Project:
BIKE RENTAL COUNT

OCTOBER - 10, 2019

EDWISOR

Submitted By: SANJAY TALLOLLI

Pre requisite:

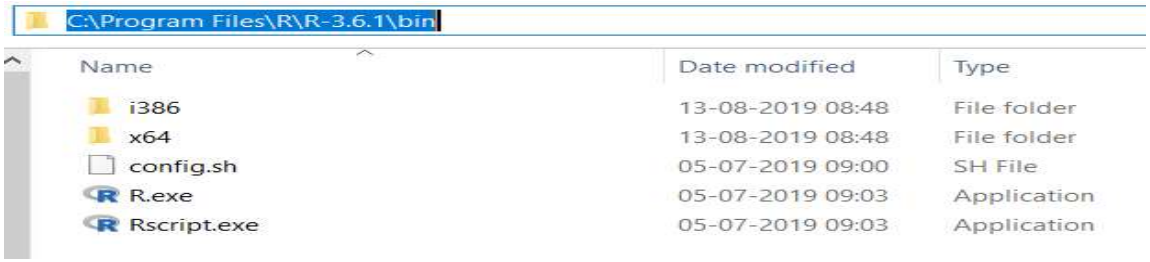
A. To run the R project in the system should have R &R-studio; Below are the versions of both which are used:
   1. R version 3.6.1 (64 bit).
   2. R-studio IDE (Version 1.2.1335).

B. To run the Python (ipynb) file in the system should have Anaconda Navigator to be installed Below are the versions of software used:
   1. Anaconda Navigator (Version -1.9.7)
   2. Jupyter Notebook (Version – 6.0.0)
   3. Python Version – 3.6.5

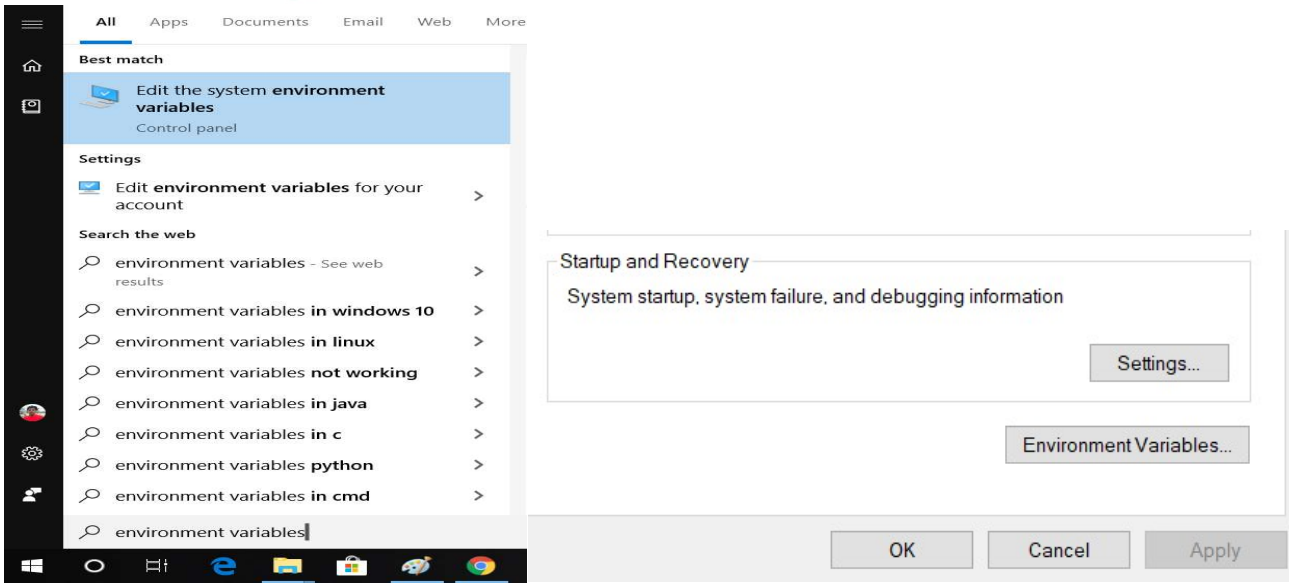
C. Download the file and save it in your local. (In my case):
   D:/Edwisor2019/Projects/Bike Rental Count/day.csv
   D:/Edwisor2019/Projects/ Bike Rental Count / Bike_Rental_Count_R.r (R-Script File)
   D:/Edwisor2019/Projects/ Bike Rental Count / Bike_Rental_Count _Python.py
   D:/Edwisor2019/Projects/ Bike Rental Count / Bike_Rental_Count _ Python _ deploy.py (Python-Script Files)
   Bike_Rental_Count_R. ipynb (R – Notebook)
   Bike_Rental_Count_Python.ipynb (Python – Notebook)
   Bike_Rental_Count_Python_deploy.ipynb (Python – Notebook)
   Bike Rental Count Project Report.docx or .pdf format

Steps to execute the R-code:

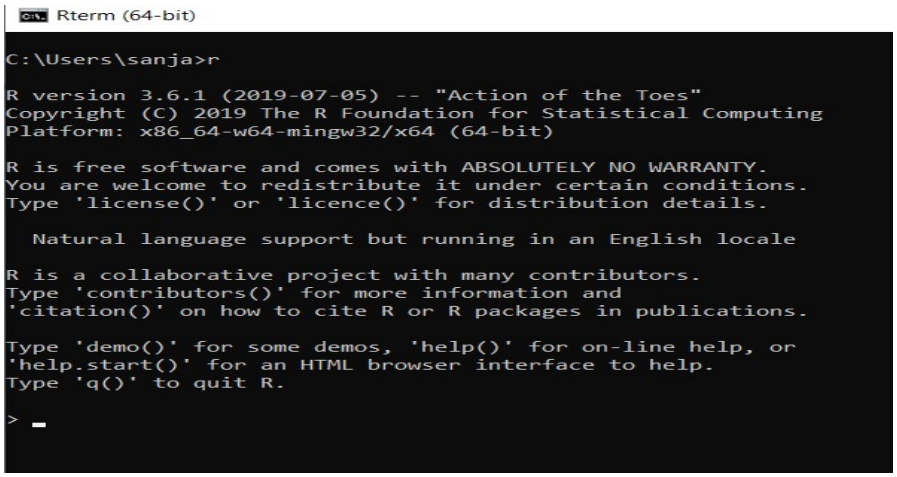
i. Run the application by following the below steps using Command Prompt:
   - Option – 1: Set Path: Check path of your R base software viz., R.exe and Rscript.exe (In my case: C:\Program Files\R\R-3.6.1\bin) copy the path



- Press windows key, search for environment variables



- Select environment variables under Advanced tab,
- under system variables → select path→ then edit→ paste the copied path then click ok three times.
- To check working of R from command prompt, press windows key search for cmd then enter, type r or R in prompt then enter you will get R command prompt as shown below.



- Go to the path where you have saved R code files along with all input files if any, select the path of the file at top of the window then type "cmd" in that search bar and press enter.
- You will be directed to command prompt with the location from where you executed the command.
- Next, Type the R-script followed by filename with extension (.R) and press enter. Code will run and print's the output.
- Option – 2: In R-studio - Open R file (Go to New-> select open file → select the r code file.
- To run the R-script, Ctrl+Alr+R to run the entire r script OR if you want to run line by line, select the line you want to execute then Ctrl+Enter to get the output in console and plots in Plot window. Alternatively, at top of the window ribbons you will see Run button click the same.

Steps to execute the Python code

1) For Python please open Anaconda prompt type Jupyter notebook and load ipynb file and run the code
2) .py file also attached please download the same run in cmd suggested version of python is 3.6 because kNN fancy impute supported only in python 3.6.
3) To run from command prompt:
   - Open Command line: Start menu -> Run and type cmd
   - Type: C:\python36\python.exe D:\Edwisor2019\Projects\Bike Rental Count\Bike_Rental_Count_Python.py
   - Or if your system is configured correctly, you can drag and drop your script from Explorer onto the Command Line window and press enter.
4) For Complete project report Open pdf document named "Bike Rental Count Project Report. Pdf"

1. Introduction

1.1 Problem Statement:

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

In the bike rental dataset, we need to predict what could be the counting of bike rental for a given season, month and year with environmental conditions like windspeed, temperature, humidity etc., based on the historical data for the two years i.e. 2011 and 2012.

1.2 Data

Understanding of data is the very first and important step in the process of finding solution of any business problem. Let's have a quick preview of the input data Will discuss in detail about the data understanding in the CRISP-DM process section.

Let's understand shape of training and test dataset:

```
Shape of training data is:  (731, 16)
```

Preview of historical data for two years:

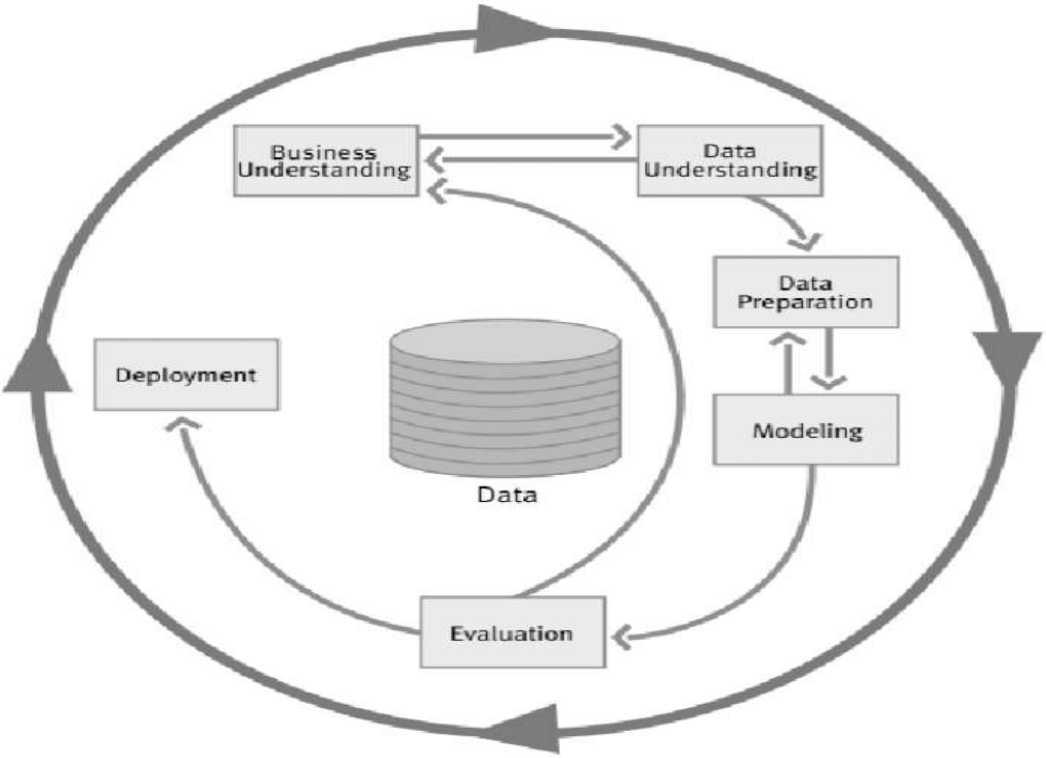
	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364	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

2 METHODOLOGY:

2.1 CRISP-DM Process:

CRISP-DM stands for cross-industry process for data mining. The CRISP-DM methodology provides a structured approach to planning a data mining works. The project follows CRISP DM process to develop the model for the given problem. It involves the following major steps:

1. Business understanding.
2. Data understanding.
3. Data preparation.
4. Modeling.
5. Evaluation.
6. Deployment.



2.2 Business Understanding:

Set objectives - This primary objective is what we want to accomplish from a business perspective which uncover important factors that could influence the outcome of the project, that would be "How to accurately predict the Bike rental count for a particular environment condition? What are the influential factors which affects that !?"

Produce project plan – The plan should specify the steps to be performed during the rest of the project, including the initial selection of tools and techniques i.e., in our case the plan will be involved Understanding the available historic data and clean the data by converting into a proper shape, i.e., a data free from anomalies like missing values, outliers that can possibly produce errors; scaling the variables; applying various stat or ML models and choosing the best model for a Bike Rental Count Prediction.

Business success criteria – Here we'll lay out the criteria that we'll use to determine whether the project has been successful from the business point of view ideally be specific and measurable. In our case the success would be related to able to accurately predict the bike rental count for a particular environment condition as input, with maximum accuracy.

2.3 Data Understanding:

We must predict bike rental count based on environmental conditions and the historical dataset of two years has dependent & independent variables, since our target variable is a continuous variable therefore this problem comes under supervised machine learning Regression problem. There are total 731 observations, 16 variables in historical data.

Data Dictionary: - The details of data attributes in the dataset are as follows let's understand each attribute in detail.

instant: Record index

dteday: Date in yyyy-mm-dd format

season: Season (1: springer, 2: summer, 3: fall, 4: winter)

yr: Year (0: 2011, 1:2012)

mnth: Month (1 to 12) mean 1: January, 2: February…….12: December

hr: Hour (0 to 23)

holiday: weather day is holiday or not (extracted from Holiday Schedule)

weekday: Day of the week i.e., 0: Sunday, 1: Monday.....6: Saturday

workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted from Freemeteo)

1: Clear, Few clouds, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds

4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)

atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt-t_min), t_min=-16, t_max=+50 (only in hourly scale)

hum: Normalized humidity. The values are divided to 100 (max)

windspeed: Normalized wind speed. The values are divided to 67 (max)

casual: count of casual users

registered: count of registered users

cnt: count of total rental bikes including both casual and registered

Preview of historical data for two years:

	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

Here our response / target variable is (cnt) and predictor/independent variables are (instant), (dteday), (season), (yr), (mnth), (holiday), (workingday), (weekday), (weathersit), (temp), (atemp), (hum), (windspeed), (casual), (registered).

```
Shape of training data is:  (731, 16)
```

2.4 Data Pre - Processing:

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.

Data preprocessing is defining the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model. As we already know the quality of our inputs decide the quality of our output. So, once we have got our business hypothesis ready, it makes sense to spend lot of time and efforts here. Approximately, data exploration, cleaning and preparation can take up to 60-70% of our total project time. This process is often called as Exploratory Data Analysis

Listed below are data pre-processing techniques used for this Project

1) Data Exploration and Cleaning
2) Missing Value Analysis
3) Outlier analysis
4) Analyzing Distribution of each variable with respect to target variable using Visualization
5) Feature Selection
6) Feature Scaling

Let's discuss /explain this technique in detail.

2.4.1 Data Exploration and Cleaning :

Data Pre-processing is the very first step which comes with any data science project is data exploration and cleaning, according to our bike rental count prediction project below are the points which are identified in this stage:

we first imported the data into R and Python environments, explored the data by looking its dimensions, data structures, variable names, summary of data to have a glance at our data we checked first and last rows of the train dataset .renamed variables for more understanding of the data, Identified the variables for Bike Rent count prediction like what are the independent variables and target variable so our target variable is cnt (count) and predictors as mentioned below.

```
731   16

'data.frame'

'instant'  'dteday'  'season'  'yr'  'mnth'  'holiday'  'weekday'  'workingday'  'weathersit'  'temp'  'atemp'  'hum'  'windspeed'  'casual'  'registered'
'cnt'

'data.frame':   731 obs. of  16 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ weekday   : int  6 0 1 2 3 4 5 6 0 1 ...
 $ workingday: int  0 0 1 1 1 1 1 0 0 1 ...
 $ weathersit: int  2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.1604460	331	654	985
2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.2485390	131	670	801
3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.2483090	120	1229	1349
4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.1602960	108	1454	1562
5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.1869000	82	1518	1600
6	2011-01-06	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606

- . As we know that some negative values in fare amount, so we must remove or impute those values.

2.4.2 Missing Value Analysis

In real world missing value is the common occurrence of incomplete observations in an asset of data. Missing values can arise due to many reasons, error in uploading data,

unable to measure an observation etc. Due to presence of missing values in the form of 0, NA or NAN. These will affect the accuracy of model which we are building. Hence it is necessary to check for any missing values in the given data.

Let's check of the dataset for missing values and identify what type are they?
In the given data there is no any missing values. So, we need not impute any missing values.

In R:

Missing Value anlysis

```
In [82]: # Check missing values in dataset
         sum(is.na(bike_data))   # there is no missing values present in this dataset
```

```
0
```

In Python:

Missing Value anlysis

```
In [268]: # checking for missing values in dataset
          bike_data.isnull().sum()
```

```
Out[268]: instant        0
          dteday         0
          season         0
          year           0
          month          0
          holiday        0
          weekday        0
          workingday     0
          weather        0
          temperature    0
          atemp          0
          humidity       0
          windspeed      0
          casual         0
          registered     0
          count          0
          dtype: int64
```

2.4.3 Outlier Analysis

An Outlier is any inconsistent or abnormal observation in a variable of dataset that deviates from the rest of the observations. These inconsistent observations can be due to manual error, poor quality/ malfunctioning equipment's, Experimental error or correct but exceptional data based on business use case. It can cause an error in predicting the target variable/s. Hence, we need to check for the outliers and either remove the observations containing them or replace them with NA, then impute or set upper limits/lower limits or mean/median values imputation.

For Outlier analysis, Boxplot to visualize and summary descriptive statistics to check range of each numeric variable and sorted the variables to detect some strange values like zeros and negative values.

From below mentioned image for R & python descriptive summary statistics shows some anomalies in data which are nothing but outliers let's discuss about each variable From the boxplot almost all the variables except "windspeed" and "humidity" does not have outliers.

In Python:

	temperature	atemp	humidity	windspeed	count
count	731.000000	731.000000	731.000000	731.000000	731.000000
mean	0.495385	0.474354	0.627894	0.190486	4504.348837
std	0.183051	0.162961	0.142429	0.077498	1937.211452
min	0.059130	0.079070	0.000000	0.022392	22.000000
25%	0.337083	0.337842	0.520000	0.134950	3152.000000
50%	0.498333	0.486733	0.626667	0.180975	4548.000000
75%	0.655417	0.608602	0.730209	0.233214	5956.000000
max	0.861667	0.840896	0.972500	0.507463	8714.000000

```
# unique values of categories variables
bike_data[cat_cnames].nunique()
```

```
season        4
year          2
month        12
holiday       2
weekday       7
workingday    2
weather       3
dtype: int64
```

Shape of dataset:

```
Shape of training data is:  (731, 16)
```

In R:

```
    instant              dteday        season           yr        
 Min.   :  1.0   2011-01-01:  1   Min.   :1.000   Min.   :0.0000  
 1st Qu.:183.5   2011-01-02:  1   1st Qu.:2.000   1st Qu.:0.0000  
 Median :366.0   2011-01-03:  1   Median :3.000   Median :1.0000  
 Mean   :366.0   2011-01-04:  1   Mean   :2.497   Mean   :0.5007  
 3rd Qu.:548.5   2011-01-05:  1   3rd Qu.:3.000   3rd Qu.:1.0000  
 Max.   :731.0   2011-01-06:  1   Max.   :4.000   Max.   :1.0000  
                 (Other)   :725                                   
      mnth          holiday           weekday        workingday    
 Min.   : 1.00   Min.   :0.00000   Min.   :0.000   Min.   :0.000  
 1st Qu.: 4.00   1st Qu.:0.00000   1st Qu.:1.000   1st Qu.:0.000  
 Median : 7.00   Median :0.00000   Median :3.000   Median :1.000  
 Mean   : 6.52   Mean   :0.02873   Mean   :2.997   Mean   :0.684  
 3rd Qu.:10.00   3rd Qu.:0.00000   3rd Qu.:5.000   3rd Qu.:1.000  
 Max.   :12.00   Max.   :1.00000   Max.   :6.000   Max.   :1.000  

   weathersit         temp             atemp              hum        
 Min.   :1.000   Min.   :0.05913   Min.   :0.07907   Min.   :0.0000  
 1st Qu.:1.000   1st Qu.:0.33708   1st Qu.:0.33784   1st Qu.:0.5200  
 Median :1.000   Median :0.49833   Median :0.48673   Median :0.6267  
 Mean   :1.395   Mean   :0.49538   Mean   :0.47435   Mean   :0.6279  
 3rd Qu.:2.000   3rd Qu.:0.65542   3rd Qu.:0.60860   3rd Qu.:0.7302  
 Max.   :3.000   Max.   :0.86167   Max.   :0.84090   Max.   :0.9725  

