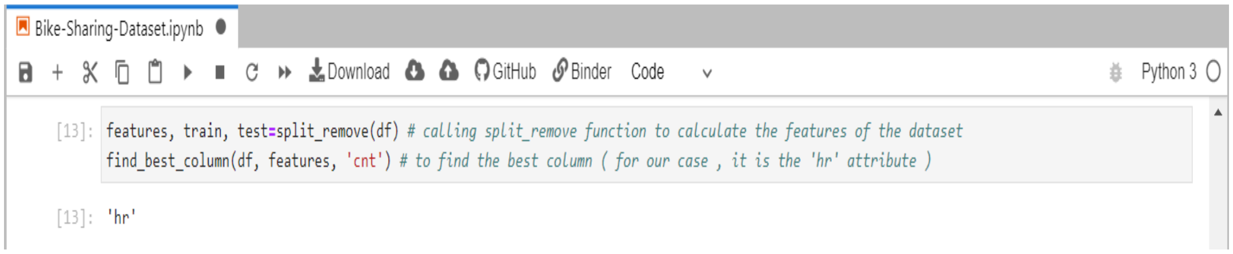
This is the last step in calculation of **most significant attribute**. The code and corresponding results are as follows :

The function code for the calculation of best column is as follows :



```
[7]: # Algorithm for the finding the best column is as follows
     def find_best_column(df, features, target):
         information_gain_list=[] # initialization of information_gain_list array

         # Calculation and append the information gain we obtain for each column
         for col in features:
             info_gain=information_gain(df, col, 'cnt')
             information_gain_list.append(info_gain)

         # Find the index of the column with the highest information gain from the information_gain_list array
         highest_index=information_gain_list.index(max(information_gain_list))
         # using index to find the name of the column with the highest information gain
         highest_name=features[highest_index]

         return highest_name
```

Now we will make use of the above function to derive at the final result.



```
[13]: features, train, test=split_remove(df) # calling split_remove function to calculate the features of the dataset
      find_best_column(df, features, 'cnt') # to find the best column ( for our case , it is the 'hr' attribute )

[13]: 'hr'
```

<u>**Most significant attribute**</u> **:** From the results obtained above , we come to the conclusion that **hr** column is the best column of the dataset and in fact the **"most significant attribute"** of the dataset which influences the properties of the dataset. Now let we sum up the results obtained for the prediction of bike rentals and most significant attribute which we have discussed in the previous 2 sections in the subsequent conclusion section.

CONCLUSION

Let us first summarize the results of the regression analysis done on the dataset for prediction of bike rentals into a table :

Regression Models Used	**Mean Square Error (MSE)**
Linear Regression Model	20299.00692
Decision Tree Regression Model	3202.22925
Random Forest Regression Model	1838.76994

Looking at the above table we come to the following conclusions :

- **Linear Regression** performs worst since its MSE is very high. The reasons for the *higher MSE* could be the case when the dataset contains few high rental counts leading to a massive penalty by the MSE.
- **Decision Tree Regression** works better than Linear Regression but due to huge differences in MSE for the training and test datasets, it seems to be *overfitting.*
- **Random Forest Regression** performed much better than the previous two regression models and it also was *less overfitting.*

We also performed the most significant attribute analysis on the dataset and came to the conclusion that : **hr** attribute of the dataset is the **most significant variable** for bike rental predictions and the reason for the same could be that it generates **highest amount of impurity** in the ML algorithms.

REFERENCES

Notes and Presentation slides of IDS Course 2021-22

Bike Sharing Dataset - December 20 , 2013
[http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset]

Introduction to Bike Sharing Systems - December 5 , 2021
[https://en.wikipedia.org/wiki/Bicycle-sharing_system]

About pandas python library
[https://pandas.pydata.org/]

About scikit-learn machine learning in python
[https://scikit-learn.org/stable/]

From Correlation to Causation - Sanjeev Suresh , May 11 2020
[https://towardsdatascience.com/from-correlation-to-causation-49f566eea954]

About Heat Map - September 1 , 2021
[https://en.wikipedia.org/wiki/Heat_map]

About Scatter Plot - W3 schools
[https://www.w3schools.com/python/python_ml_scatterplot.asp]

About Histogram
[https://en.wikipedia.org/wiki/Histogram]

About Feature Engineering - omni sci
[https://www.omnisci.com/technical-glossary/feature-engineering]

Linear Regression in Machine Learning - javatpoint
[https://www.javatpoint.com/linear-regression-in-machine-learning]

Decision Tree Regression in Machine Learning - Gurucharan MK , July 15 2020
[https://towardsdatascience.com/machine-learning-basics-decision-tree-regression-1d73ea003fda]

Random Forest Regression in Machine Learning - Chaya Bakshi , June 9 2020
[https://levelup.gitconnected.com/random-forest-regression-209c0f354c84]

Random Forest Classification techniques - Tony Yiu , June 12 2019
[https://towardsdatascience.com/understanding-random-forest-58381e0602d2#:~:text=The%20random%20forest%20is%20a,that%20of%20any%20individual%20tree.]

About Ensemble learning method - Dr. Robi Polikar , December 22 2008
[http://www.scholarpedia.org/article/Ensemble_learning]