   windspeed          casual        registered       cnt        
 Min.   :0.02239   Min.   :   2.0   Min.   :  20   Min.   :  22  
 1st Qu.:0.13495   1st Qu.: 315.5   1st Qu.:2497   1st Qu.:3152  
 Median :0.18097   Median : 713.0   Median :3662   Median :4548  
 Mean   :0.19049   Mean   : 848.2   Mean   :3656   Mean   :4504  
 3rd Qu.:0.23321   3rd Qu.:1096.0   3rd Qu.:4776   3rd Qu.:5956  
 Max.   :0.50746   Max.   :3410.0   Max.   :6946   Max.   :8714  
```

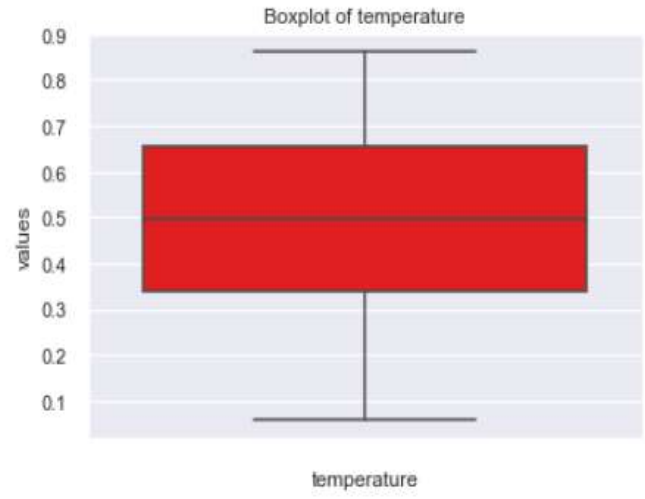
Shape of dataset

In [75]:
```
#Check the dimensions(no of rows and no of columns)
print(paste(" Shape of training data is :   No. of Rows =",nrow(bike_data)," And No. of Columns =", ncol(bike_data)," "))
```

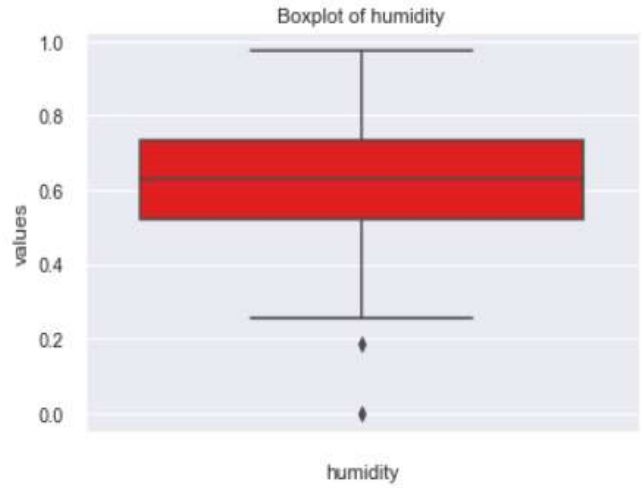
```
[1] " Shape of training data is :   No. of Rows = 731  And No. of Columns = 16  "
```

In python: Box plots for both numeric and categorical variables:

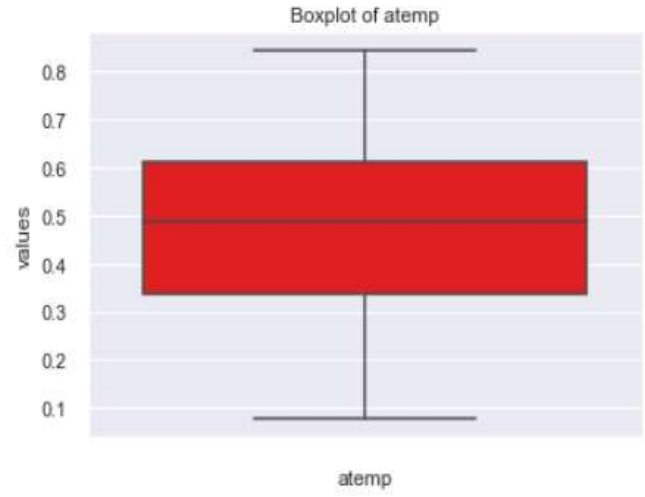
temperature



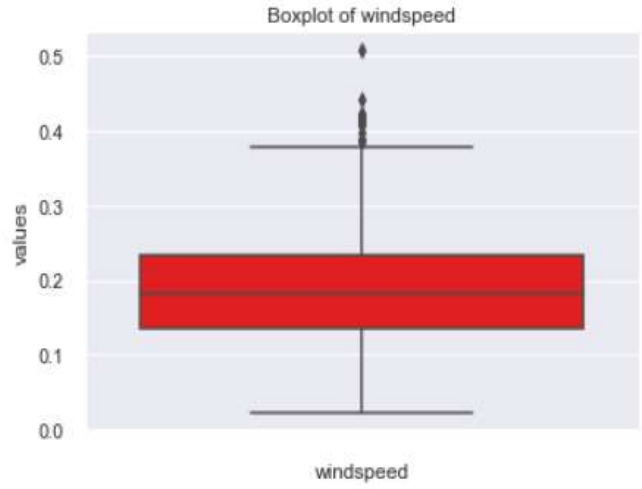
humidity



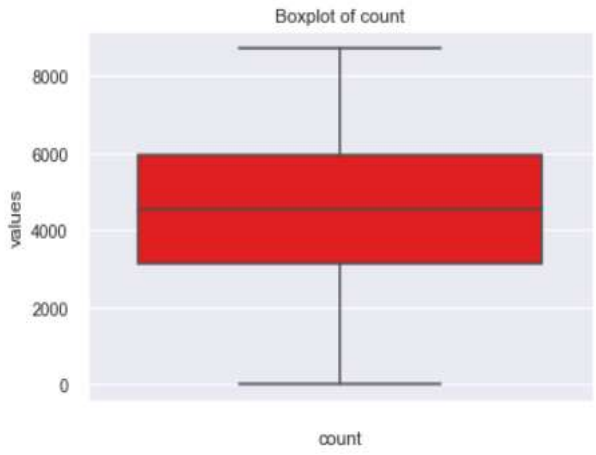
atemp



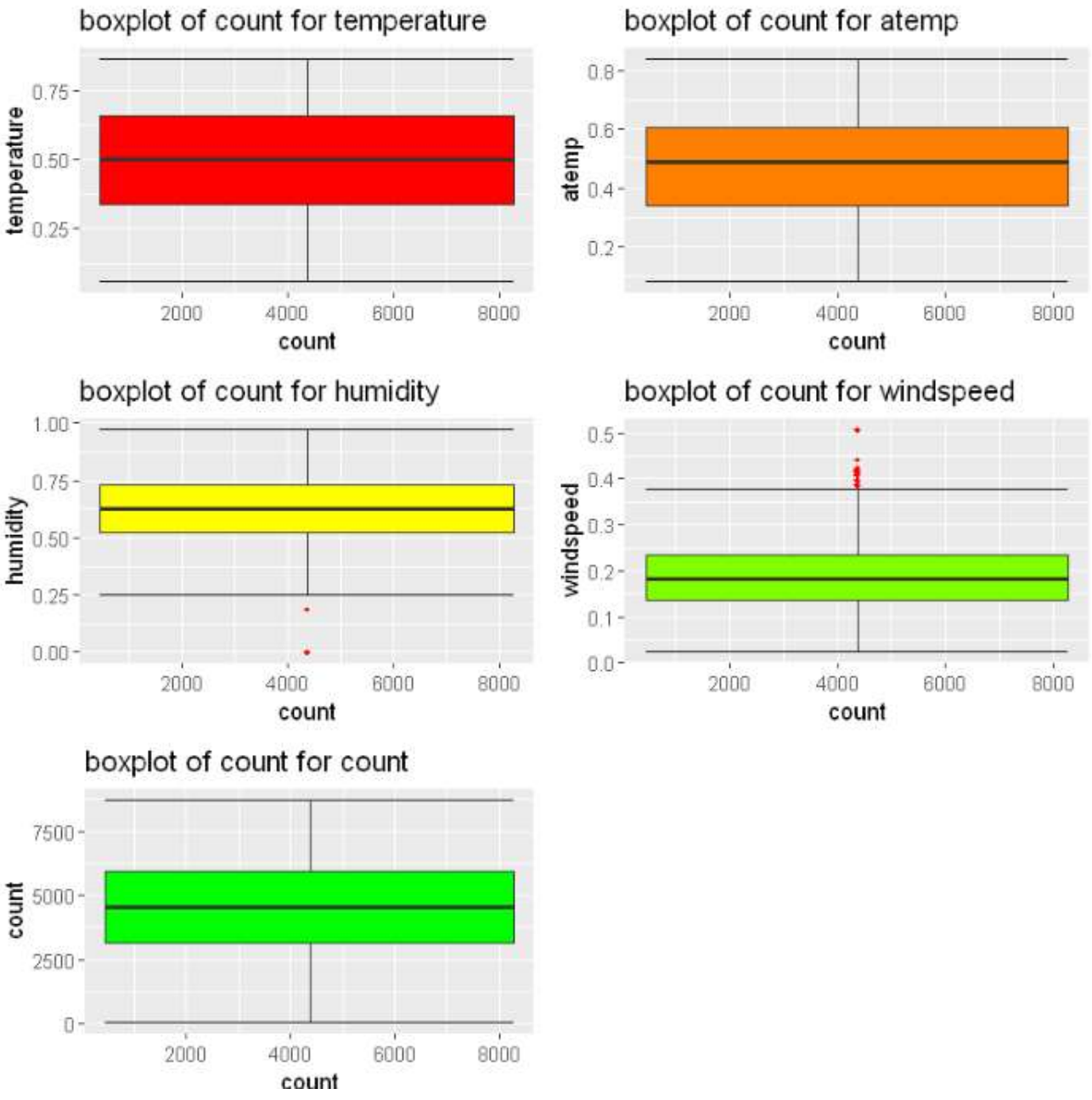
windspeed



count



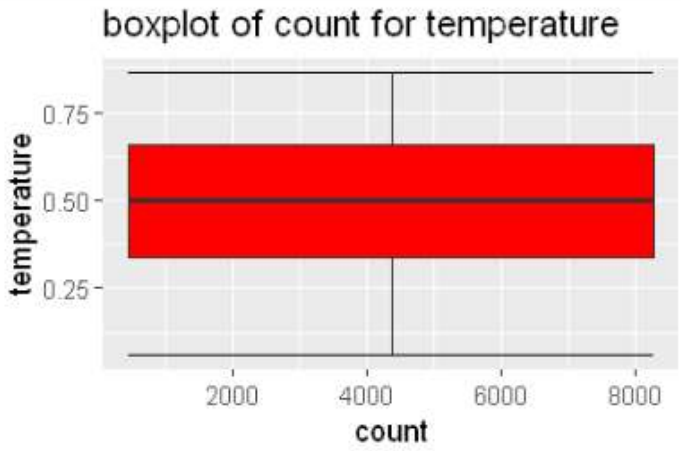
In R: Box plots for both numeric and categorical variables:

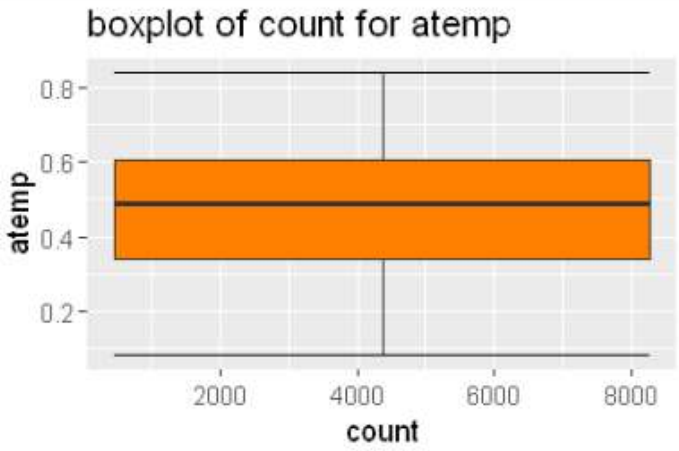


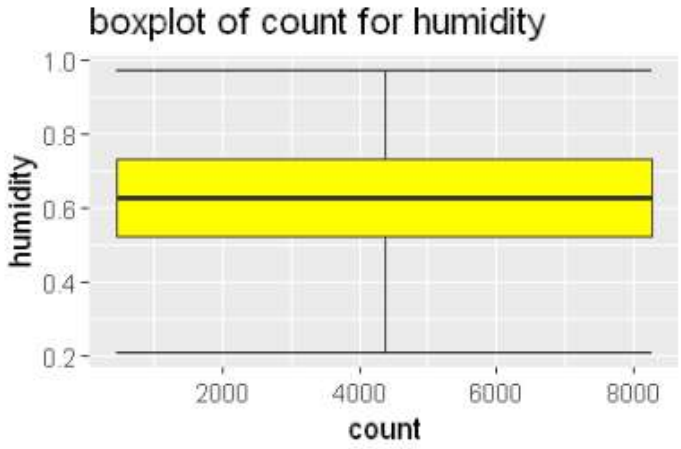
For our project we opted for capping method in which we are going to impute outlier with upper fence and lower fence value reason behind to opt this method is we don't want to delete the observations with outliers as data collection is also a crucial step in data analytics for which client has to spend more money if don't have any past data specially for start-up companies .by keeping this point into consideration we tried to

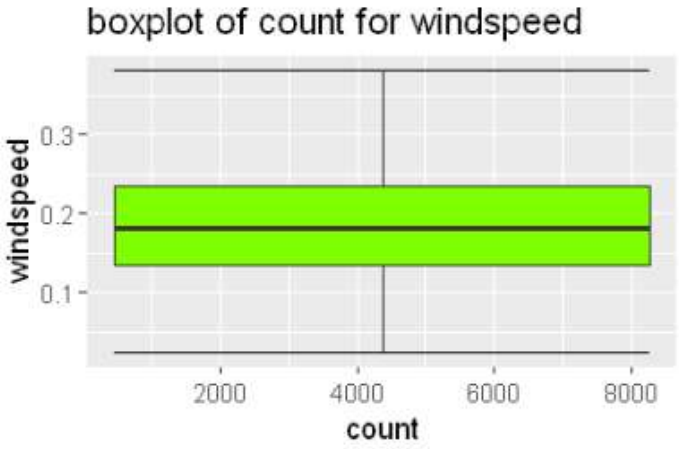
retain the data wherever possible in the pre-processing. Boxplot stat method is one of the methods to remove outlier and we also learnt we can also treat these outliers as missing values and impute them with mean or median or knn method or delete such observations.

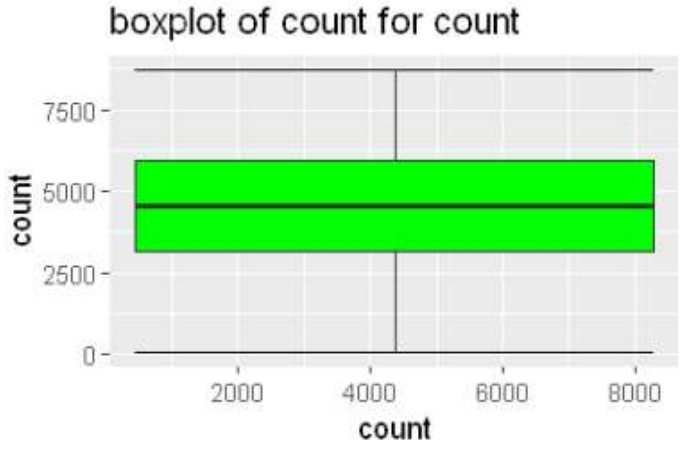
In R:- Box Plot after removing outliers





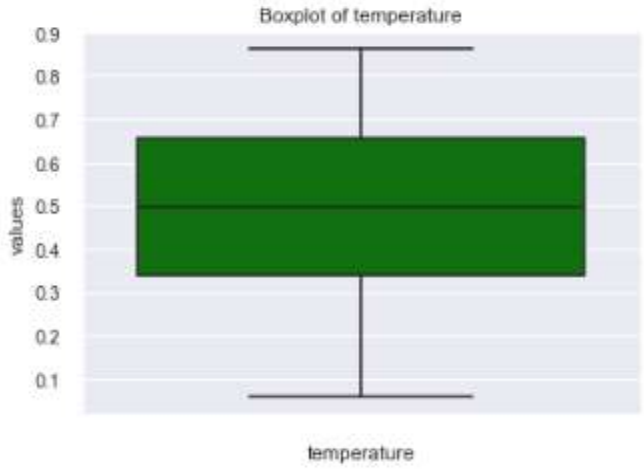




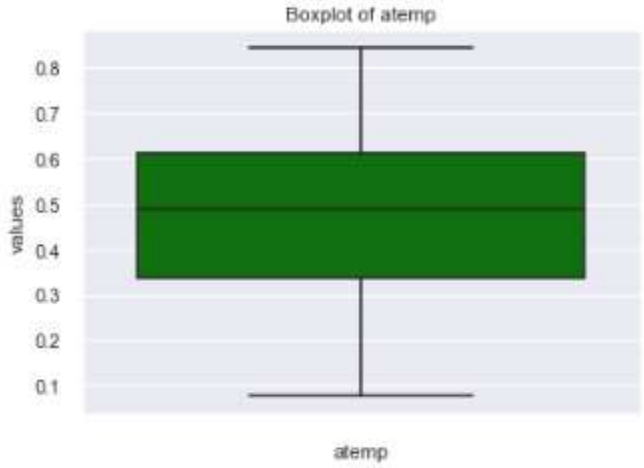


In Python:- Box Plot after removing outliers

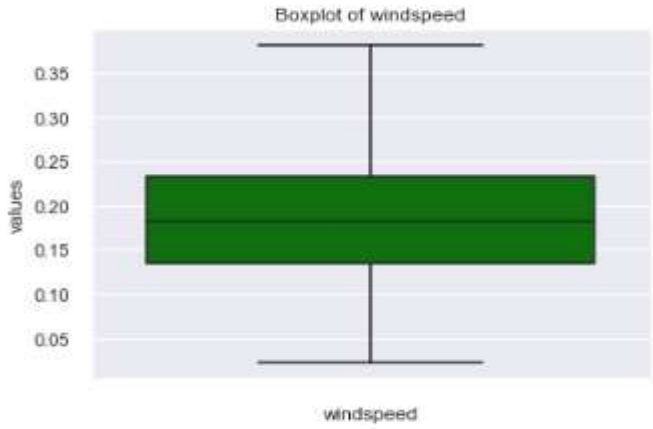
temperature



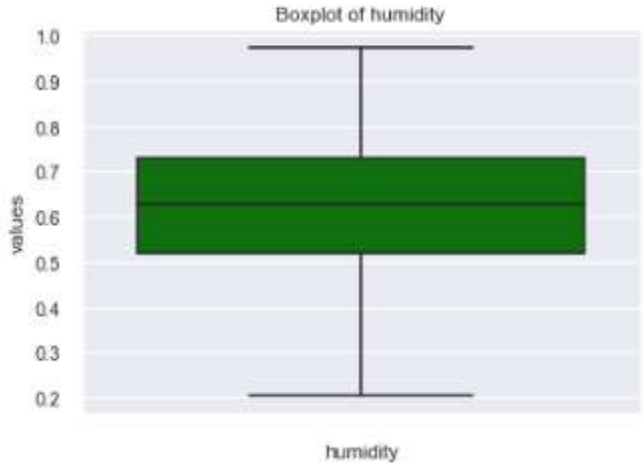
atemp



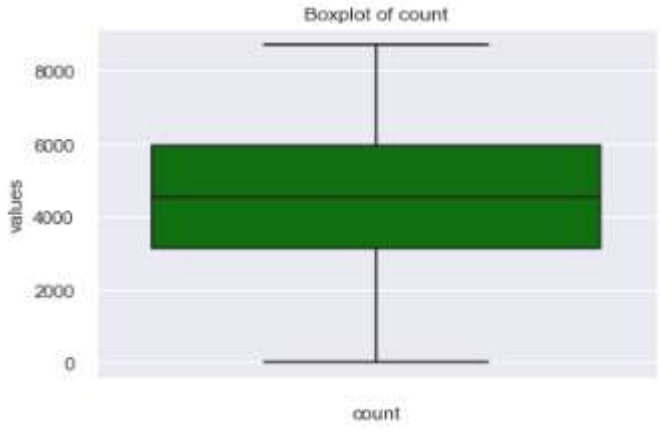
windspeed



humidity



count



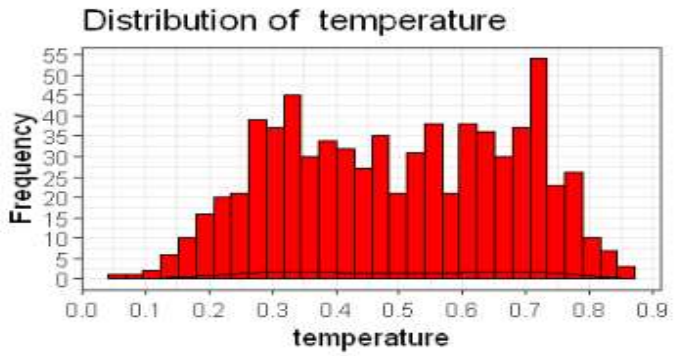
2.4.4 Feature Engineering :

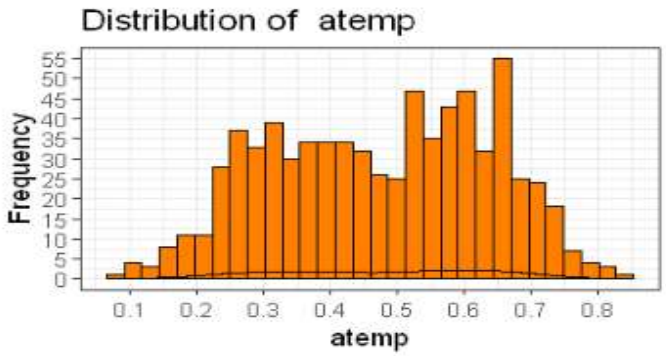
Feature engineering is the science (and art) of extracting more information from existing data, not adding any new data to it, but making the data more meaningful and usable. In our Project, No need to add any new/feature variable to our data.

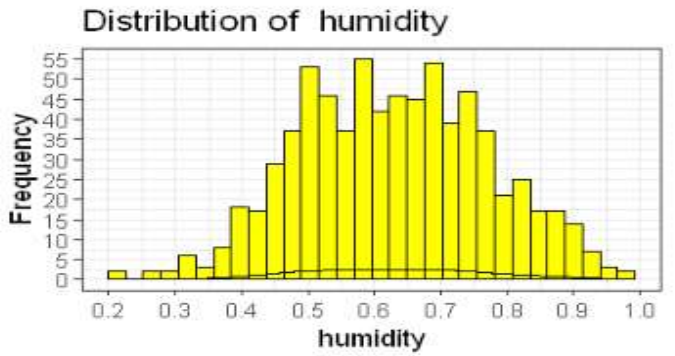
2.4.5 Frequency Distribution Visualization:

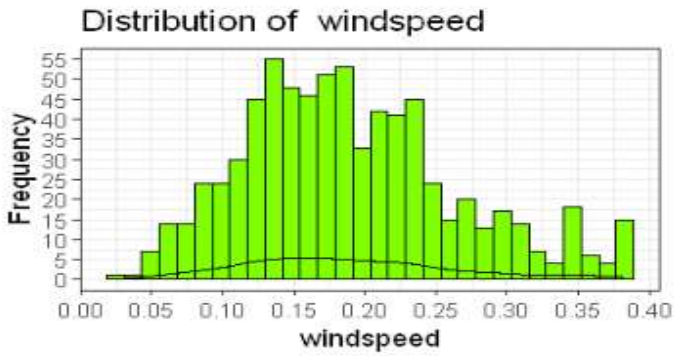
To check distribution of each continuous variable we plotted histogram for each variable Both in R and Python, we can also check distribution using summary or describe function. In our project it can be observed that from the below plot we can say predictor and target variables are normally distributed.

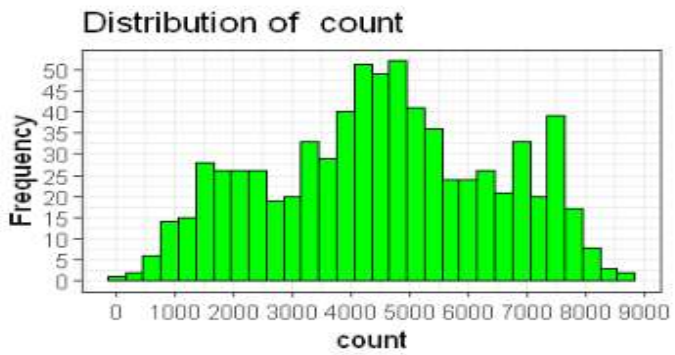
In R:



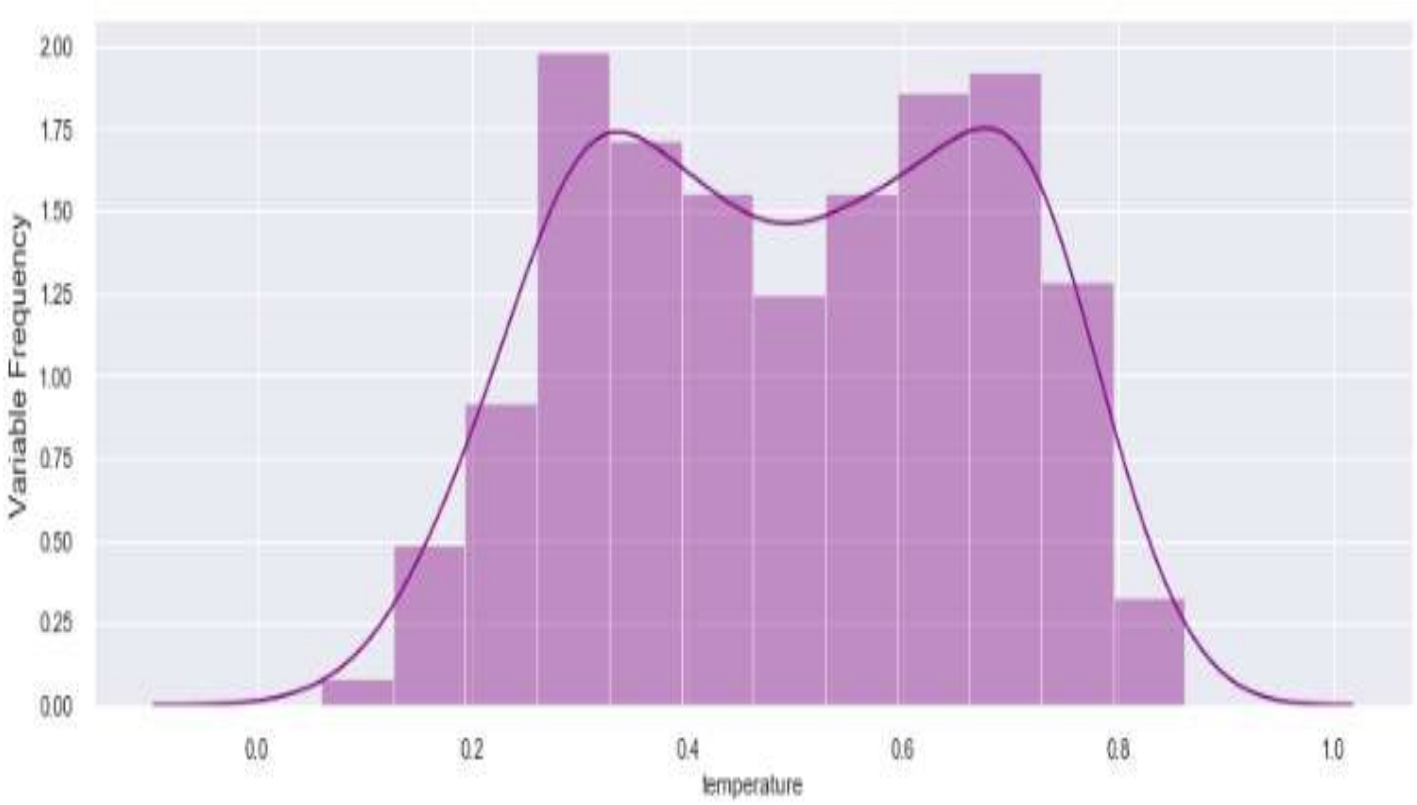


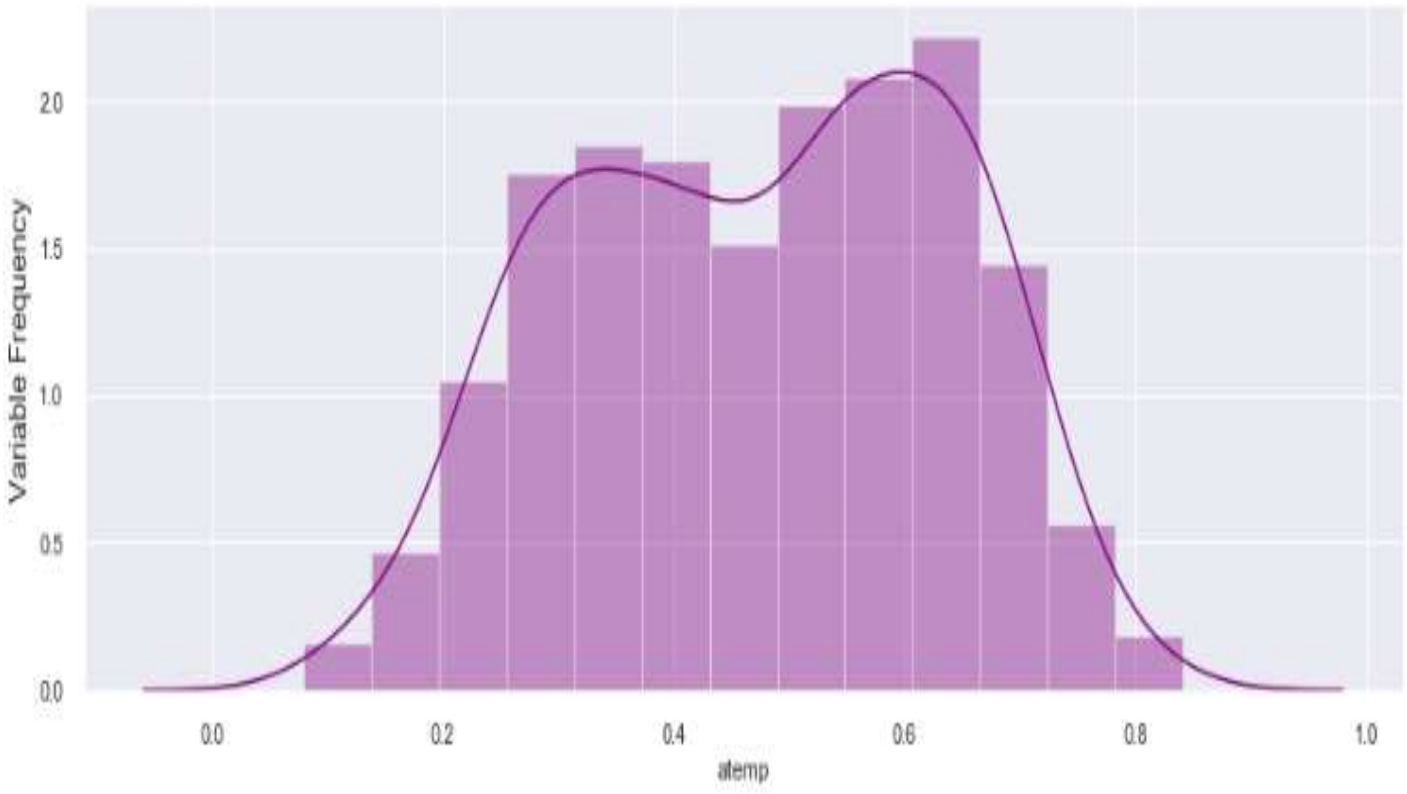


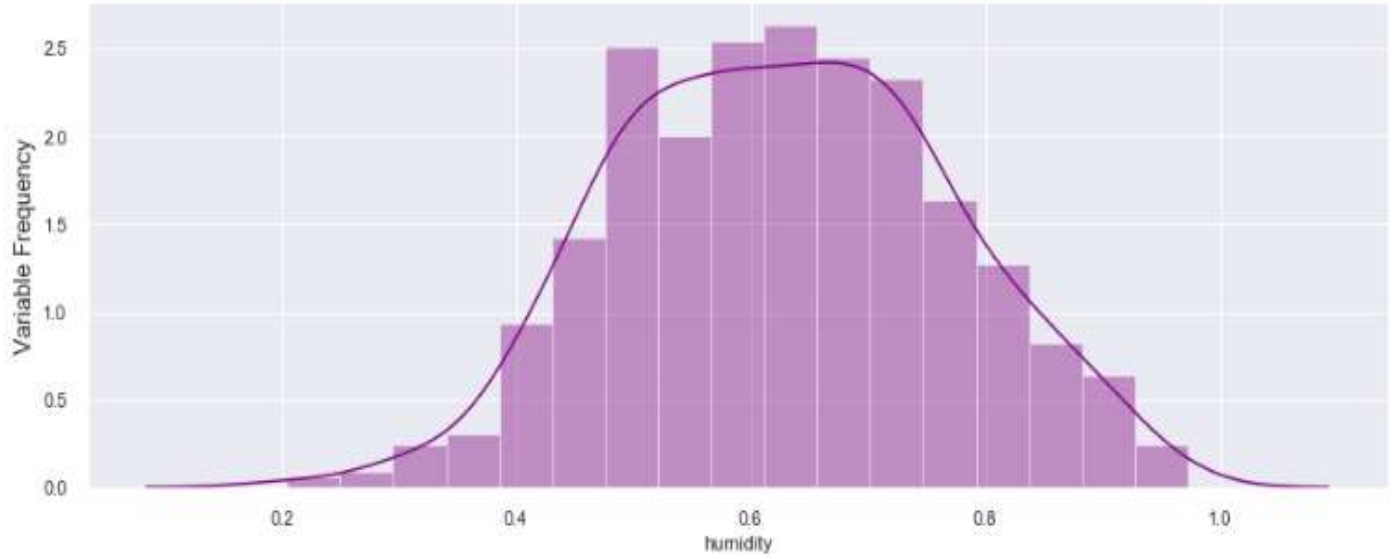


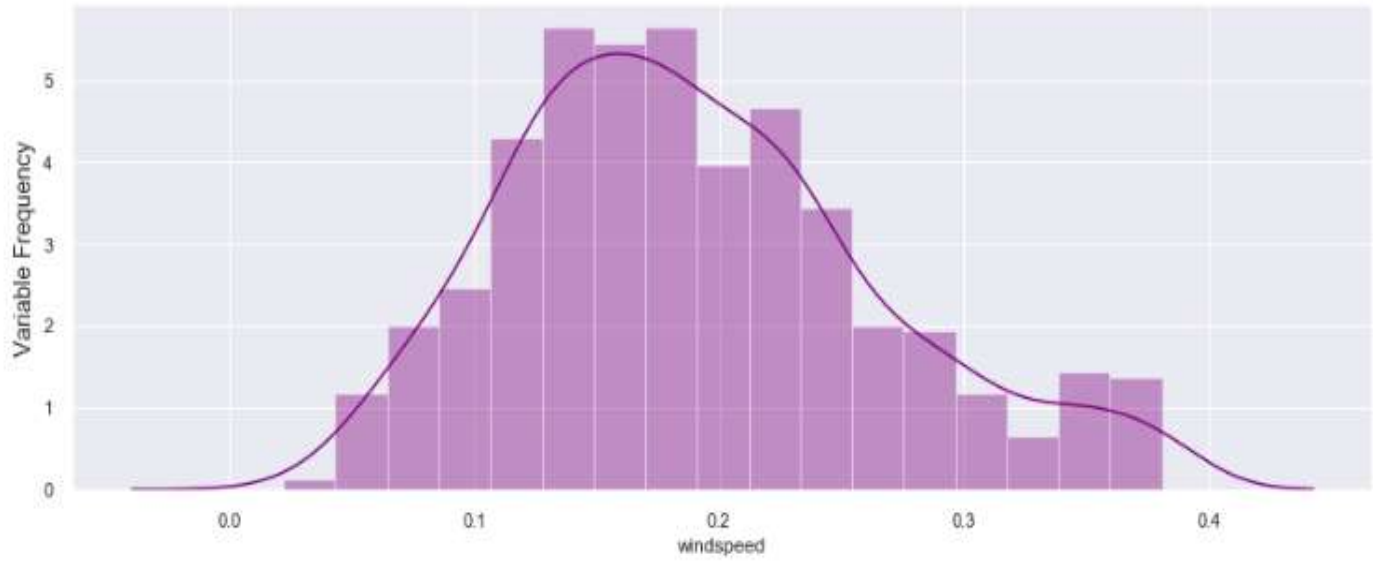


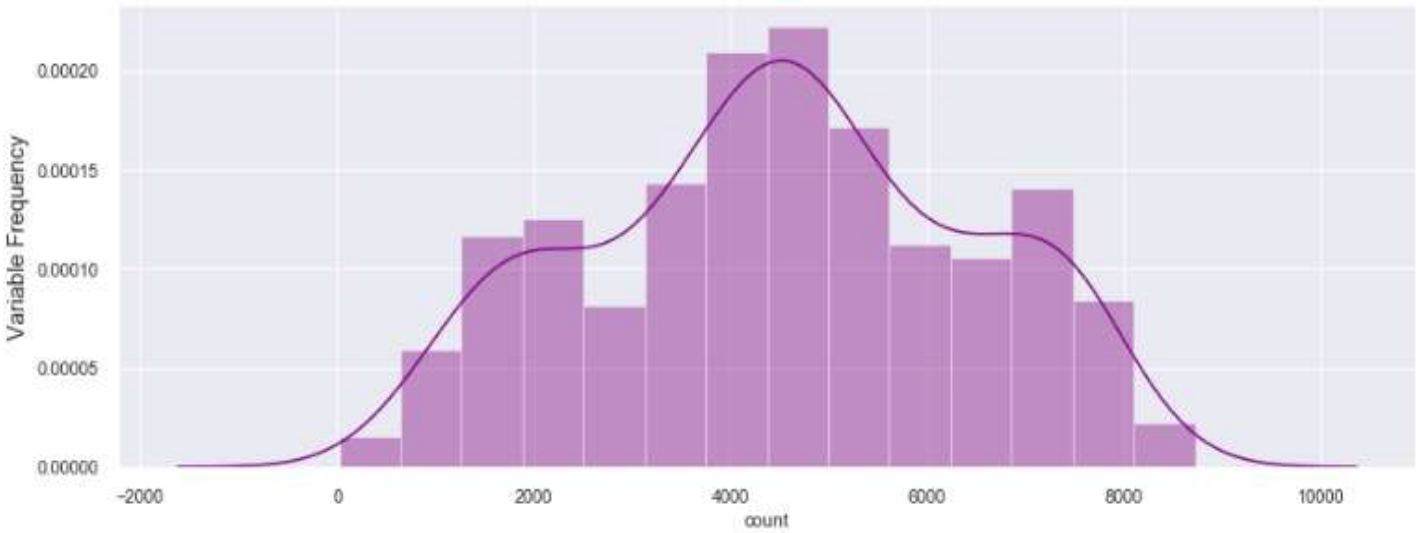
In Python:


Variable Frequency
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
temperature


Variable Frequency
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
atemp


Variable Frequency
2.5
2.0
1.5
1.0
0.5
0.0
0.2
0.4
0.6
0.8
1.0
humidity


Variable Frequency
5
4
3
2
1
0
0.0
0.1
0.2
0.3
0.4
windspeed

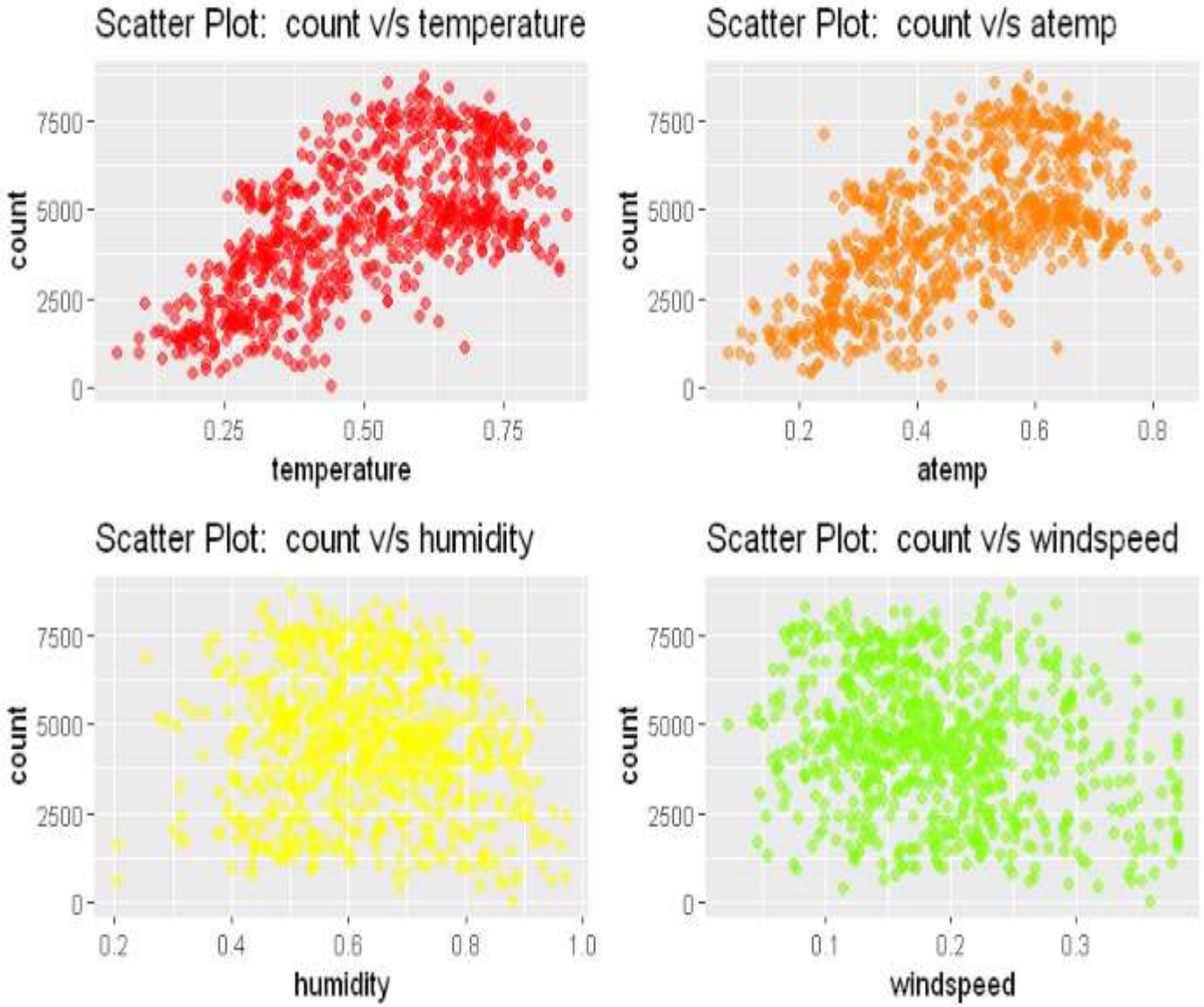

Variable Frequency
0.00020
0.00015
0.00010
0.00005
0.00000
-2000
0
2000
4000
6000
8000
10000
count

Effect of continuous variables with respect to target variable

Let's check the impact of each continuous variables on target variable using scatterplot

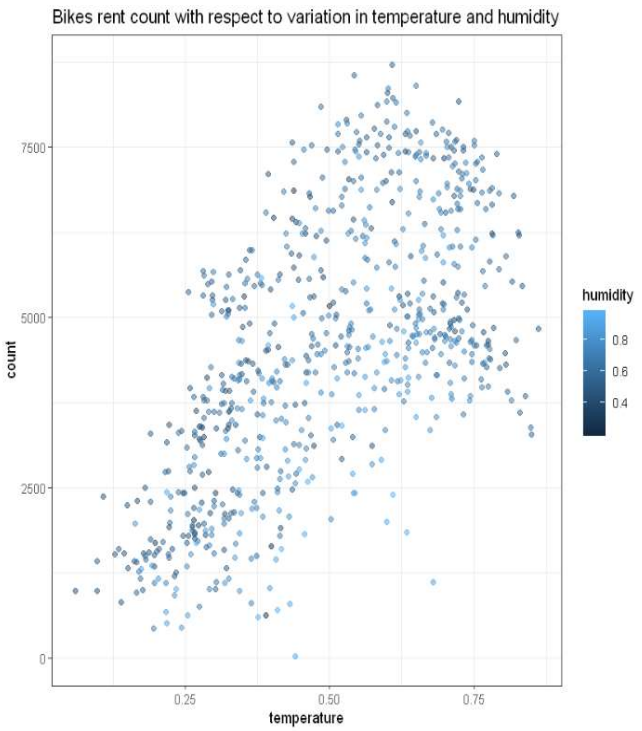
In R:

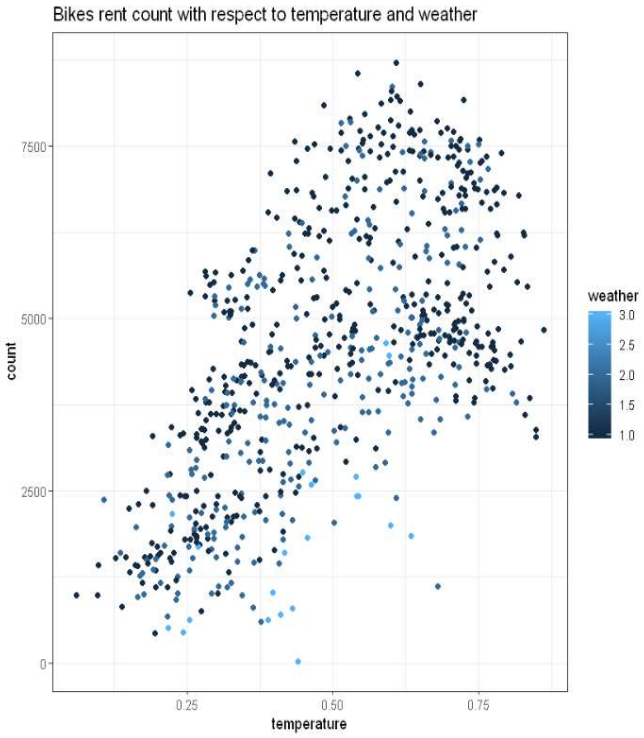
Count Vs Temperature: From the below plot we can say as temperature increase Bike rental count also increases.

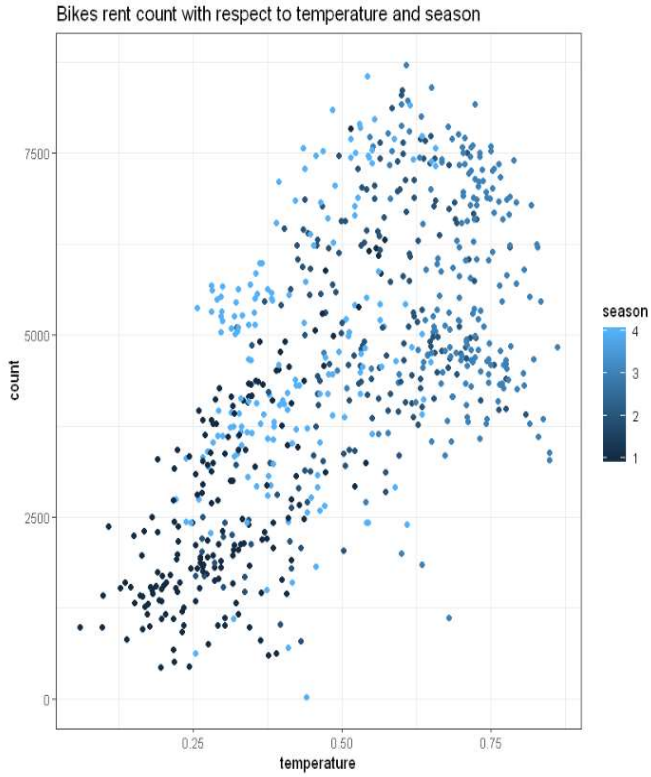


Count Vs Humidity: From the below plot we can say humidity doesn't have any effect on bike rent count

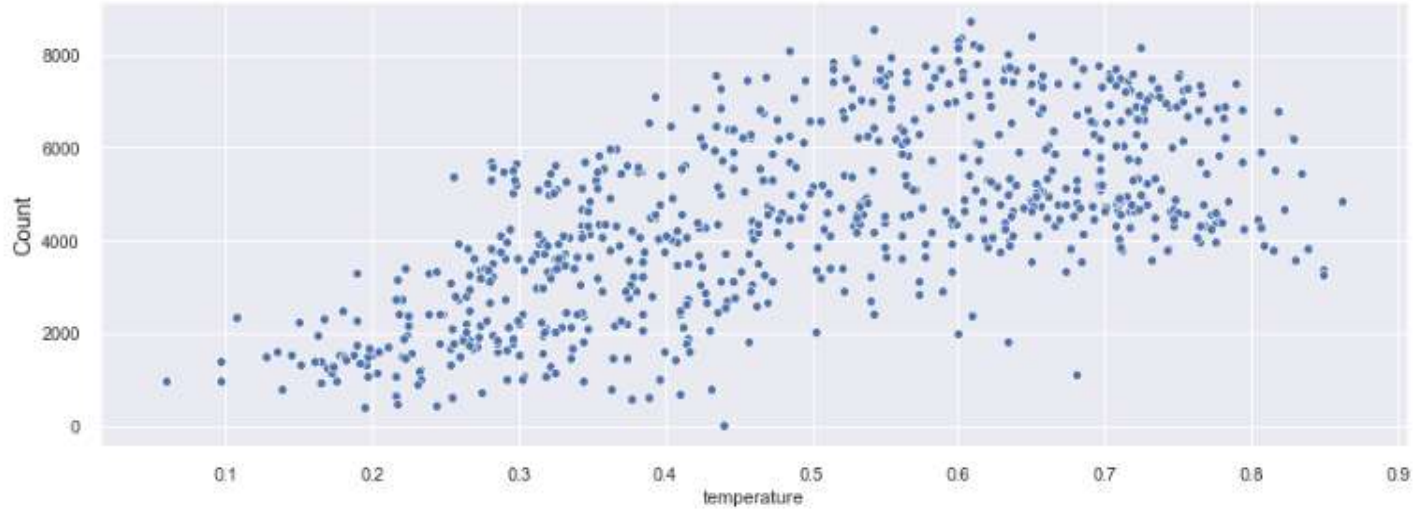
Count Vs Windspeed: From the below plot we can say windspeed doesn't have any effect on bike rent count

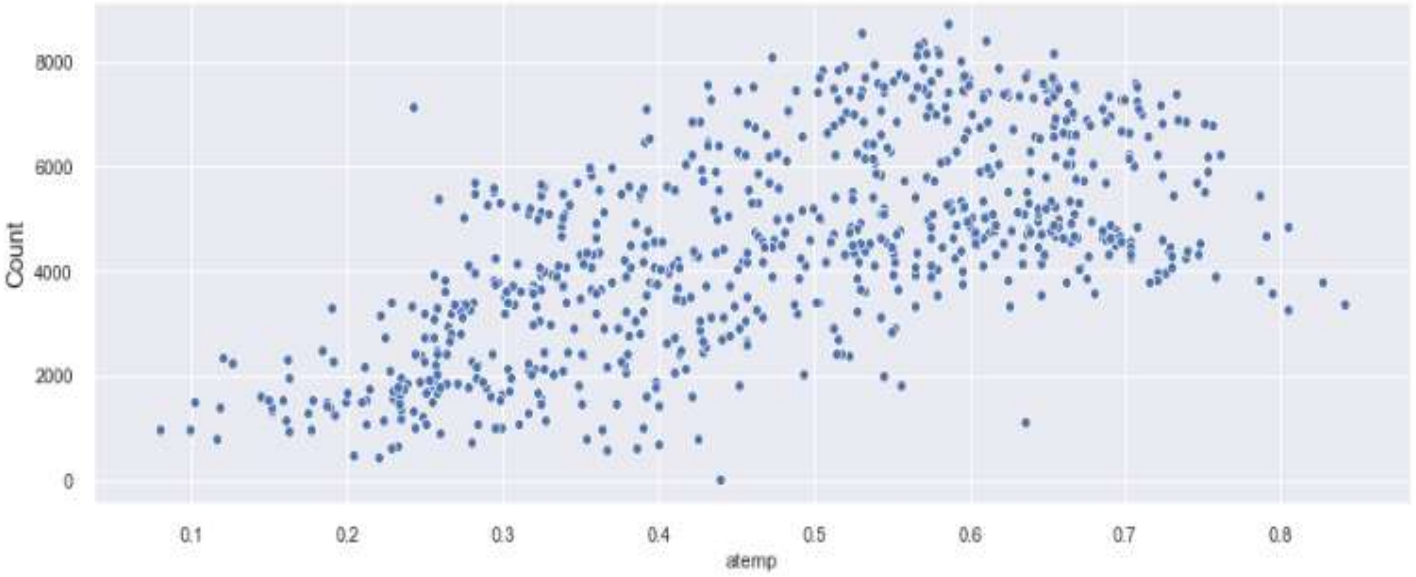

Bikes rent count with respect to variation in temperature and humidity
count
temperature
humidity
0.8
0.6
0.4
7500
5000
2500
0
0.25
0.50
0.75

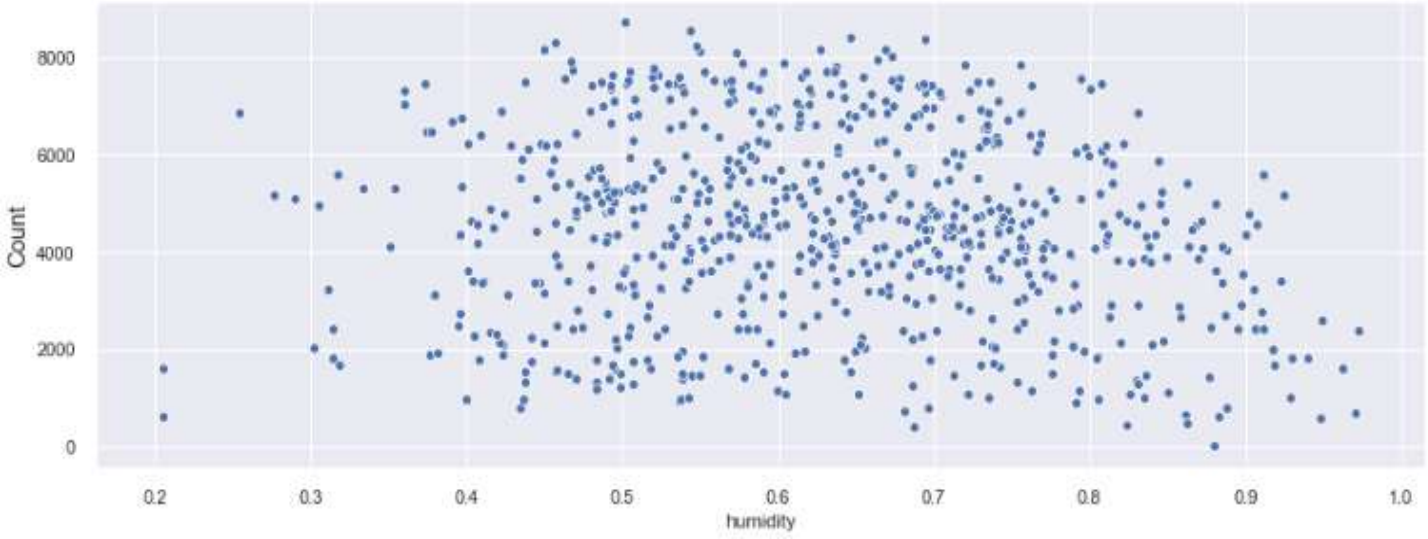

Bikes rent count with respect to temperature and weather
count
temperature
weather
3.0
2.5
2.0
1.5
1.0
7500
5000
2500
0
0.25
0.50
0.75

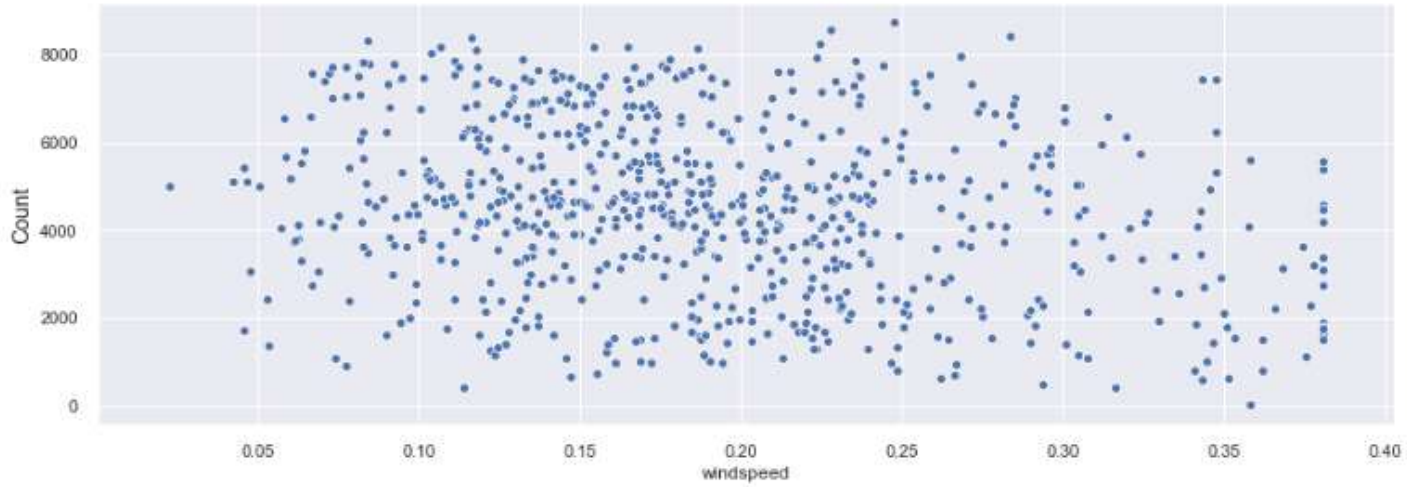

Bikes rent count with respect to temperature and season
count
temperature
season
4
3
2
1
7500
5000
2500
0
0.25
0.50
0.75

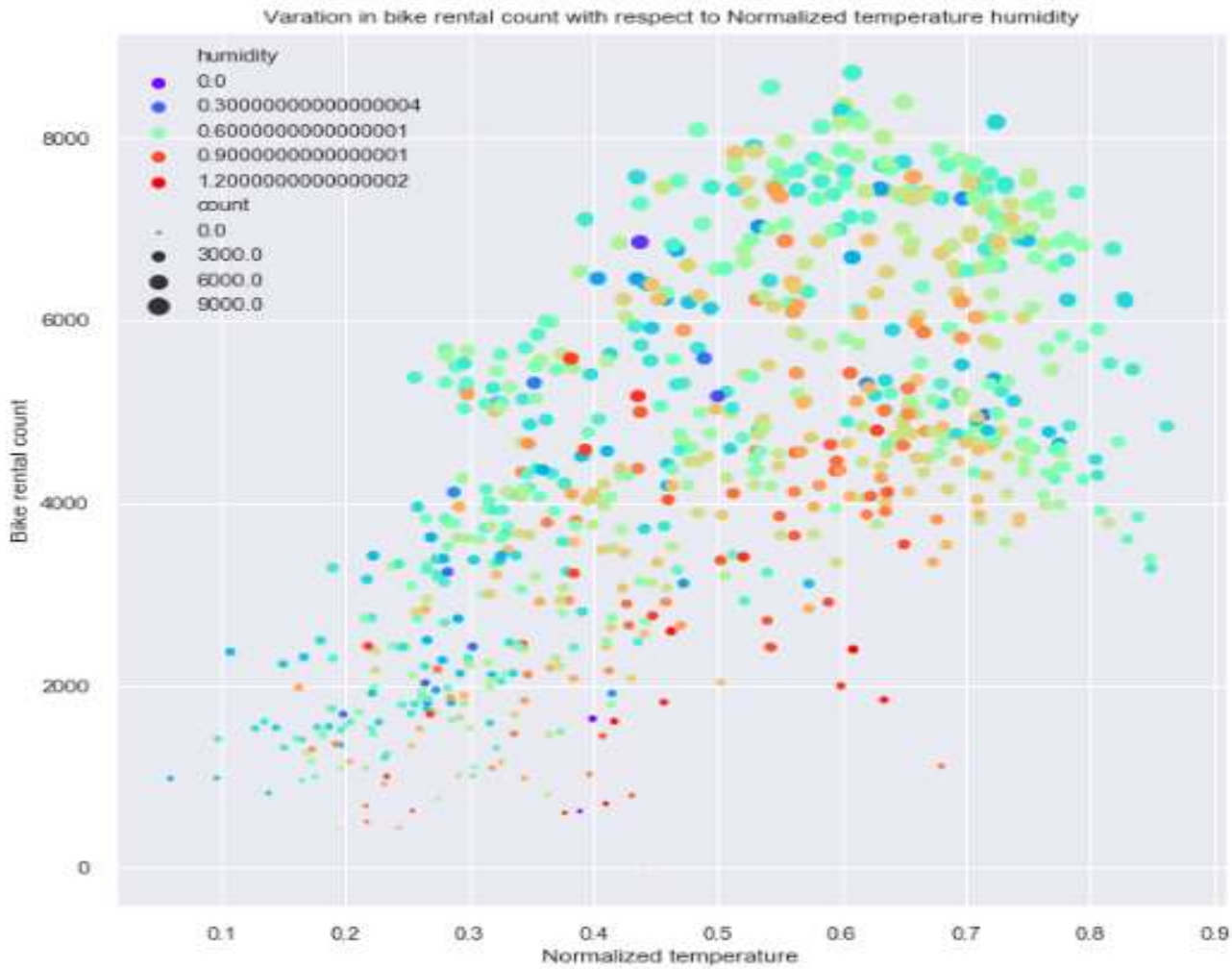
In Python:

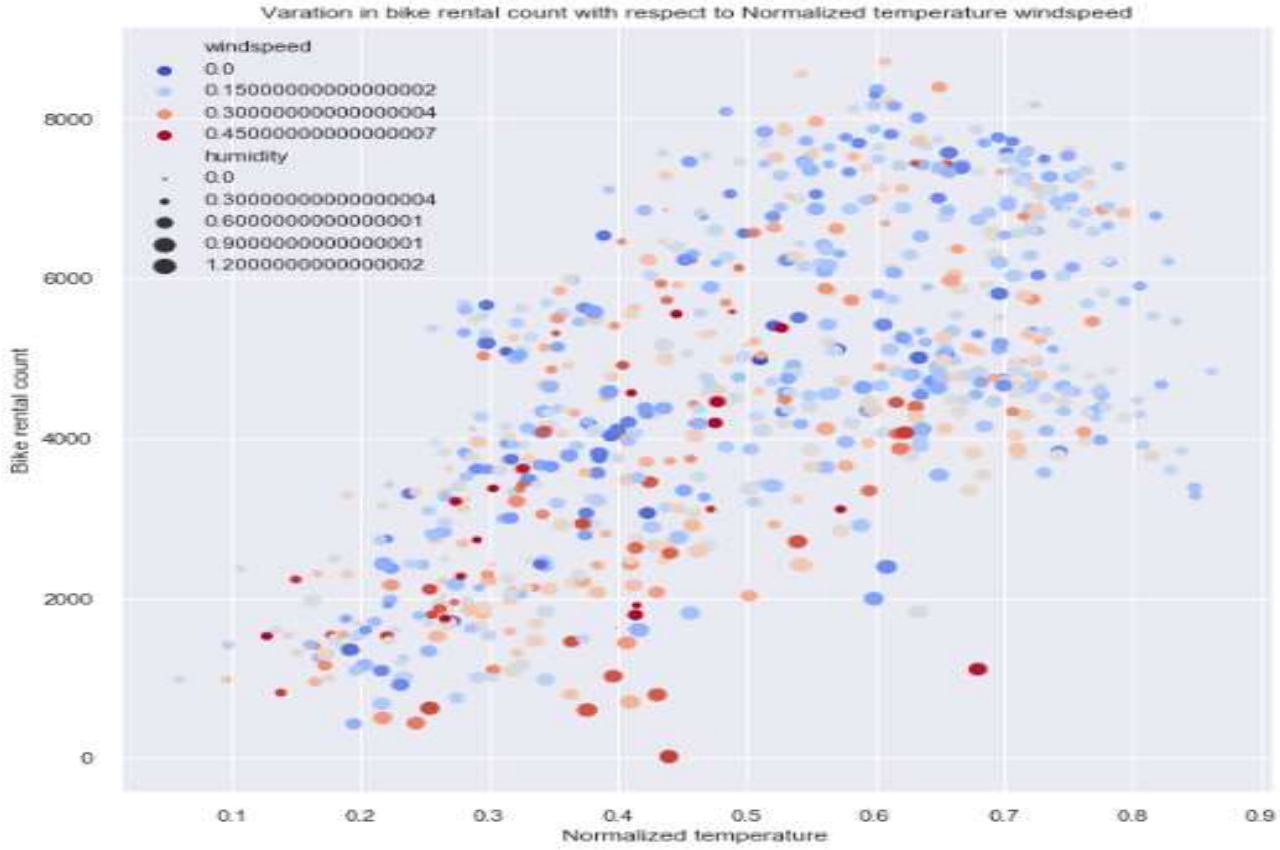

Count
8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
temperature

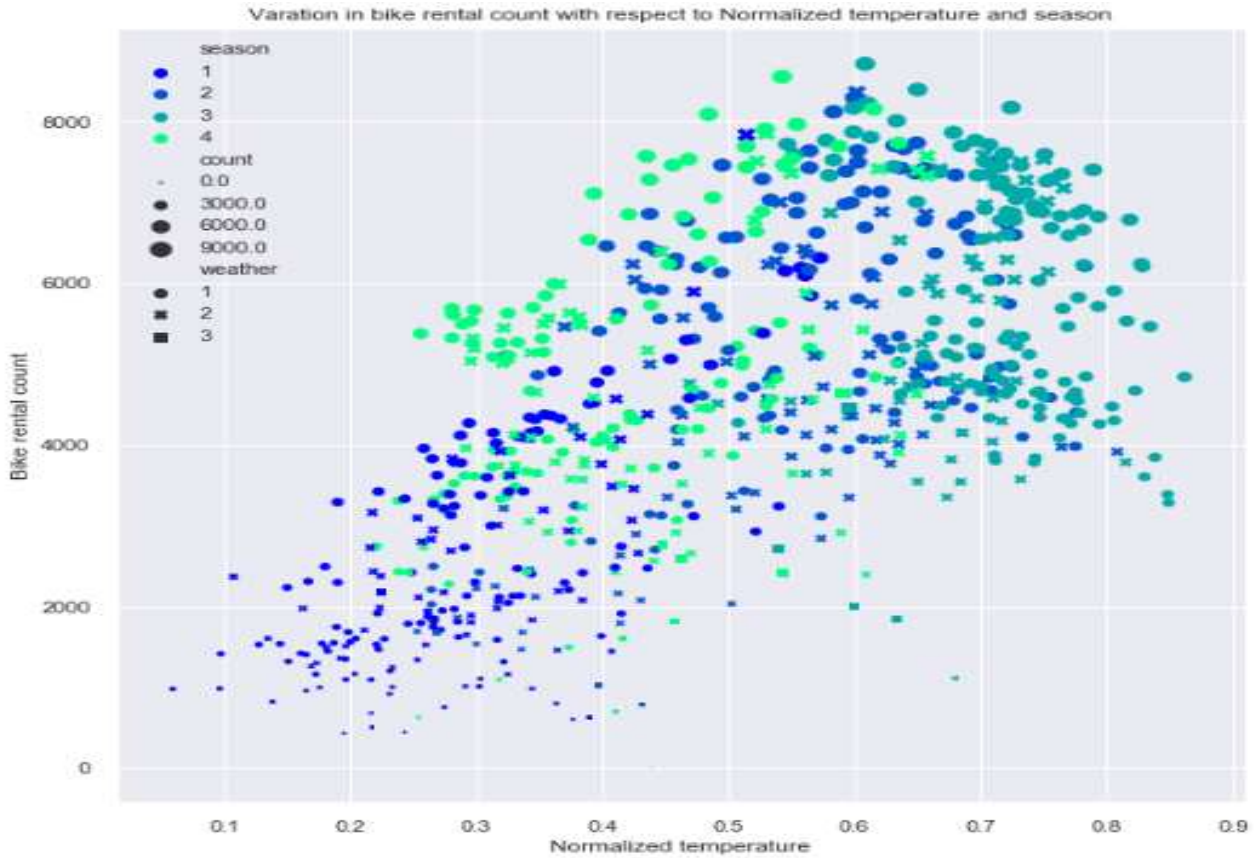

Count
8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
atemp


Count
8000
6000
4000
2000
0
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1.0
humidity


Count
8000
6000
4000
2000
0
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
windspeed


Varation in bike rental count with respect to Normalized temperature humidity
humidity
0.0
0.30000000000000004
0.6000000000000001
0.9000000000000001
1.2000000000000002
count
0.0
3000.0
6000.0
9000.0
Bike rental count
8000
6000
4000
2000
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Normalized temperature


Varation in bike rental count with respect to Normalized temperature windspeed
windspeed
0.0
0.15000000000000002
0.30000000000000004
0.45000000000000007
humidity
0.0
0.30000000000000004
0.6000000000000001
0.9000000000000001
1.2000000000000002
Bike rental count
Normalized temperature


Varation in bike rental count with respect to Normalized temperature and season
season
1
2
3
4
count
0.0
3000.0
6000.0
9000.0
weather
1
2
3
Bike rental count
Normalized temperature

Effect of categorical variables on target variable

To check distribution of each categorical variable with respect to target variable we used bar plot and we can also look at the frequency table

Count Vs season :- Bike rent count is high in season 3(fall) and low in season1(springer)

Count Vs year :- Bike rent count is high in year 1 (in 2012)

Count Vs month :- Bike rent count is high in month of august and low in Jan

Count Vs holiday : - Bike rent count is high on holidays i.e., 0

Count Vs weekday : - From bar plot we can see maximum bikes rented on 5th day and least bikes on day 0.

Count Vs workingday : - Bike rent count is high on working day i.e., 1

Count Vs weather : - Bike rent count is high on weather 1: i.e., when the weather is Clear, Few clouds, Partly cloudy, Partly cloudy

In R:

season	count
1	471348
2	918589
3	1061129
4	841613

year	count
0	1243103
1	2049576

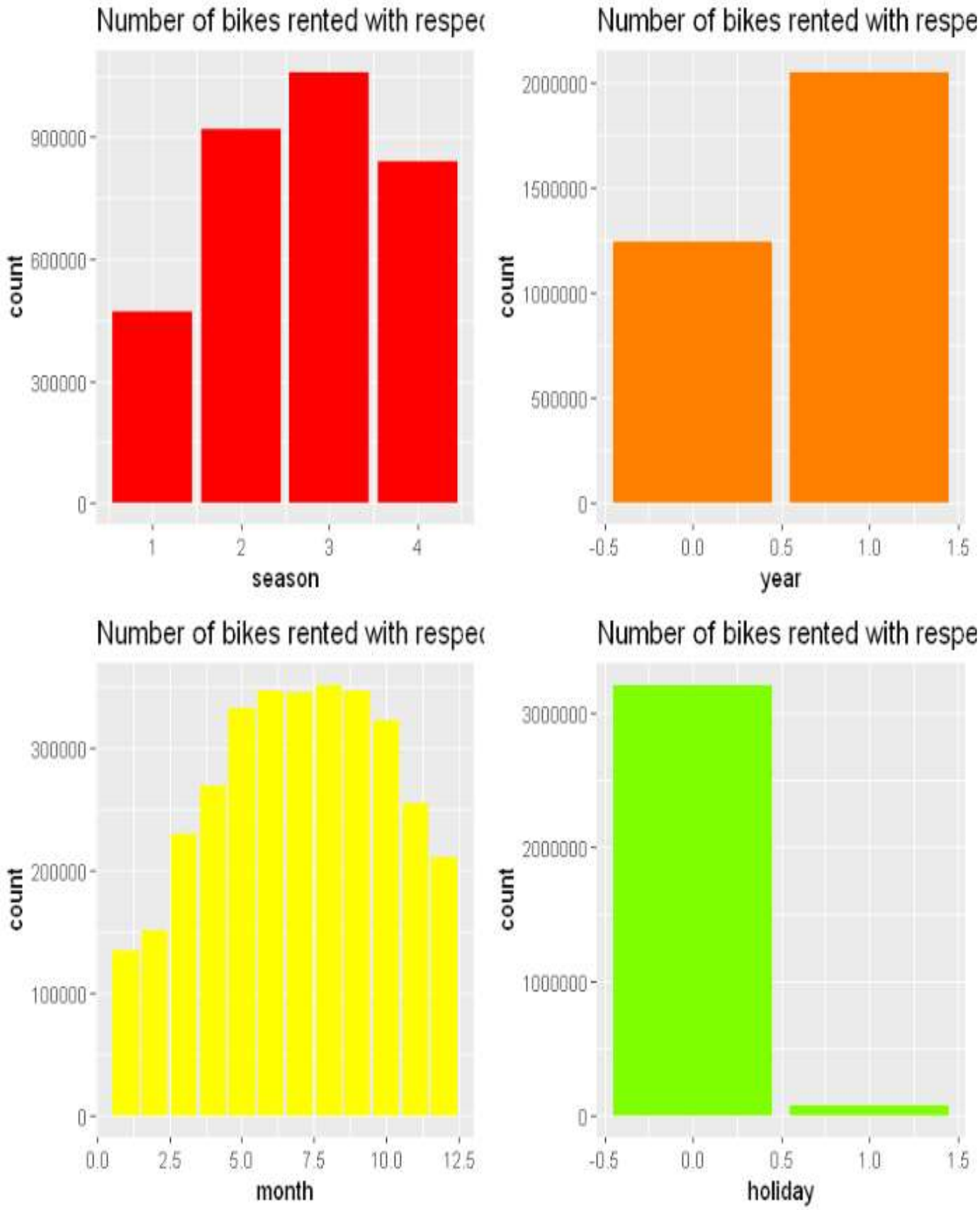
month	count
1	134933
2	151352
3	228920
4	269094
5	331686
6	346342
7	344948
8	351194
9	345991
10	322352
11	254831
12	211036

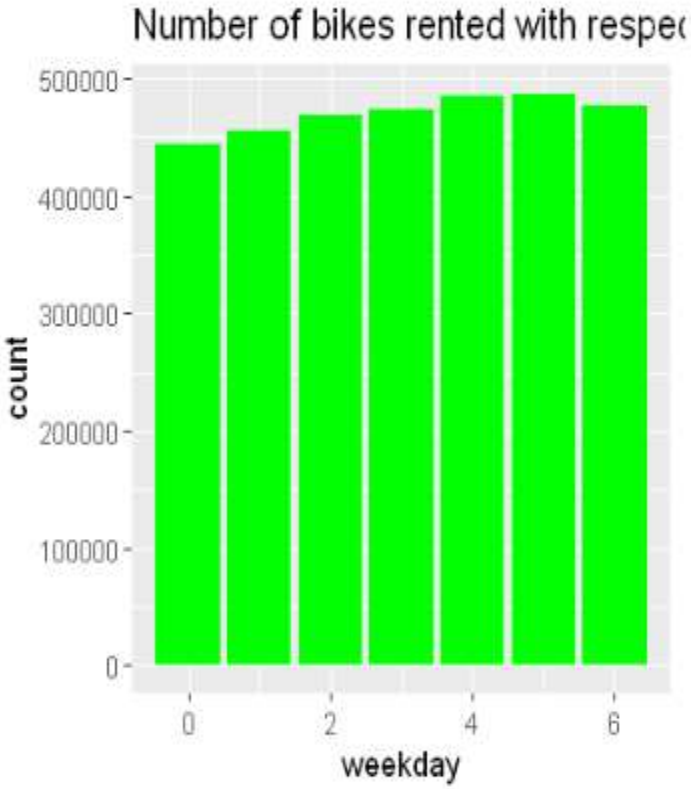
holiday	count
0	3214244
1	78435

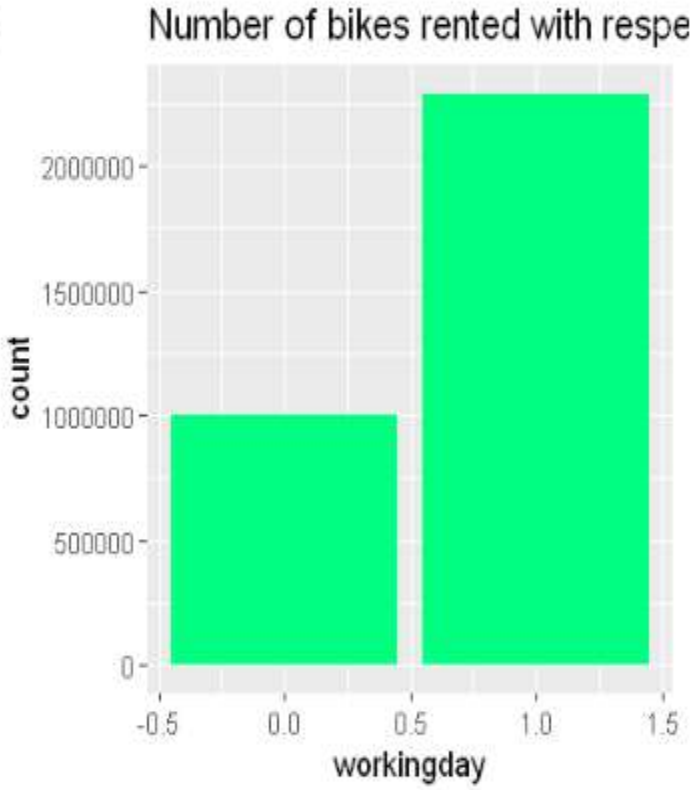
weekday	count
0	444027
1	455503
2	469109
3	473048
4	485395
5	487790
6	477807

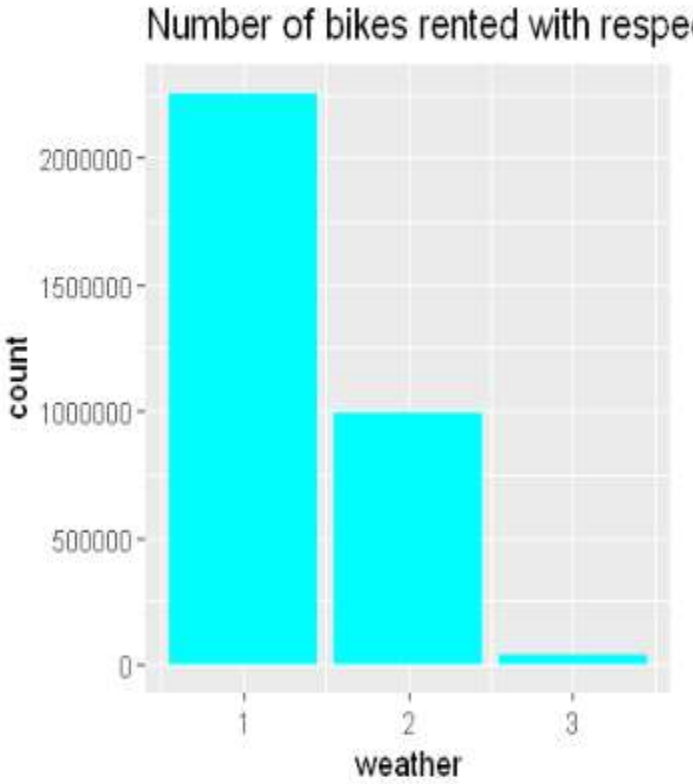
workingday	count
0	1000269
1	2292410

weather	count
1	2257952
2	996858
3	37869

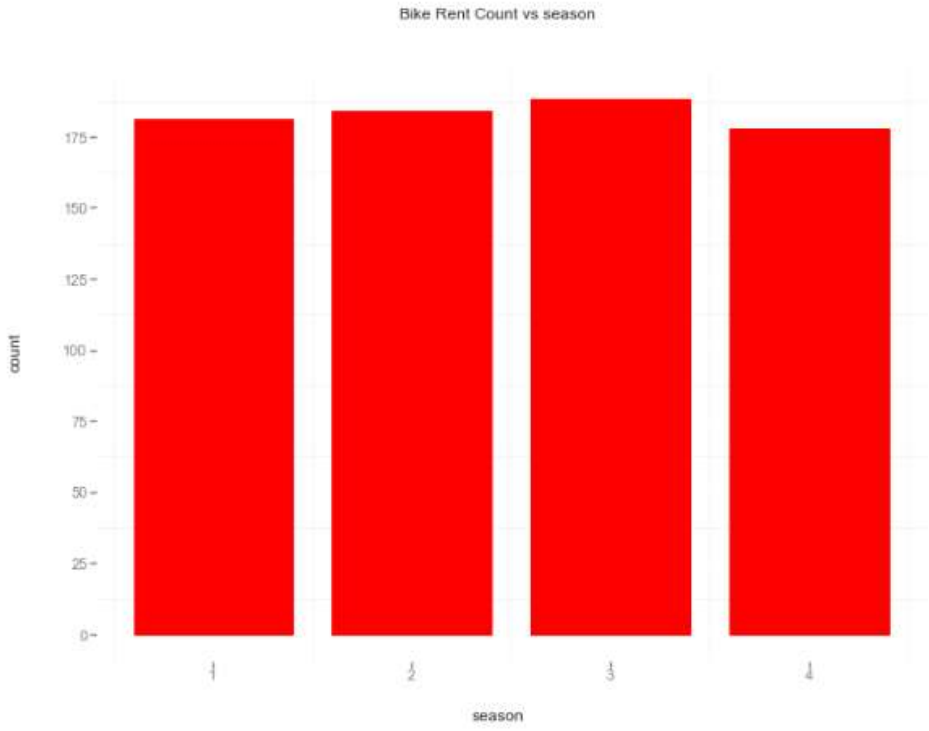

Number of bikes rented with respec
count
900000
600000
300000
0
1
2
3
4
season
Number of bikes rented with respe
count
2000000
1500000
1000000
500000
0
-0.5
0.0
0.5
1.0
1.5
year
Number of bikes rented with respec
count
300000
200000
100000
0
0.0
2.5
5.0
7.5
10.0
12.5
month
Number of bikes rented with respe
count
3000000
2000000
1000000
0
-0.5
0.0
0.5
1.0
1.5
holiday


Number of bikes rented with respec
count
500000
400000
300000
200000
100000
0
0
2
4
6
weekday


Number of bikes rented with respe
count
2000000
1500000
1000000
500000
0
-0.5
0.0
0.5
1.0
1.5
workingday

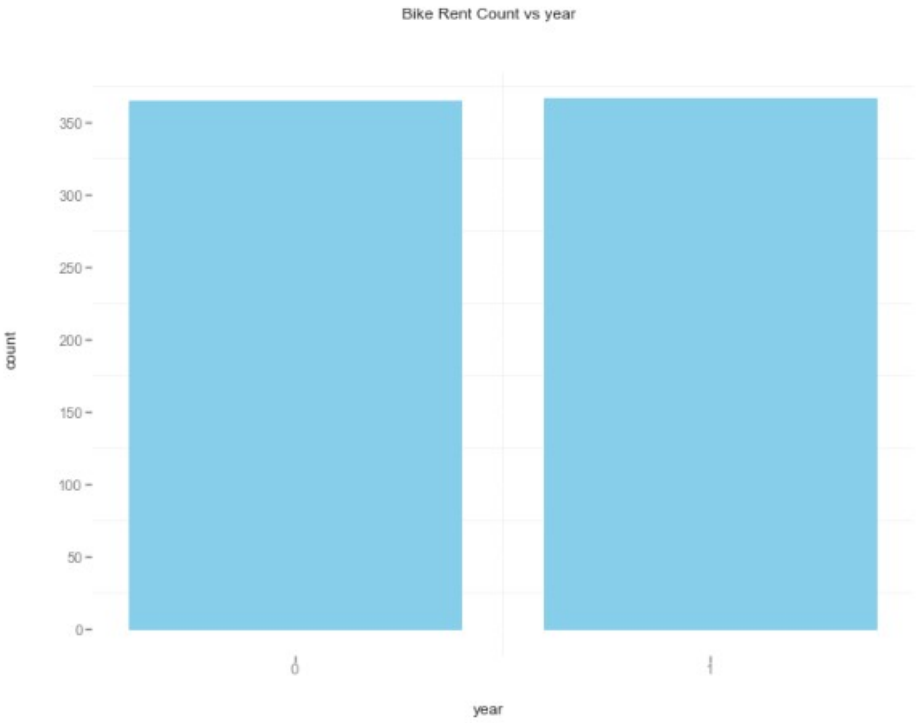

Number of bikes rented with respect to weather
count
2000000
1500000
1000000
500000
0
1
2
3
weather

In Python:



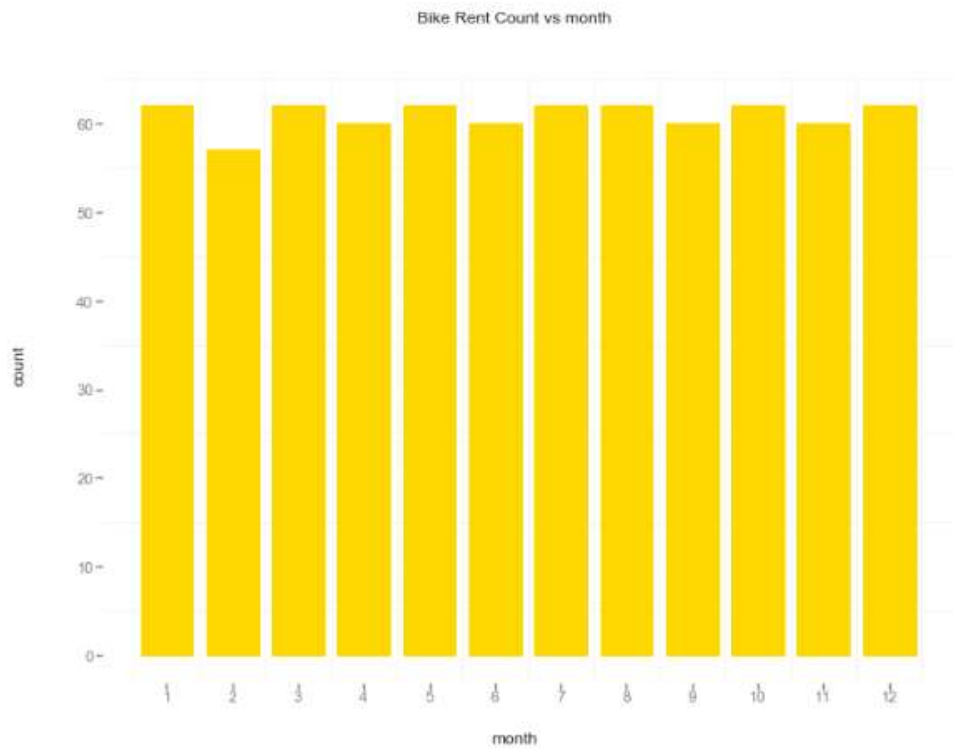

```
season
1     471348.0
2     918589.0
3    1061129.0
4     841613.0
Name: count, dtype: float64
```

Count Vs season :- Bike rent count is high in season 3(fall) and low in season1(springer)



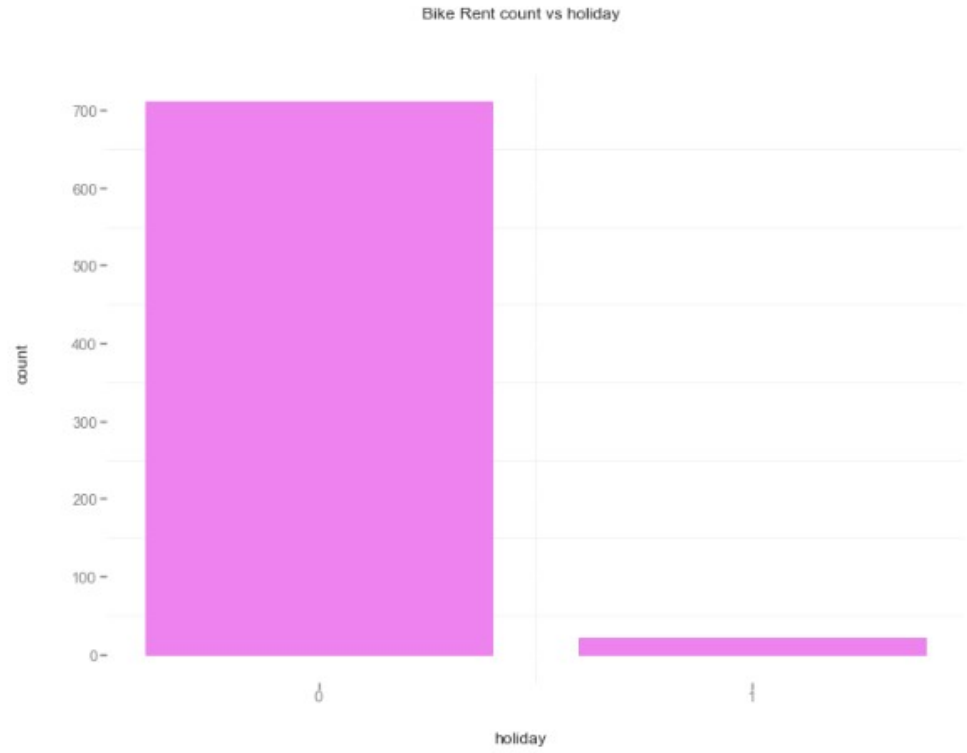

```
year
0    1243103.0
1    2049576.0
Name: count, dtype: float64
```

Count Vs year :- Bike rent count is high in year 1 (in 2012)



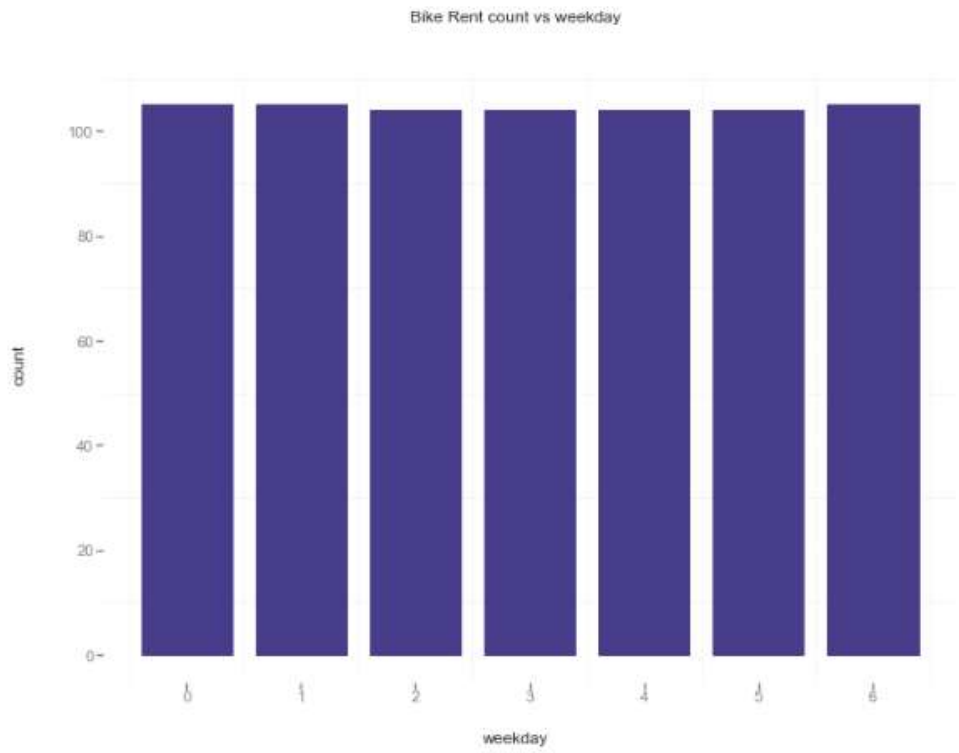

```
month
1     134933.0
2     151352.0
3     228920.0
4     269094.0
5     331686.0
6     346342.0
7     344948.0
8     351194.0
9     345991.0
10    322352.0
11    254831.0
12    211036.0
Name: count, dtype: float64
```

Count Vs month :- Bike rent count is high in month of august and low in Jan



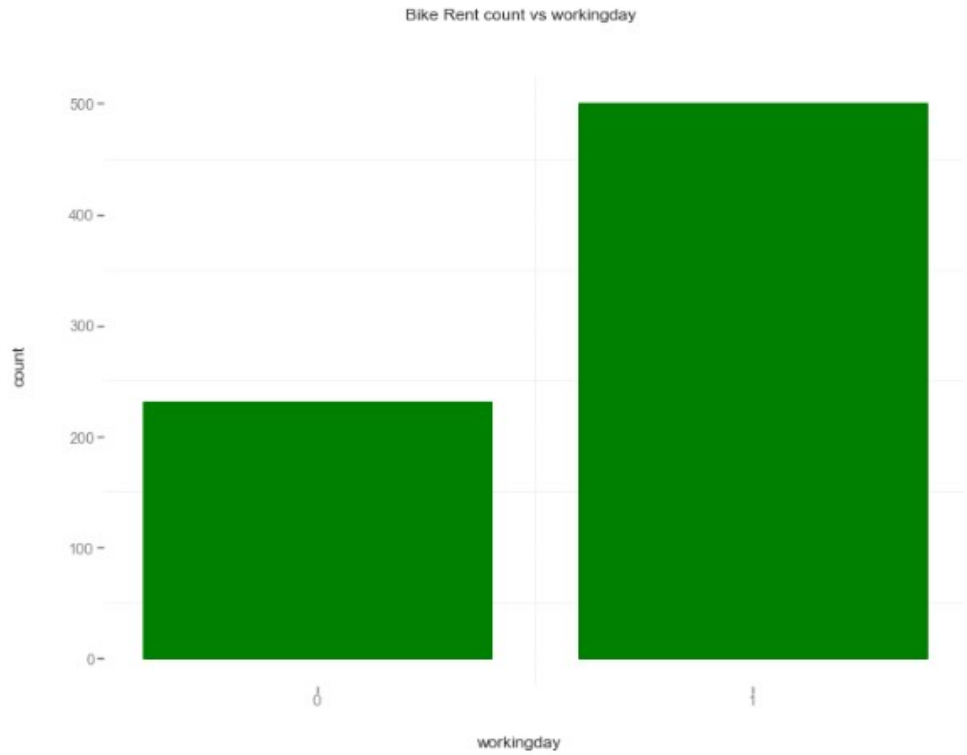

```
holiday
0    3214244.0
1      78435.0
Name: count, dtype: float64
```

Count Vs holiday : - Bike rent count is high on holidays i.e., 0



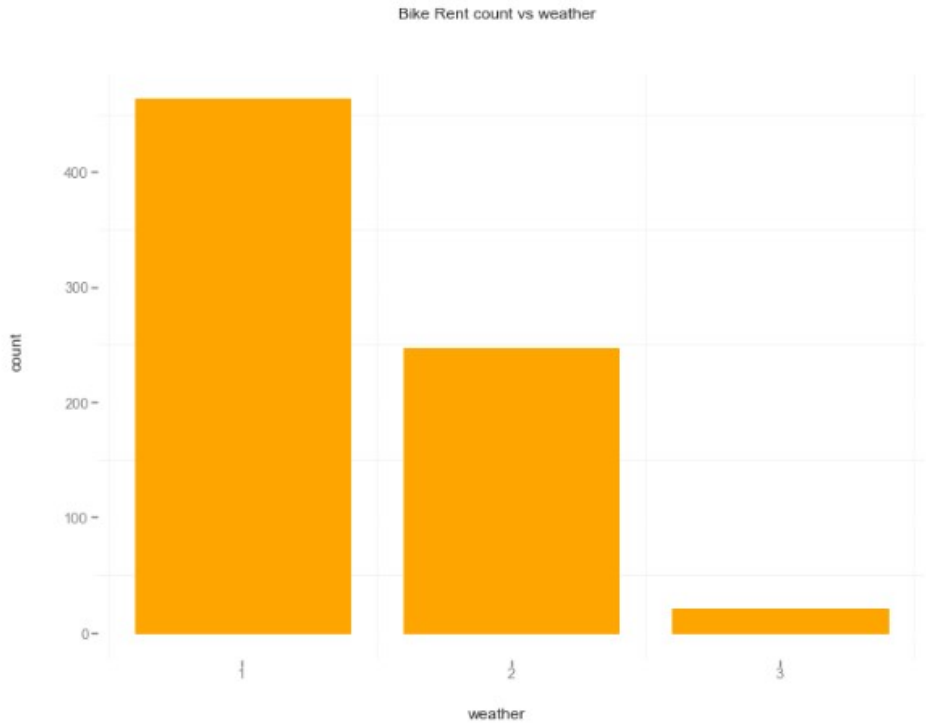

```
weekday
0    444027.0
1    455503.0
2    469109.0
3    473048.0
4    485395.0
5    487790.0
6    477807.0
Name: count, dtype: float64
```

Count Vs weekday : - From bar plot we can see maximum bikes rented on 5th day and least bikes on day 0.



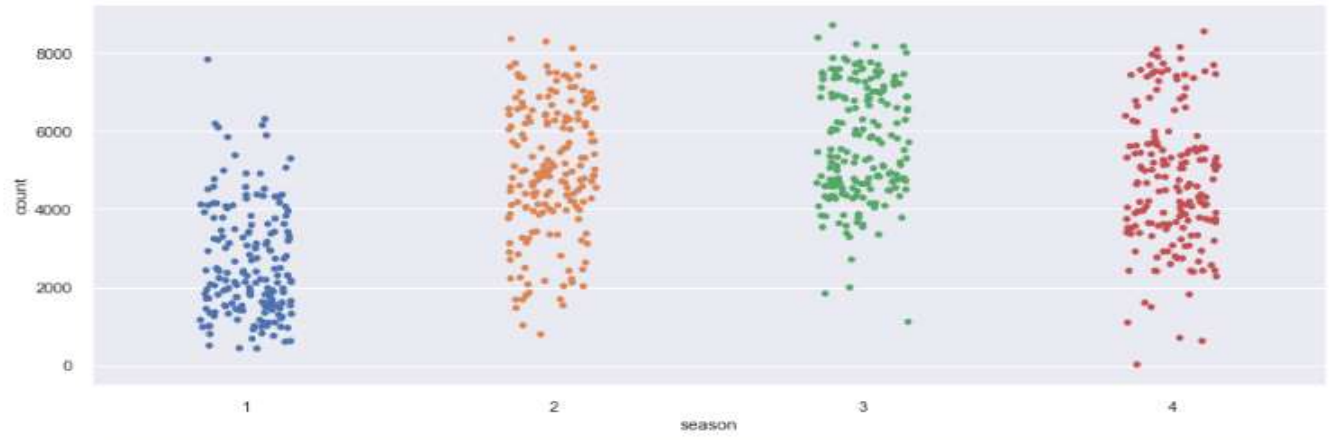

```
workingday
0    1000269.0
1    2292410.0
Name: count, dtype: float64
```

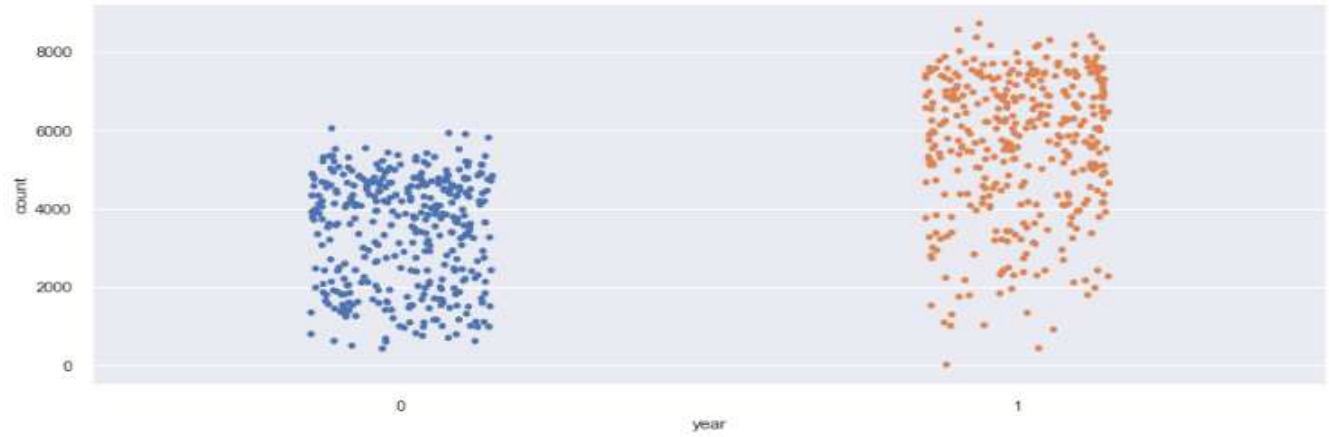
Count Vs workingday : - Bike rent count is high on working day i.e., 1

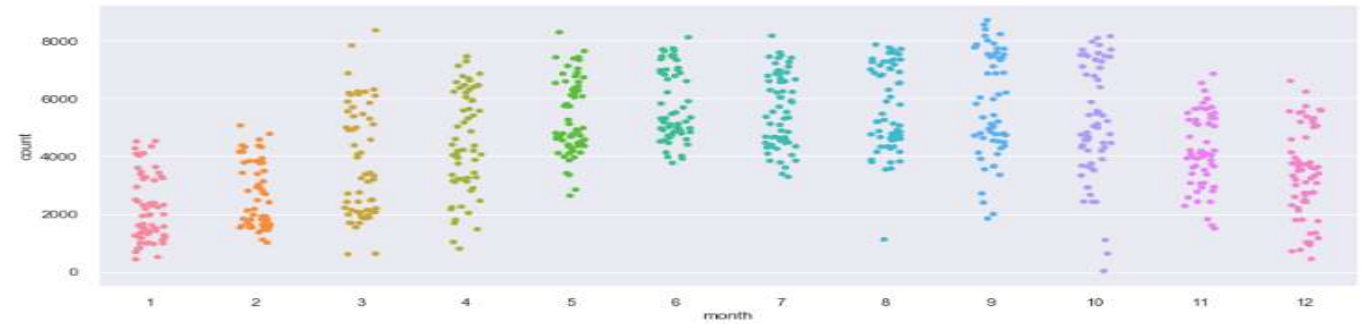



```
weather
1    2257952.0
2     996858.0
3      37869.0
Name: count, dtype: float64
```

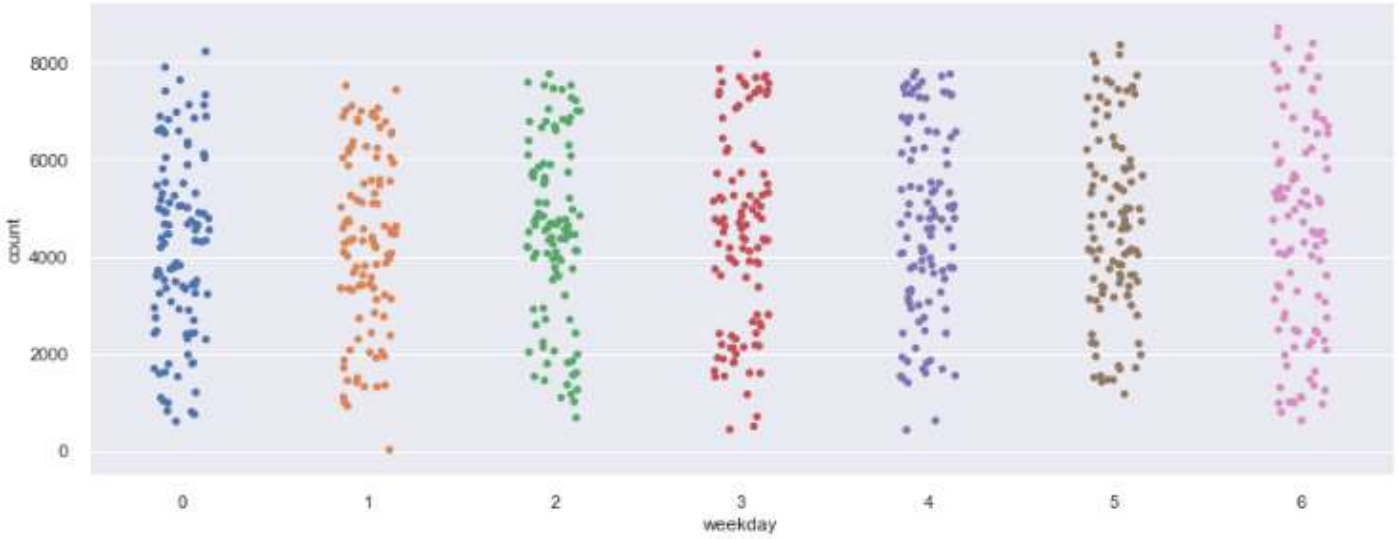
Count Vs weather : - Bike rent count is high on weather 1: i.e., when the weather is Clear, Few clouds, Partly cloudy, Partly cloudy



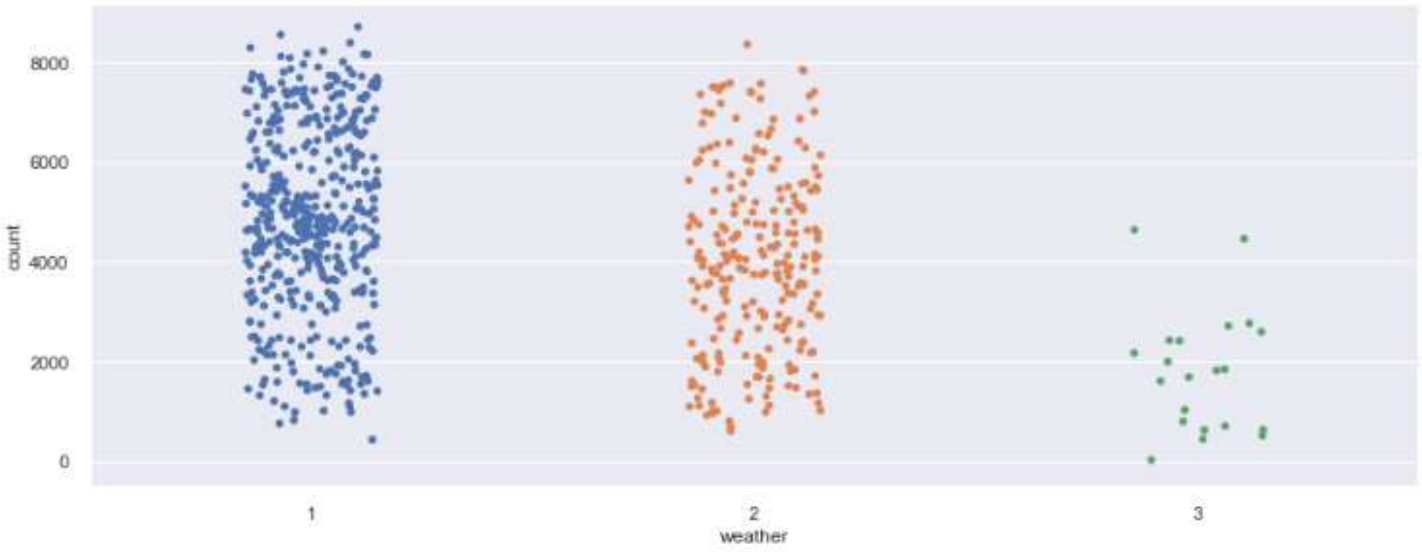



count
8000
6000
4000
2000
0
1
2
3
4
5
6
7
8
9
10
11
12
month


count
8000
6000
4000
2000
0
0
1
holiday


count
8000
6000
4000
2000
0
0
1
2
3
4
5
6
weekday


count
8000
6000
4000
2000
0
0
1
workingday



2.4.6 Feature Selection

Before creating a model, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. Hence, feature selection can help in reducing the time for computation of model as well as the complexity of the model.

Also, few models require the independent variables to be free from multicollinear effect, hence it is needed to perform various procedures to ensure that the independent variables are not collinear.

Correlation analysis includes correlation matrix and correlation plot to find out significant continuous variables and we found that there is no multi-collinearity among the predictor variables.

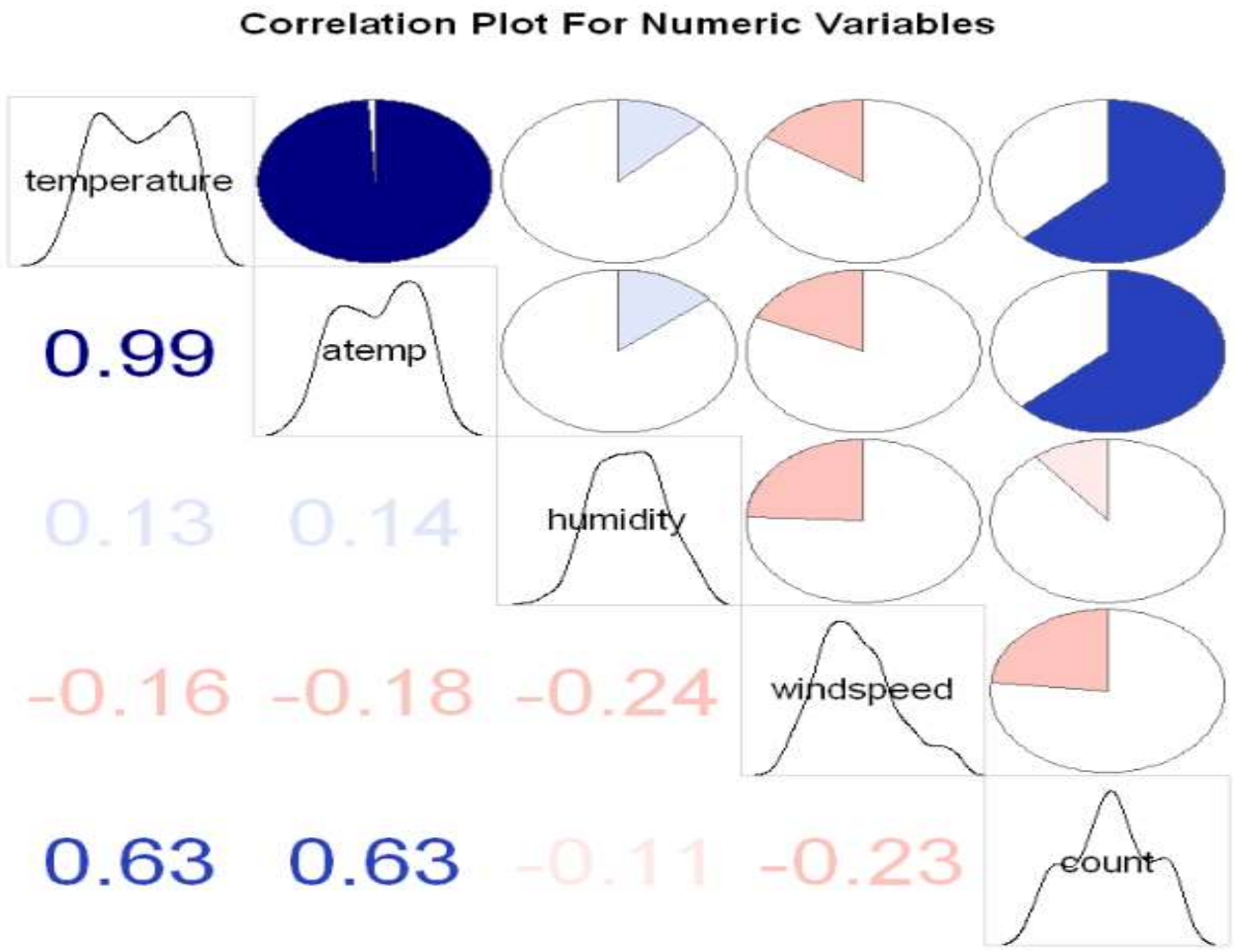
Using ANOVA test to find out significant categorical variable.

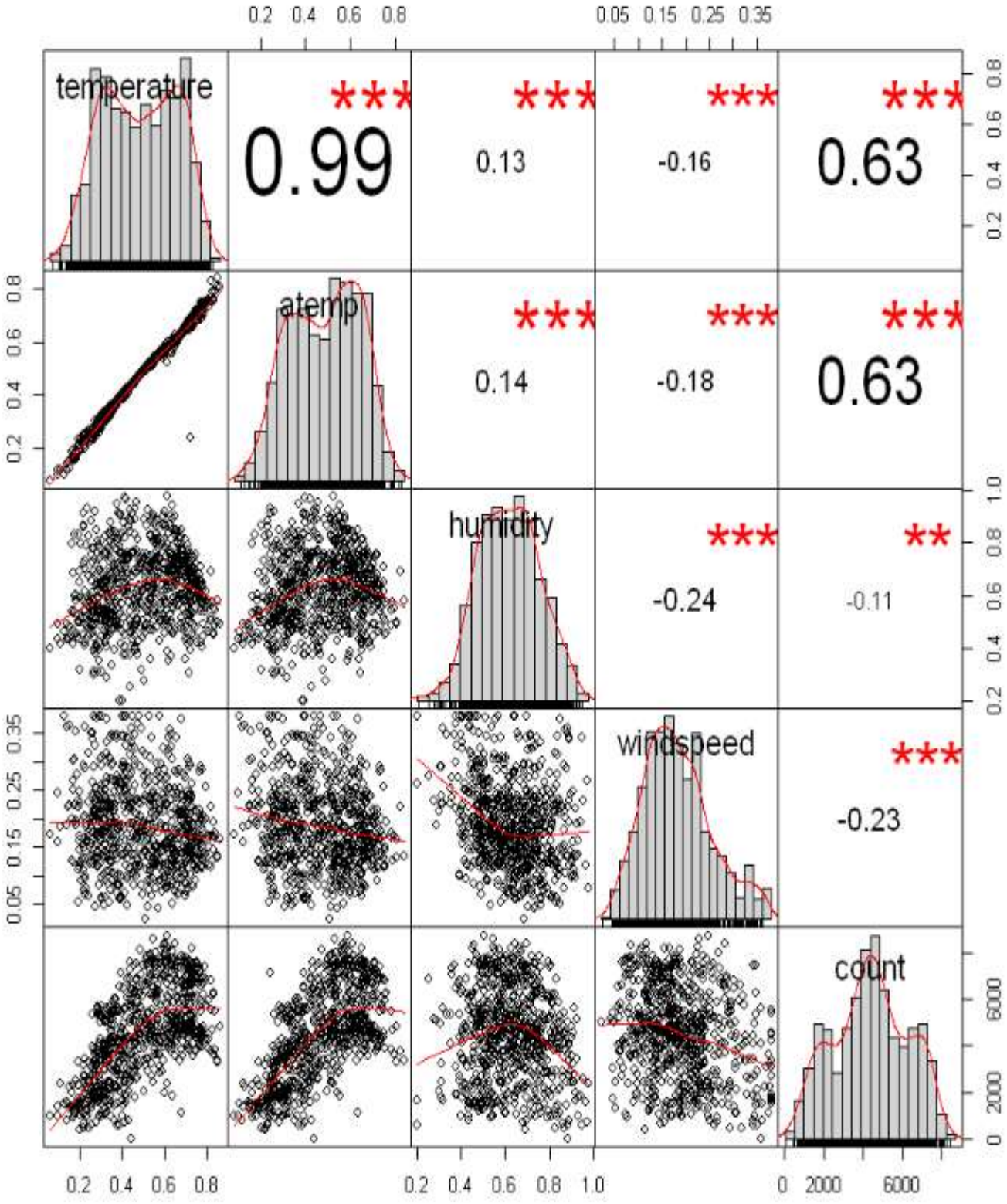
In R: Correlation matrix, Correlation Plot and ANOVA Test Results:

	temperature	atemp	humidity	windspeed	count
temperature	1.0000000	0.9917016	0.1267216	-0.1569155	0.6274940
atemp	0.9917016	1.0000000	0.1399240	-0.1829480	0.6310657
humidity	0.1267216	0.1399240	1.0000000	-0.2411599	-0.1056645
windspeed	-0.1569155	-0.1829480	-0.2411599	1.0000000	-0.2336573
count	0.6274940	0.6310657	-0.1056645	-0.2336573	1.0000000

By looking at correlation matrix we can say temperature and atemp are highly correlated (>0.99)

From correlation matrix and correlation plot we can say temperature variable is positively correlated with atemp variable. and rest of the variables also weekly correlated




temperature
atemp
humidity
windspeed
count
0.99
0.13
-0.16
0.63
0.14
-0.18
0.63
-0.24
-0.11
-0.23
***
**

ANOVA Test Results in R:

```
[1] "season"
                Df     Sum Sq   Mean Sq F value               Pr(>F)
Bike_Rent[, i]   1  451797359 451797359     144 <0.0000000000000002 ***
Residuals      729 2287738033   3138187
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "year"
                Df     Sum Sq   Mean Sq F value               Pr(>F)
Bike_Rent[, i]   1  879828893 879828893   344.9 <0.0000000000000002 ***
Residuals      729 1859706499   2551038
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "month"
                Df     Sum Sq   Mean Sq F value              Pr(>F)
Bike_Rent[, i]   1  214744463 214744463   62.01 0.0000000000000124 ***
Residuals      729 2524790929   3463362
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "holiday"
                Df     Sum Sq  Mean Sq F value Pr(>F)
Bike_Rent[, i]   1   12797494 12797494   3.421 0.0648 .
Residuals      729 2726737898  3740381
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "weekday"
                Df     Sum Sq  Mean Sq F value Pr(>F)
Bike_Rent[, i]   1   12461089 12461089   3.331 0.0684 .
Residuals      729 2727074303  3740843
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "workingday"
                Df     Sum Sq  Mean Sq F value Pr(>F)
Bike_Rent[, i]   1   10246038 10246038   2.737 0.0985 .
Residuals      729 2729289354  3743881
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
[1] "weather"
                Df     Sum Sq   Mean Sq F value               Pr(>F)
Bike_Rent[, i]   1  242288753 242288753   70.73 <0.0000000000000002 ***
Residuals      729 2497246639   3425578
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

From the anova result, we can observe working day, weekday and holiday has p value > 0.05, so delete these variables, not to be consider in model.

After Correlation and ANOVA Analysis we have remaining variables in our dataset as below

731 8

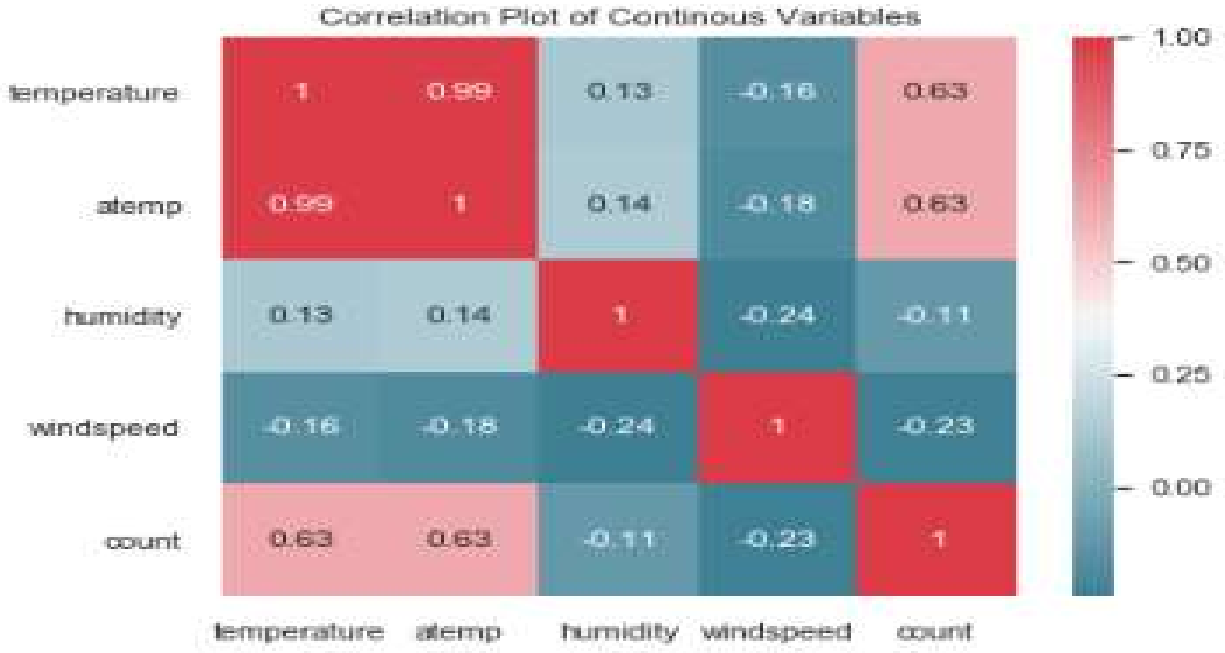
season	year	month	weather	temperature	humidity	windspeed	count
1	0	1	2	0.344167	0.805833	0.1604460	985
1	0	1	2	0.363478	0.696087	0.2485390	801
1	0	1	1	0.196364	0.437273	0.2483090	1349
1	0	1	1	0.200000	0.590435	0.1602960	1562
1	0	1	1	0.226957	0.436957	0.1869000	1600
1	0	1	1	0.204348	0.518261	0.0895652	1606

'season' 'year' 'month' 'weather' 'temperature' 'humidity' 'windspeed' 'count'

In Python: Correlation matrix, Correlation Plot and ANOVA Test Results:

	temperature	atemp	humidity	windspeed	count
temperature	1.000000	0.991702	0.126722	-0.156916	0.627494
atemp	0.991702	1.000000	0.139924	-0.182948	0.631066
humidity	0.126722	0.139924	1.000000	-0.241160	-0.105664
windspeed	-0.156916	-0.182948	-0.241160	1.000000	-0.233657
count	0.627494	0.631066	-0.105664	-0.233657	1.000000

From correlation matrix and correlation plot we can say distance variable is positively correlated with target variable which means this variable carries more information to predict the target variable. rest other variables also weekly correlated.



In Python: -ANOVA Test Results:

	sum_sq	df	F	PR(>F)
season	4.517974e+08	1.0	143.967653	2.133997e-30
Residual	2.287738e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
year	8.798289e+08	1.0	344.890586	2.483540e-63
Residual	1.859706e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
month	2.147445e+08	1.0	62.004625	1.243112e-14
Residual	2.524791e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
holiday	1.279749e+07	1.0	3.421441	0.064759
Residual	2.726738e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
weekday	1.246109e+07	1.0	3.331091	0.068391
Residual	2.727074e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
workingday	1.024604e+07	1.0	2.736742	0.098495
Residual	2.729289e+09	729.0	NaN	NaN
	sum_sq	df	F	PR(>F)
weather	2.422888e+08	1.0	70.729298	2.150976e-16
Residual	2.497247e+09	729.0	NaN	NaN

From the anova result, we can observe working day, weekday and holiday has p value > 0.05, so delete this variable not consider in model.

After Correlation and ANOVA Analysis we have remaining variables are

```
(731, 8)
```

```
Index(['season', 'year', 'month', 'weather', 'temperature', 'humidity',
       'windspeed', 'count'],
      dtype='object')
```

	season	year	month	weather	temperature	humidity	windspeed	count
0	1	0	1	2	0.344167	0.805833	0.160446	985.0
1	1	0	1	2	0.363478	0.696087	0.248539	801.0
2	1	0	1	1	0.196364	0.437273	0.248309	1349.0
3	1	0	1	1	0.200000	0.590435	0.160296	1562.0
4	1	0	1	1	0.226957	0.436957	0.186900	1600.0

2.4.6 Feature Scaling

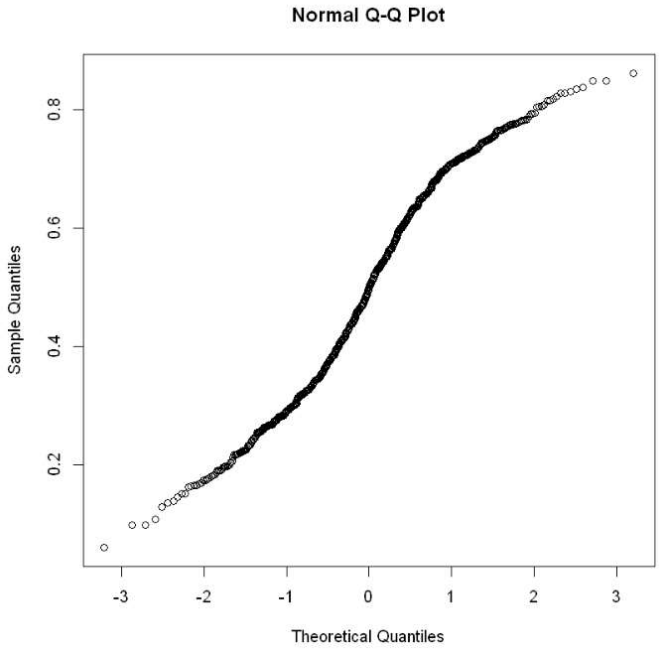
Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. Widely used feature scaling methods are min max scaling and Standardization.

In the given dataset, for continuous variables data is already Normalized, so we need not to scale the data. We can check normality of data using normal QQ plot, histogram plot and summary of variables

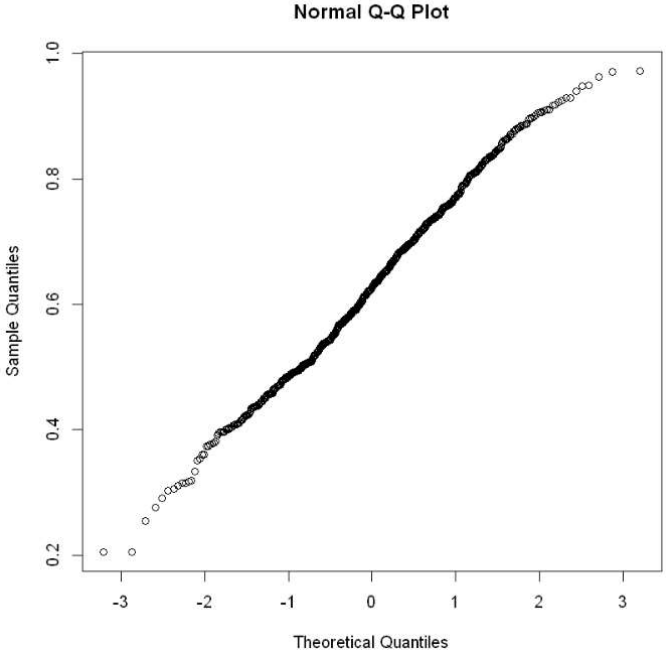
In R : Let's check the Normality of numeric variables

Normal QQ Plots:

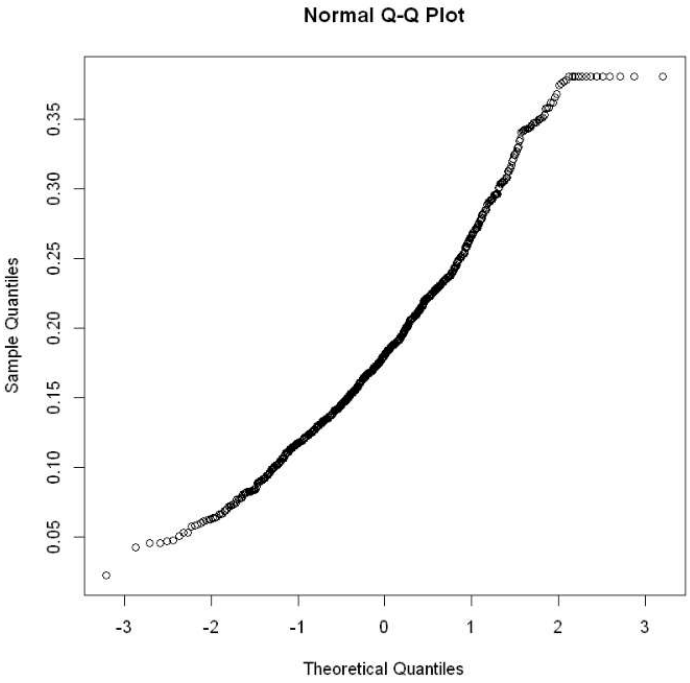
[1] "temperature"



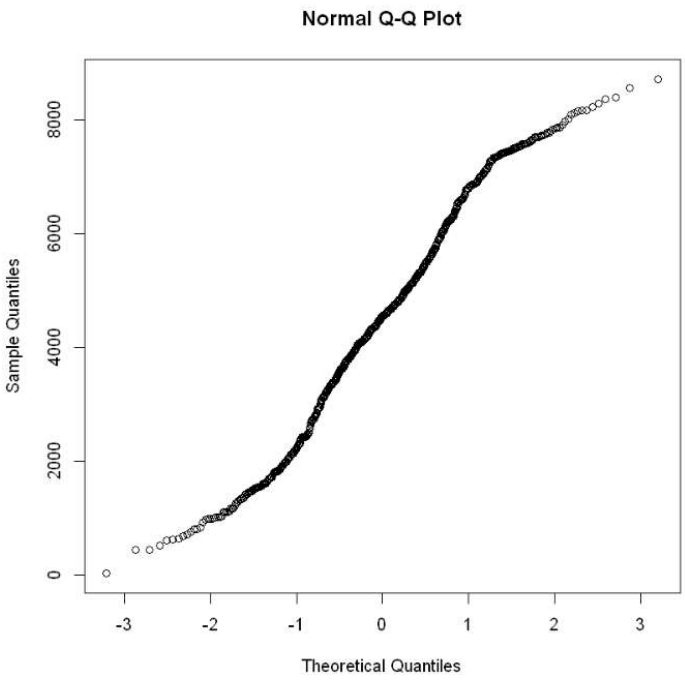
[1] "humidity"

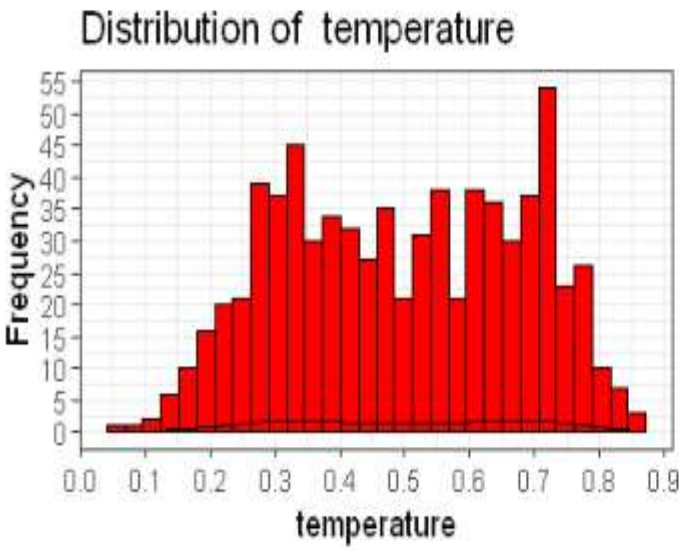


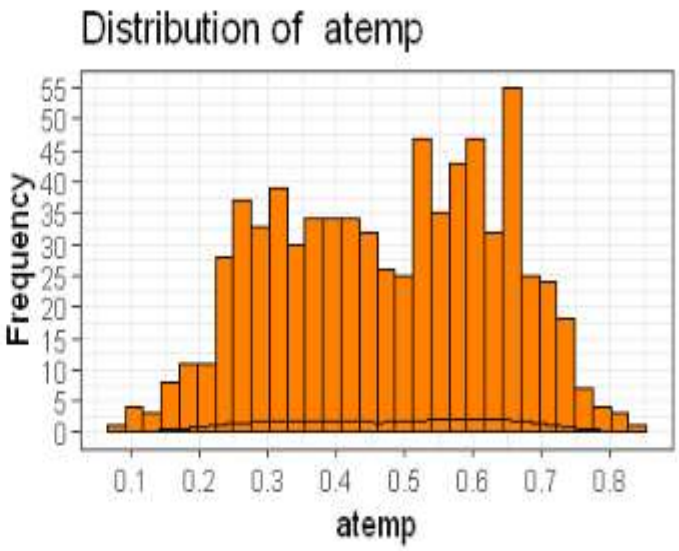
[1] "windspeed"

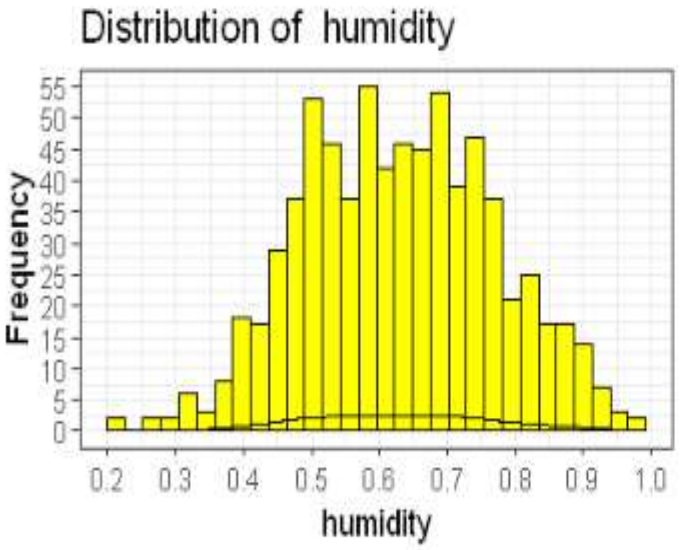


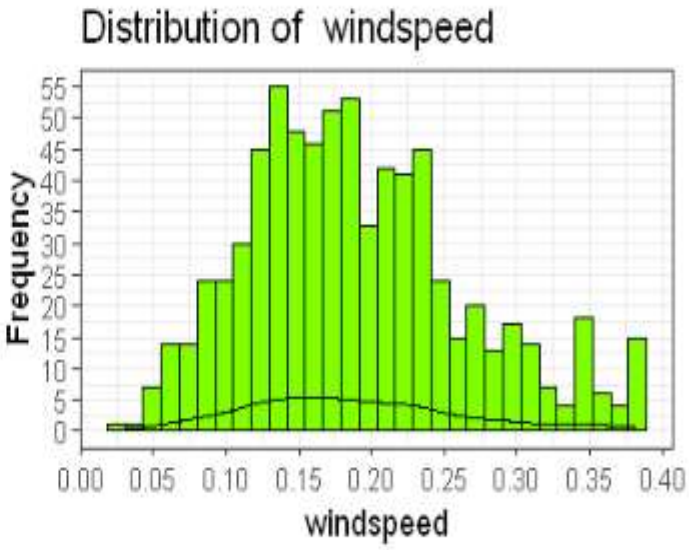
[1] "count"

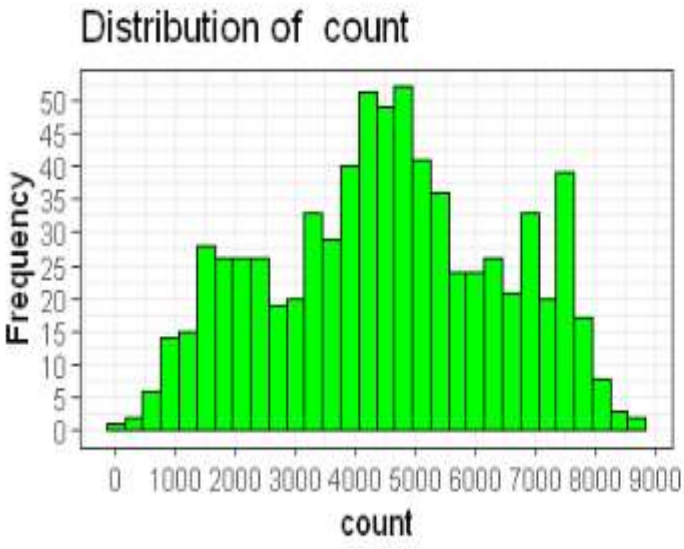



Distribution of temperature
Frequency
temperature


Distribution of atemp
Frequency
atemp


Distribution of humidity
Frequency
humidity


Distribution of windspeed
Frequency
windspeed

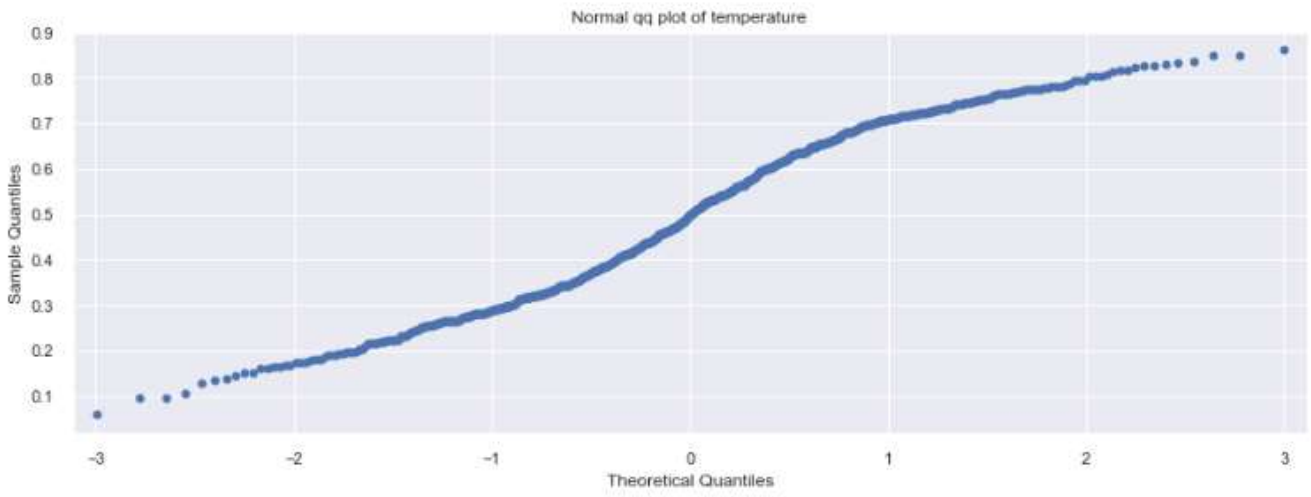

Distribution of count
Frequency
count

Summary of Continuous Variables:

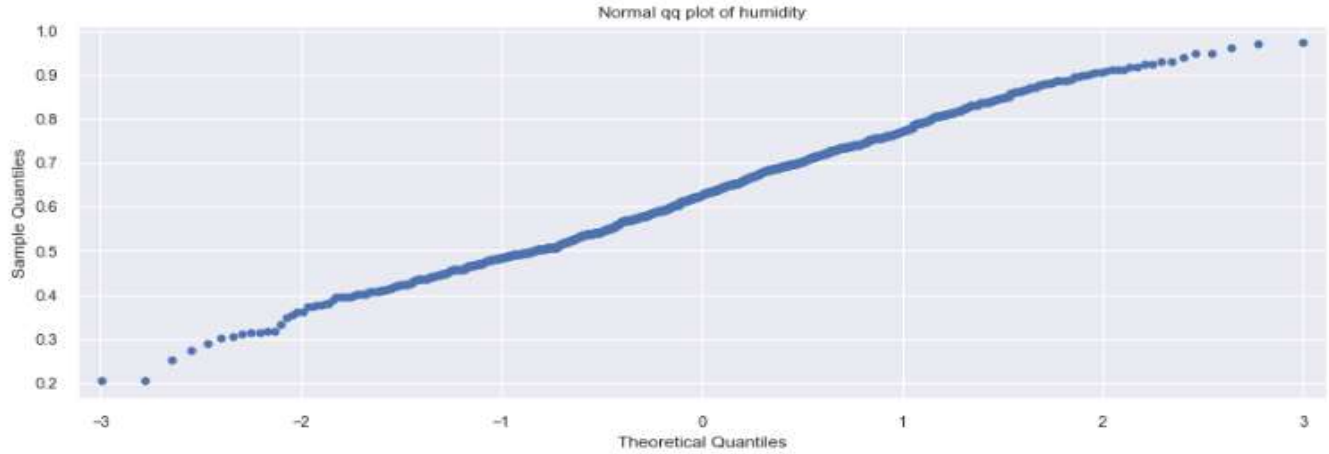
```
[1] "temperature"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.05913 0.33708 0.49833 0.49538 0.65542 0.86167
[1] "humidity"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.2047  0.5200  0.6267  0.6282  0.7302  0.9725
[1] "windspeed"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.02239 0.13495 0.18097 0.18985 0.23321 0.38061
[1] "count"
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     22    3152    4548    4504    5956    8714
```

In Python : Let's check the Normality of numeric variables

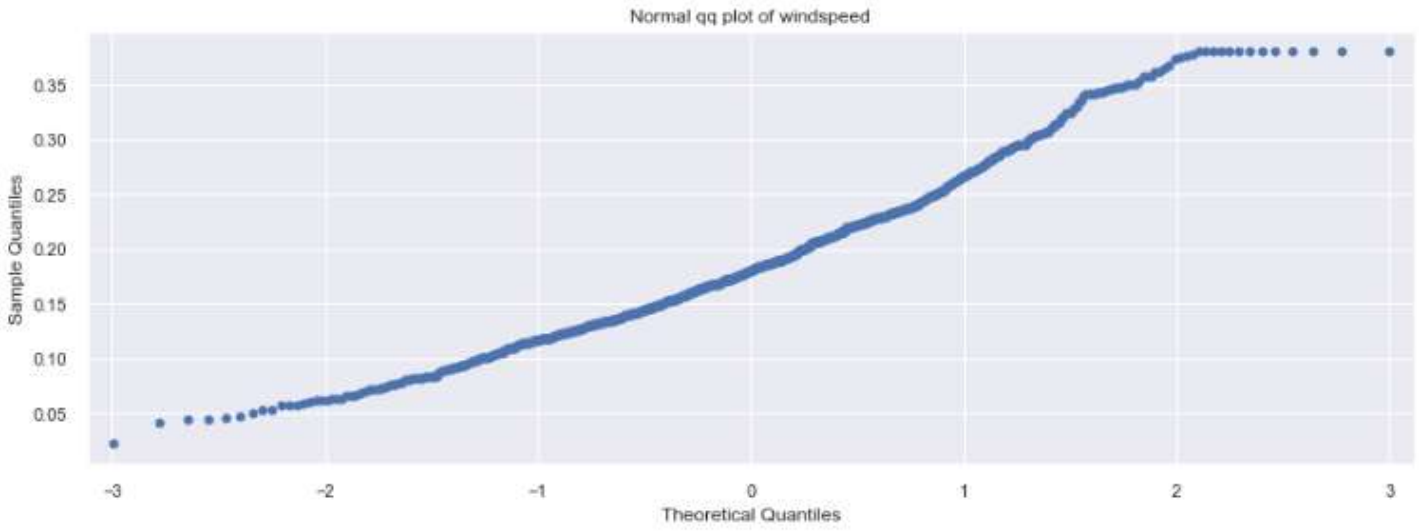
temperature



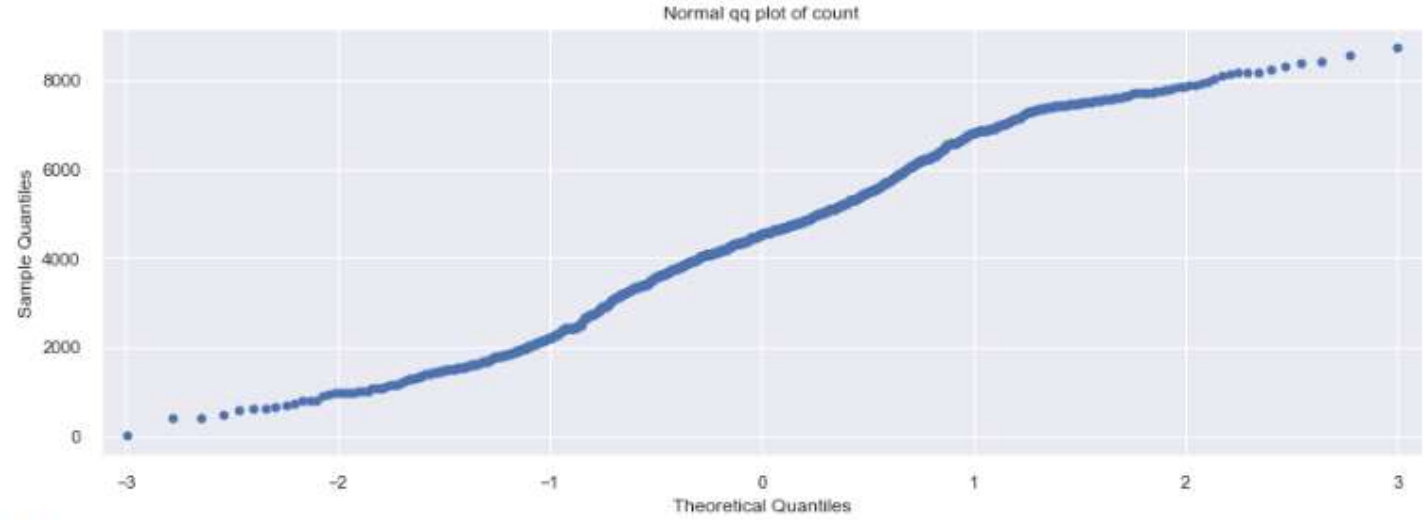
humidity



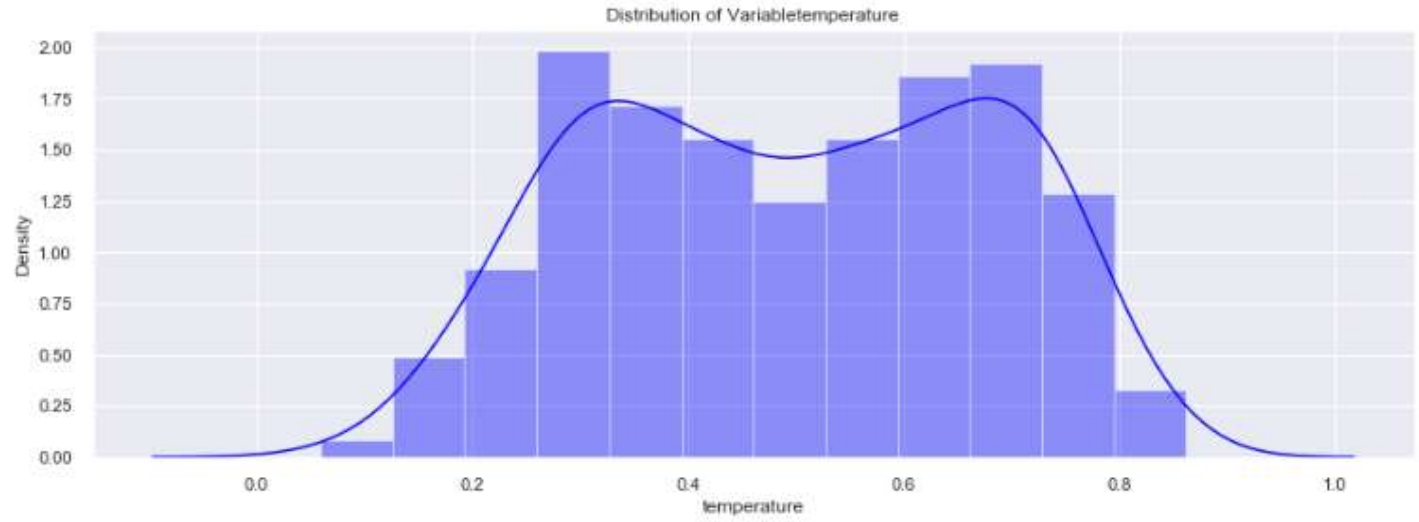
windspeed



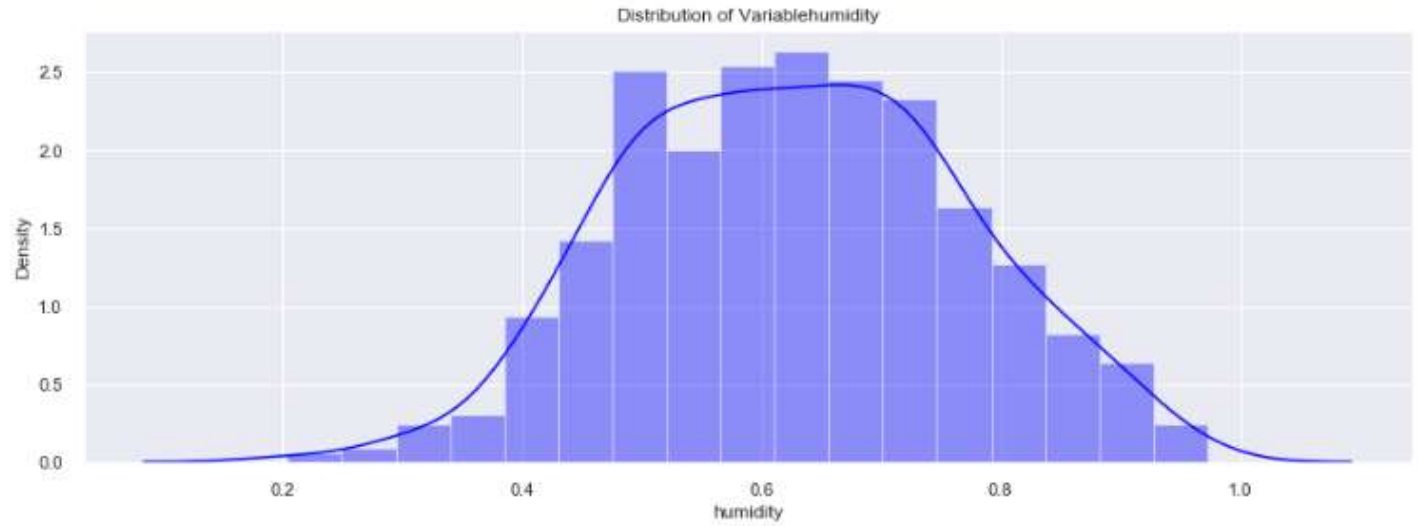
count



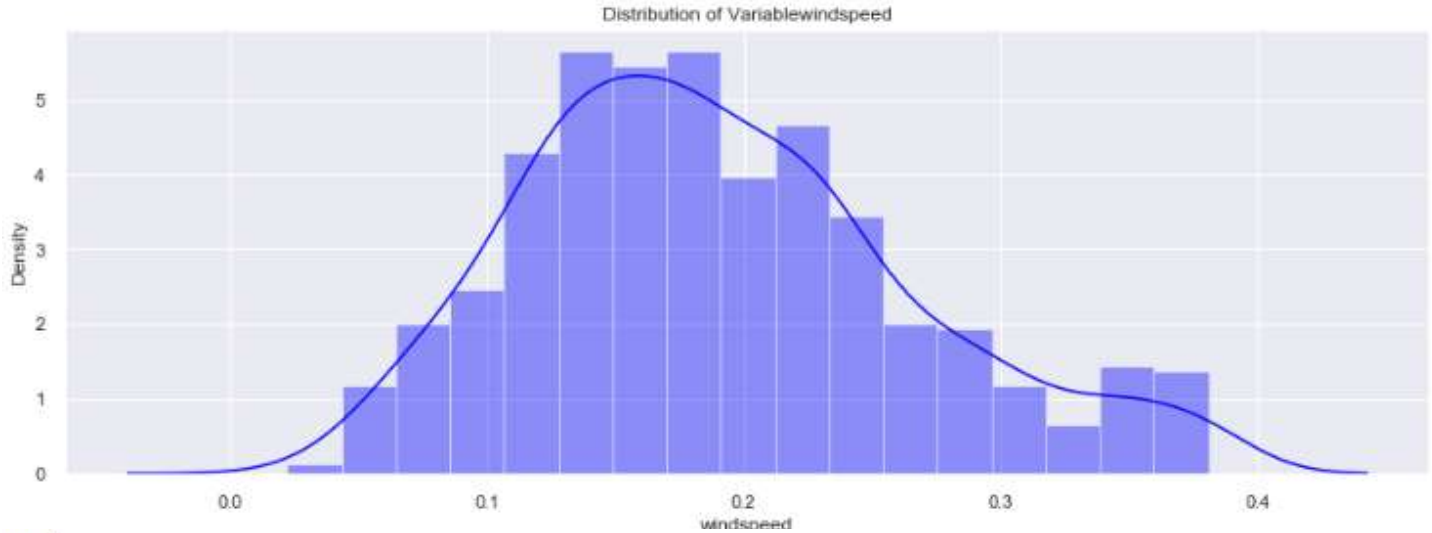
temperature



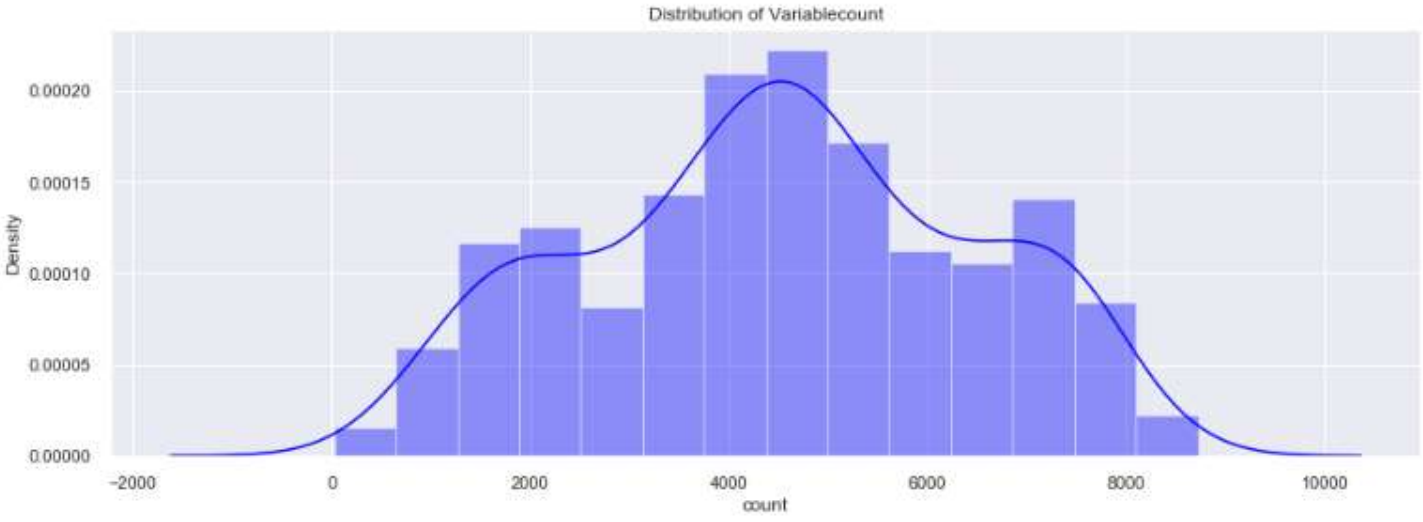
humidity



windspeed



count



Summary of each Variable:

	season	year	month	weather	temperature	humidity	windspeed	count
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.628197	0.189846	4504.348837
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.141320	0.075644	1937.211452
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.204687	0.022392	22.000000
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.520000	0.134950	3152.000000
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.626667	0.180975	4548.000000
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.233214	5956.000000
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.380611	8714.000000

2.5 Predictive Modeling:

Predictive modeling is a commonly used statistical technique to predict future behavior. Predictive modeling solutions are a form of data-mining technology that works by analyzing historical and current data and generating a model to help predict future outcomes. It's a technique that uses mathematical and computational methods to predict an event or outcome. A mathematical approach uses an equation-based model that describes the phenomenon under consideration. The model is used to forecast an outcome at some future state or time based upon changes to the model inputs. The model parameters help explain how model inputs influence the outcome.

Predictive modeling is a process that uses data mining and probability to forecast outcomes. Each model is made up of a number of predictors, which are variables that are likely to influence future results. Once data has been collected for relevant predictors, a statistical model is formulated. The model may employ a simple linear equation, or it may be a complex neural network, mapped out by sophisticated software. As additional data becomes available, the statistical analysis model is validated or revised.

2.5.1 Model Selection

It is the task of selecting a statistical model from a set of candidate models, given data. In the simplest cases, a pre-existing set of data is considered. However, the task can also involve the design of experiments such that the data collected is well-suited to the problem of model selection.

Once completing data cleaned next process is model selection it is based on problem statement. In bike rental count prediction problem statement understood that it comes under supervised machine learning because it has both input and output variables and its regression problem as out target variable is count which is of numeric / continuous type. So, we can consider linear regression, Decision Tree, Random Forest etc.,

In our project used linear regression, Decision Tree, Random Forest and Gradient Boosting models.
Error matrix chosen for the given problem statement is Root Mean Squared Error (RMSE) and R2(R-Squared). Before building an any model we divided the preprocessed Bike_Rent data set in to train and test set. Data was divided into 80:20 ratio, 80% of data was used as 'train' set and rest of the 20% was used as 'test' set. The training set is used to fit the model and the test set is used to estimate the model prediction accuracy.

2.5.2 Multiple Linear Regression

Multiple linear regression (MLR), also known simply as multiple regression, is a statistical technique that uses several explanatory variables to predict the outcome of a response variable. The goal of multiple linear regression (MLR) is to model the linear relationship between the explanatory (independent) variables and response (dependent) variable.

In essence, multiple regression is the extension of ordinary least-squares (OLS) regression that involves more than one explanatory variable.

The Formula for Multiple Linear Regression Is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_p x_{ip} + \epsilon$$

where, for $i = n$ observations:

y_i = dependent variable

x_i = expanatory variables

β_0 = y-intercept (constant term)

β_p = slope coefficients for each explanatory variable

ϵ = the model's error term (also known as the residuals)

Explaining Multiple Linear Regression

A simple linear regression is a function that allows an analyst or statistician to make predictions about one variable based on the information that is known about another variable. Linear regression can only be used when one has two continuous variables—an independent variable and a dependent variable. The independent variable is the parameter that is used to calculate the dependent variable or outcome. A multiple regression model extends to several explanatory variables.

The multiple regression model is based on the following assumptions:

- There is a linear relationship between the dependent variables and the independent variables.
- The independent variables are not too highly correlated with each other.
- Yi observations are selected independently and randomly from the population.
- Residuals should be normally distributed with a mean of 0 and variance σ.

The coefficient of determination (R-squared) is a statistical metric that is used to measure how much of the variation in outcome can be explained by the variation in the

independent variables. R2 always increases as more predictors are added to the MLR model even though the predictors may not be related to the outcome variable.

R2 by itself can't thus be used to identify which predictors should be included in a model and which should be excluded. R2 can only be between 0 and 1, where 0 indicates that the outcome cannot be predicted by any of the independent variables and 1 indicates that the outcome can be predicted without error from the independent variables.

Multiple Linear Regression model

```
Call:
lm(formula = count ~ ., data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-3881.4  -356.8   103.1   469.5  2842.4

Coefficients: (4 not defined because of singularities)
             Estimate Std. Error t value             Pr(>|t|)
(Intercept)  3706.55     410.20    9.036 < 0.0000000000000002 ***
season1     -1411.58     203.54   -6.935    0.000000000011121 ***
season2      -573.77     238.26   -2.408              0.016353 *
season3      -580.73     214.92   -2.702              0.007098 **
season4           NA         NA       NA                   NA
year0       -2068.81      66.37 -31.169 < 0.0000000000000002 ***
year1             NA         NA       NA                   NA
month1        -47.17     203.47   -0.232              0.816747
month2        -13.27     204.85   -0.065              0.948388
month3        565.19     208.14    2.715              0.006822 **
month4        377.46     272.90    1.383              0.167168
month5        661.70     289.24    2.288              0.022520 *
month6        435.48     293.69    1.483              0.138689
month7       -165.75     316.70   -0.523              0.600919
month8        272.05     304.70    0.893              0.372314
month9        931.53     248.16    3.754              0.000192 ***
month10       526.42     190.83    2.759              0.005993 **
month11       -82.85     179.59   -0.461              0.644757
month12           NA         NA       NA                   NA
weather1     1698.34     229.44    7.402    0.000000000000487 ***
weather2     1342.47     215.18    6.239    0.000000000864155 ***
weather3          NA         NA       NA                   NA
temperature  4793.81     460.47   10.411 < 0.0000000000000002 ***
humidity    -1744.93     335.33   -5.204    0.000000273625545 ***
windspeed   -3206.28     476.78   -6.725    0.000000000043071 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 786.2 on 566 degrees of freedom
Multiple R-squared:  0.8406,    Adjusted R-squared:  0.835
F-statistic: 149.3 on 20 and 566 DF,  p-value: < 0.00000000000000022
```

Let's check the assumptions of linear regression:

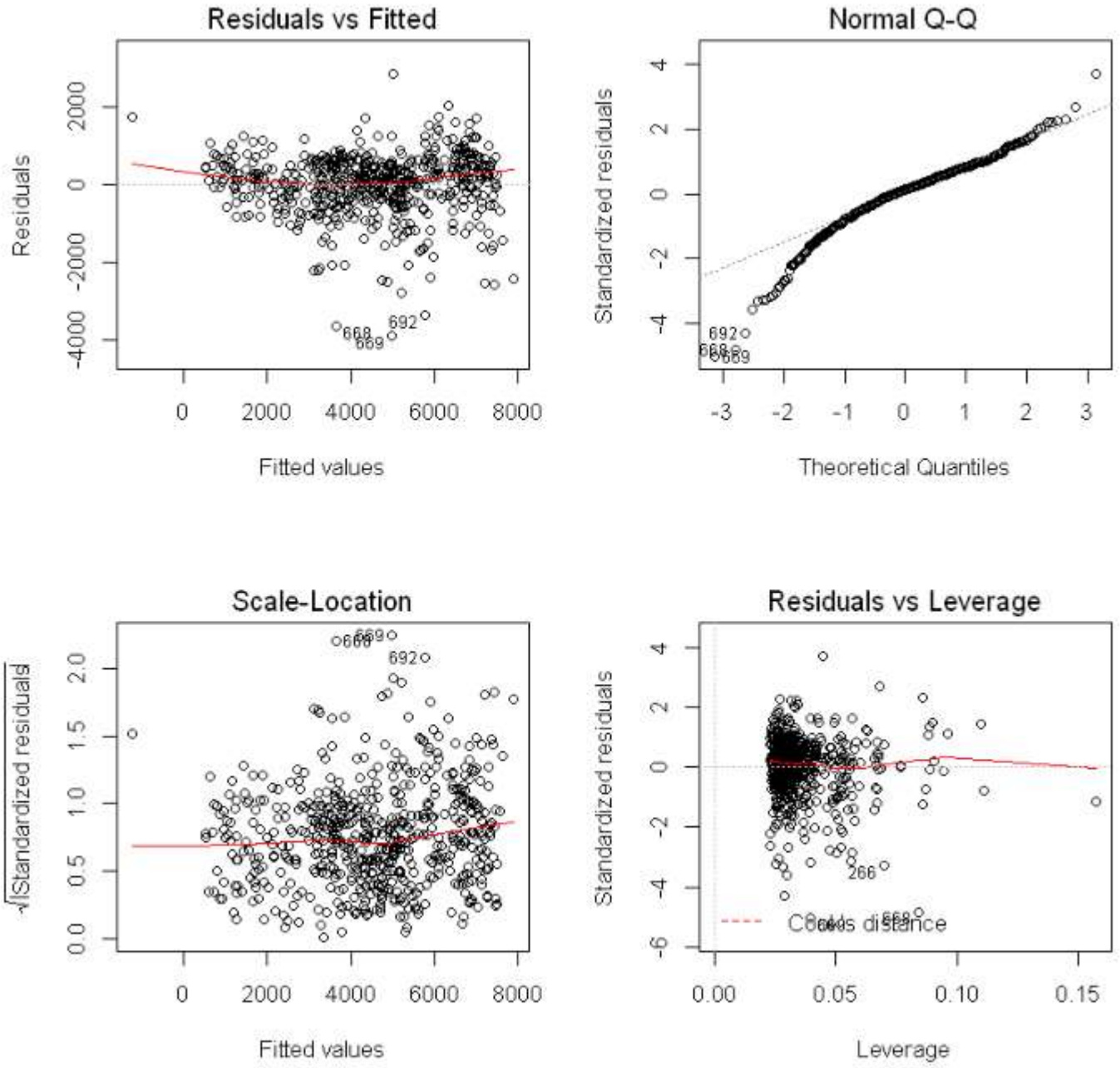
a) Error should follow normal distribution
b) Error should follow Homoscadacity or No Heteroscadacity
c) No multicollinearity among the independent variables
d) No serial / autocorrelation in error

Error should follow normal distribution:

To check this assumption, we plot normal QQ plot. The normal QQ plot helps us to determine if our target variable is normally distributed by plotting quantiles (i.e. percentiles) from our distribution against a theoretical distribution. If our data is normally distributed, it will be plotted in a generally straight line on the QQ plot. We can also plot histogram where error should show a curvy bell shape for our project this assumption satisfied.

Error should follow Homoscadacity or No Heteroscadacity :

To check this assumption, we can use residual plot Residual plots help us evaluate and improve our regression model. A residual is the difference between the observed value of the dependent variable (y) and the predicted value (ŷ). A "good" residual vs. fitted plot should be relatively shapeless without clear patterns in the data, no obvious outliers, and be generally symmetrically distributed. For our Project this assumption is violated (residuals following a pattern not scattered) we can see it in Residuals vs fitted plot.



No multicollinearity among the independent variables

To check this assumption, we are going to use Variance Inflation Factor (VIF) Variance inflation factor is a measure of the amount of multicollinearity in a set of multiple regression variables The Variance Inflation Factor (VIF) is 1/Tolerance, it is always greater than or equal to 1. There is no formal VIF value for determining presence of multicollinearity. Values of VIF that exceeds 10 are often regarded as indicating

multicollinearity, but in weaker models values above 2.5 may be a cause for concern for our project VIF values are within the range and this assumption is satisfied.

```
No variable from the 3 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( humidity ~ temperature ):  0.1267216
max correlation ( windspeed ~ humidity ):  -0.2411599

---------- VIFs of the remained variables --------
    Variables      VIF
1 temperature 1.034137
2    humidity 1.070959
3   windspeed 1.080362
```

No serial / autocorrelation in error:

Errors of all observation independent each other we can check this assumption using dwt test or Durbin Watson test The Durbin Watson (DW) statistic is a test for autocorrelation in the residuals from a statistical regression analysis. The Durbin - Watson statistic will always have a value between 0 and 4. A value of 2.0 means that there is no autocorrelation detected in the sample. Values from 0 to less than 2 indicate positive autocorrelation and values from 2 to 4 indicate negative autocorrelation. So, our model is fine with this assumption.

[46]:
```
#library(car)
dwt(LR_Model) # dwt < 2 so there is no autocorrelation in error
# All Asumptions of regression are satisfied
```

```
 lag Autocorrelation D-W Statistic p-value
   1       0.3507166      1.297814       0
 Alternative hypothesis: rho != 0
```

Using linear regression for our project we have got MAPE value about 19.48% which says our model is only 80.5% accurate. In python 83.79% accurate. it means linear regression is not performing well on our dataset. Lets check other Algorithms.

In R :

```
       RMSE    Rsquared         MAE
808.3312216   0.8309003 625.7650971
[1] "RMSE: 808.331221611972 R2: 0.830900349608336"
[1] "MAPE: 19.4865626957287"
[1] "LR_Accuracy: 80.5134373042713"
```

In Python:

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  count   R-squared:                       0.837
Model:                            OLS   Adj. R-squared:                  0.832
Method:                 Least Squares   F-statistic:                     145.0
Date:                Sun, 06 Oct 2019   Prob (F-statistic):          6.08e-207
Time:                        02:59:20   Log-Likelihood:                -4727.1
No. Observations:                 584   AIC:                             9496.
Df Residuals:                     563   BIC:                             9588.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
temperature  4578.6940    480.449      9.530      0.000    3635.002    5522.386
humidity    -1685.6382    357.064     -4.721      0.000   -2386.979    -984.298
windspeed   -2970.5399    490.617     -6.055      0.000   -3934.203   -2006.876
season_1     -128.0186    153.201     -0.836      0.404    -428.933     172.896
season_2      669.3727    155.814      4.296      0.000     363.324     975.421
season_3      653.7303    167.239      3.909      0.000     325.241     982.220
season_4     1545.9530    158.877      9.731      0.000    1233.889    1858.017
year_0        358.5551    153.565      2.335      0.020      56.925     660.185
year_1       2382.4824    153.609     15.510      0.000    2080.765    2684.200
month_1      -116.5131    195.949     -0.595      0.552    -501.394     268.367
month_2         5.2099    186.757      0.028      0.978    -361.615     372.035
month_3       519.1218    145.227      3.575      0.000     233.869     804.374
month_4       408.9712    179.140      2.283      0.023      57.107     760.835
month_5       721.9378    189.672      3.806      0.000     349.387    1094.488
month_6       431.4692    182.041      2.370      0.018      73.908     789.031
month_7       -99.1398    213.651     -0.464      0.643    -518.791     320.511
month_8       307.6803    205.925      1.494      0.136     -96.795     712.156
month_9       995.3312    162.837      6.112      0.000     675.489    1315.173
month_10      316.0378    184.794      1.710      0.088     -46.931     679.007
month_11     -463.6552    196.922     -2.355      0.019    -850.446     -76.864
month_12     -285.4136    163.535     -1.745      0.081    -606.627      35.800
weather_1    1768.3029     97.768     18.087      0.000    1576.268    1960.338
weather_2    1310.1542    115.351     11.358      0.000    1083.584    1536.724
weather_3    -337.4196    232.191     -1.453      0.147    -793.486     118.647
==============================================================================
Omnibus:                       86.992   Durbin-Watson:                   2.091
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              203.182
Skew:                          -0.793   Prob(JB):                     7.58e-45
Kurtosis:                       5.415   Cond. No.                     2.31e+16
==============================================================================

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.25e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.
```

```
Mean Absolute Precentage Error for train data=44.31264309612565
Mean Absolute Precentage Error for test data=16.20330573952452
R^2_score for train data=0.837413538323874
R^2_score for test data=0.8392349673915556
RMSE for train data=792.6189777902529
RMSE for test data=726.6869888055282
Accuracy :=83.79669426047548
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Linear Regression	83.796694	44.312643	16.203306	0.837414	0.839235	792.618978	726.686989

2.5.3 Decision Tree

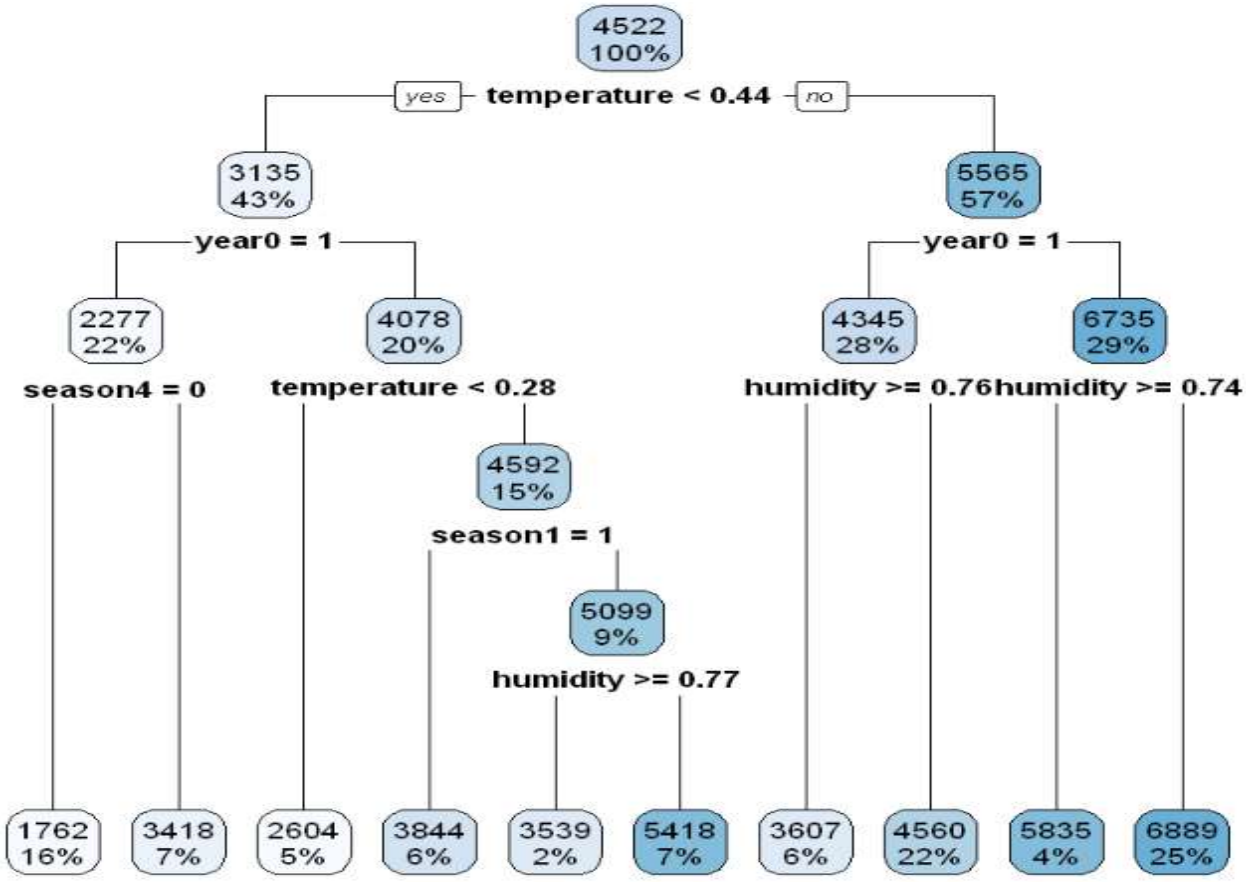
Decision Tree is a supervised machine learning algorithm, which is used to predict the data for classification and regression. It accepts both continuous and categorical variables. A decision tree is a decision support tool that uses a tree like graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Each branch connects nodes with “and” and multiple branches are connected by “or”. Extremely easy to understand by the business users. It provides the output in the form of rule, which can easily understand by a non - technical person also.

Output of Decision tree regression model is as below

```
n= 587

node), split, n, deviance, yval
      * denotes terminal node

 1) root 587 2195203000 4521.520
   2) temperature< 0.4420835 252  605469200 3134.583
     4) year0>=0.5 132  134604100 2276.826
       8) season4< 0.5 91   34976720 1762.484 *
       9) season4>=0.5 41   22121290 3418.415 *
     5) year0< 0.5 120  266915800 4078.117
      10) temperature< 0.2804165 31   26972130 2603.935 *
      11) temperature>=0.2804165 89  149108300 4591.596
        22) season1>=0.5 36   22877040 3844.083 *
        23) season1< 0.5 53   92451800 5099.340
          46) humidity>=0.765417 9   28767260 3539.111 *
          47) humidity< 0.765417 44   37294380 5418.477 *
   3) temperature>=0.4420835 335  740344400 5564.827
     6) year0>=0.5 164  104758000 4345.152
      12) humidity>=0.7577085 37   34347260 3607.297 *
      13) humidity< 0.7577085 127   44398180 4560.118 *
     7) year0< 0.5 171  157638600 6734.573
      14) humidity>=0.7427085 25   29128150 5835.400 *
      15) humidity< 0.7427085 146  104836600 6888.541 *
[1] 810.6184
[1] 934.6684
```



Using Decision tree to predict fare amount we got MAPE value is around 24.57% which means our model is 75.42% accurate . In python 73.29% accurate. Lets Cross validate.

In R:

```
[1] "RMSE: 934.668378902975 R2: 0.773816684616859"
[1] "MAPE: 24.5780826672469"
[1] "DT_Accuracy: 75.4219173327531"
```

In Python:

```
Mean Absolute Precentage Error for train data=80.87415784352679
Mean Absolute Precentage Error for test data=26.705318095966323
R^2_score for train data=0.6853648861225012
R^2_score for test data=0.6659645942279593
RMSE for train data=1102.62001385741
RMSE for test data=1047.4850631844804
Accuracy :=73.29468190403368
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Decision Tree	73.294682	80.874158	26.705318	0.685365	0.665965	1102.620014	1047.485063

(A) Random Search CV in Decision Tree

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was maxdepth = 9.
  maxdepth
9        9

[1] "RMSE: 934.668378902975 R2: 0.773816684616859"
[1] "MAPE: 24.5780826672469"
[1] "RDT_Accuracy: 75.4219173327531"
```

In Python:

```
Best Parameter={'max_depth': 7}
Best Model=DecisionTreeRegressor(criterion='mse', max_depth=7, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
Mean Absolute Precentage Error for train data=8.694353075329893
Mean Absolute Precentage Error for test data=17.195062652015018
R^2_score for train data=0.9346123894573801
R^2_score for test data=0.7333874133153626
RMSE for train data=502.6550038605767
RMSE for test data=935.8191154926612
Accuracy :=82.80493734798497
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Decision Tree Random Search CV	82.804937	8.694353	17.195063	0.934612	0.733387	502.655004	935.819115

(B) Grid Search CV in Decision Tree

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was maxdepth = 9.
  maxdepth
4        9

[1] "RMSE: 934.668378902975 R2: 0.773816684616859"
[1] "MAPE: 24.5780826672469"
[1] "GDT_Accuracy: 75.4219173327531"
```

In Python:

```
Best Parameter={'max_depth': 5}
Best Model=DecisionTreeRegressor(criterion='mse', max_depth=5, max_features=None,
                      max_leaf_nodes=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      presort=False, random_state=0, splitter='best')
Mean Absolute Precentage Error for train data=15.228885896748395
Mean Absolute Precentage Error for test data=16.59045335002796
R^2_score for train data=0.8704370348318436
R^2_score for test data=0.851354012671665
RMSE for train data=707.5590920557717
RMSE for test data=698.7602359923033
Accuracy :=83.40954664997204
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Decision Tree Grid Search CV	83.409547	15.228886	16.590453	0.870437	0.851354	707.559092	698.760236

2.5.4 Random forest

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build N number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly N no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data. The higher no of trees in the random forest will give higher no of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important. The Number of trees used in R and Python for random forest model are 200 no's

Using Random Forest model, MAPE value is around 16.73% which means our model is 83.26% accurate in R and 87.12%in Python. Lets Cross validate the same.

In R :

```
[1] "RMSE: 689.335163843773 R2: 0.876268828418663"
[1] "MAPE: 16.7324368079997"
[1] "RF_Accuracy: 83.2675631920003"
```

In Python:

```
Mean Absolute Precentage Error for train data=14.188417932620666
Mean Absolute Precentage Error for test data=12.877322509006008
R^2_score for train data=0.9811670474727333
R^2_score for test data=0.9108417399030518
RMSE for train data=269.7624408147195
RMSE for test data=541.1683388561013
Accuracy :=87.122677490994
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Random Forest	87.122677	14.188418	12.877323	0.981167	0.910842	269.762441	541.168339

(A) Random Search CV in Random Forest

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 12.
  mtry
2   12

[1] "RMSE: 691.266683327514 R2: 0.875225300207126"
[1] "MAPE: 17.0053345068384"
[1] "RRF_Accuracy: 82.9946654931616"
```

In Python:

```
Best Parameter={'n_estimators': 43, 'max_depth': 7}
Best Model=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=7,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=43,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
Mean Absolute Precentage Error for train data=21.44499602330166
Mean Absolute Precentage Error for test data=13.60794720295829
R^2_score for train data=0.9455337806472509
R^2_score for test data=0.9030014697454434
RMSE for train data=458.7603675505254
RMSE for test data=564.4612910050469
Accuracy :=86.39205279704171
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Random Forest Random Search CV	86.392053	21.444996	13.607947	0.945534	0.903001	458.760368	564.461291

(B) Grid Search CV in Random Forest

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final value used for the model was mtry = 9.
  mtry
4    9
```

```
[1] "RMSE: 687.786392700359 R2: 0.877046879979211"
[1] "MAPE: 16.9049916971439"
[1] "GRF_Accuracy: 83.0950083028561"
```

In Python:

```
Best Parameter={'n_estimators': 43, 'max_depth': 7}
Best Model=RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=7,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=11,
                      n_jobs=None, oob_score=False, random_state=0, verbose=0,
                      warm_start=False)
Mean Absolute Precentage Error for train data=20.813934760410895
Mean Absolute Precentage Error for test data=13.955204920480801
R^2_score for train data=0.9412924308902957
R^2_score for test data=0.8924388712911968
RMSE for train data=476.28765560933397
RMSE for test data=594.4006330494107
Accuracy :=86.0447950795192
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Random Forest Grid Search CV	86.044795	20.813935	13.955205	0.941292	0.892439	476.287656	594.400633

2.5.5 Gradient Boosting

Gradient boosting is a machine learning technique for regression and classification problems, It produces a prediction model in the form of an ensemble of weak learner models and produce a strong learner with less misclassification and higher accuracy. It feed the error from one decision tree to another decision tree and generates a strong classifier or Regressor

Using Gradient Boosting model, MAPE value is around 16.81% which means our model is 83.18% accurate in R and 87.26%in Python. Lets Cross validate the same.

In R:

```
[1] "RMSE: 730.654150085469 R2: 0.85934201909568"
[1] "MAPE: 16.8172465673327"
[1] "GB_Accuracy: 83.1827534326673"
```

In Python:

```
Mean Absolute Precentage Error for train data=13.956539271371224
Mean Absolute Precentage Error for test data=12.738142439093943
R^2_score for train data=0.9436481774759837
R^2_score for test data=0.9068019129082193
RMSE for train data=466.6338717157675
RMSE for test data=553.2928870013445
Accuracy :=87.26185756090605
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Gradient Boosting	87.261858	13.956539	12.738142	0.943648	0.906802	466.633872	553.292887

(A) Random Search CV in Gradient Boosting

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 4347, interaction.depth =
 3, shrinkage = 0.02560172 and n.minobsinnode = 5.
  n.trees interaction.depth  shrinkage n.minobsinnode
2    4347                 3 0.02560172              5
[1] "RMSE: 687.786392700359 R2: 0.877472086322398"
[1] "MAPE: 16.9351833585913"
[1] "RGB_Accuracy: 83.0648166414087"
```

In Python:

```
Best Parameter={'n_estimators': 81, 'max_depth': 5}
Best Model=GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=81,
                          n_iter_no_change=None, presort='auto', random_state=0,
                          subsample=1.0, tol=0.0001, validation_fraction=0.1,
                          verbose=0, warm_start=False)
Mean Absolute Precentage Error for train data=5.768812904940644
Mean Absolute Precentage Error for test data=12.881924358472062
R^2_score for train data=0.9846272603812205
R^2_score for test data=0.9071666433060924
RMSE for train data=792.6189777902529
RMSE for test data=726.6869888055282
Accuracy :=87.11807564152794
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Gradient Boosting Random Search CV	87.118076	5.768813	12.881924	0.984627	0.907167	792.618978	726.686989

(B) Grid Search CV in Gradient Boosting

In R:

```
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were n.trees = 4349, interaction.depth =
 2, shrinkage = 0.01 and n.minobsinnode = 2.
  n.trees interaction.depth shrinkage n.minobsinnode
2    4349                 2      0.01              2

[1] "RMSE: 680.22095501262 R2: 0.879823468817697"
[1] "MAPE: 16.7521902013304"
[1] "GGB_Accuracy: 83.2478097986696"
```

In Python:

```
Best Parameter={'max_depth': 5, 'n_estimators': 19}
Best Model=GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=19,
                          n_iter_no_change=None, presort='auto', random_state=0,
                          subsample=1.0, tol=0.0001, validation_fraction=0.1,
                          verbose=0, warm_start=False)
Mean Absolute Precentage Error for train data=20.461529580692606
Mean Absolute Precentage Error for test data=15.445265768969858
R^2_score for train data=0.9204600413607151
R^2_score for test data=0.8888399403361145
RMSE for train data=554.3894499460075
RMSE for test data=604.26295954657
Accuracy :=84.55473423103014
```

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Gradient Boosting Grid Search CV	84.554734	20.46153	15.445266	0.92046	0.88884	554.38945	604.26296

3. Conclusion:

3.1 MAE, MSE, RMSE, R-squared, MAPE

We have calculated RMSE, R-Square, MAE (Mean Absolute Error) for all the three models.

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors) Residuals are a measure of how far from the regression line data points are,

RMSE is a measure of how spread out these residuals are. In other words, it tells you how concentrated the data is around the line of best fit. Whereas R squared is a relative measure of fit, RMSE is an absolute measure of fit. As the square root of a variance, RMSE can be interpreted as the standard deviation of the unexplained variance and has the useful property of being in the same units as the response variable. Lower values of RMSE indicate better fit.

R- squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-Squared tells how much variance of dependent variable explained by the independent variable. It is a measure if goodness of fit in regression line. We also said about MAPE value for all the Four models which is a measure of prediction accuracy of a forecasting method. It measures accuracy in terms of percentage

3.2 Hyper parameter tuning:

In statistics, hyperparameter is a parameter from a prior distribution; it captures the prior belief before data is observed. In any machine learning algorithm, these parameters need to be initialized before training a model. Choosing appropriate hyperparameters plays a crucial role in the success of good model. Since it makes a huge impact on the learned model. For example, if the learning rate is too low, the model will miss the important patterns in the data. If it is high, it may have collisions.

We used two techniques of Hyperparameter for our models :

- **Random Search**
- **Grid Search**

Random Search :

Random search is a technique where random combinations of the hyperparameters are used to find the best solution for the built model. In this search pattern, random combinations of parameters are considered in every iteration. The chances of finding the optimal parameter are comparatively higher in random search because of the

random search pattern where the model might end up being trained on the optimised parameters without any aliasing.

Grid Search :

Grid search is a technique which tends to find the right set of hyperparameters for the particular model. Hyperparameters are not the model parameters and it is not possible to find the best set from the training data. Model parameters are learned during training when we optimize a loss function using something like a gradient descent. In this tuning technique, we simply build a model for every combination of various hyperparameters and evaluate each model. The model which gives the highest accuracy wins. The pattern followed here is similar to the grid, where all the values are placed in the form of a matrix. Each set of parameters is taken into consideration and the accuracy is noted. Once all the combinations are evaluated, the model with the set of parameters which give the top accuracy is considered to be the best.

We used these two techniques for Decision tree, Random Forest, Gradient boosting models.

3.3 Selected Model Results:

Predicted Bike Rental count using Random Forest Model (83.26%) in R.

In R:

Model	Test_Accuracy	MAPE_Train	MAPE_Test	Rsquare_Train	Rsquare_Test	Rmse_Train	Rmse_Test
Linear Regression	80.51344	45.85389	19.48656	0.8406345	0.8309003	771.9967	808.3312
Decision Tree for Regression	75.42192	47.76958	24.57808	0.8242901	0.7738167	810.6184	934.6684
Random Search in Decision Tree	75.42192	47.76958	24.57808	0.8242901	0.7738167	810.6184	934.6684
Gird Search in Decision Tree	75.42192	47.76958	24.57808	0.8242901	0.7738167	810.6184	934.6684
Random Forest	83.26756	25.52668	16.73244	0.9672237	0.8762688	361.3621	689.3352
Random Search in Random Forest	82.99467	24.66052	17.00533	0.9733249	0.8752253	325.3256	691.2667
Grid Search in Random Forest	83.09501	25.35598	16.90499	0.9697742	0.8770469	346.8116	687.7864
Gradient Boosting	83.18275	33.91118	16.81725	0.9054856	0.8593420	595.6798	730.6542
Random Search in Gradient Boosting	83.06482	25.73804	16.93518	0.9679926	0.8774721	346.8116	687.7864
Grid Search in Gradient Boosting	83.24781	25.83045	16.75219	0.9679843	0.8798235	357.5077	680.2210

	Predicted_Bike_Rental_Count	Actual_Bike_Rental_Count
65	1747	605
341	2293	705
9	1285	822
478	3331	1027
28	1170	1167
11	1250	1263
10	1312	1321
729	2234	1341
32	1591	1360
13	1313	1406
59	1947	1446
33	1495	1526
34	1660	1550
5	1340	1600
40	1575	1605
326	2554	1607
37	1785	1623
722	2716	1749
730	2678	1796
55	1509	1807
51	1796	1812
250	3011	1996
102	3533	2034

In Python:

Predicted Bike Rental count using Gradient Boosting Model (87.26%) in Python

	Model Name	Accuracy	MAPE_Train	MAPE_Test	R-squared_Train	R-squared_Test	RMSE_train	RMSE_test
0	Linear Regression	83.796694	44.312643	16.203306	0.837414	0.839235	792.618978	726.686989
1	Decision Tree	73.294682	80.874158	26.705318	0.685365	0.665965	1102.620014	1047.485063
2	Decision Tree Random Search CV	82.804937	8.694353	17.195063	0.934612	0.733387	502.655004	935.819115
3	Decision Tree Grid Search CV	83.409547	15.228886	16.590453	0.870437	0.851354	707.559092	698.760236
4	Random Forest	87.122677	14.188418	12.877323	0.981167	0.910842	269.762441	541.168339
5	Random Forest Random Search CV	86.392053	21.444996	13.607947	0.945534	0.903001	458.760368	564.461291
6	Random Forest Grid Search CV	86.044795	20.813935	13.955205	0.941292	0.892439	476.287656	594.400633
7	Gradient Boosting	87.261858	13.956539	12.738142	0.943648	0.906802	466.633872	553.292887
8	Gradient Boosting Random Search CV	87.118076	5.768813	12.881924	0.984627	0.907167	792.618978	726.686989
9	Gradient Boosting Grid Search CV	84.554734	20.461530	15.445266	0.920460	0.888840	554.389450	604.262960

	Predicted_Bike_Rental_Count	Actual_Bike_Rental_Count
139	1665.0	605.0
113	1297.0	623.0
118	1387.0	1349.0
25	1448.0	1360.0
114	1282.0	1406.0
22	1671.0	1472.0
145	1404.0	1510.0
86	818.0	1526.0
82	1468.0	1550.0
88	1795.0	1623.0
107	656.0	1685.0
40	2400.0	1796.0
38	2053.0	1817.0
24	1819.0	1969.0
93	1566.0	1985.0
65	2061.0	2121.0
48	954.0	2169.0
42	1866.0	2192.0
70	2109.0	2236.0
11	1833.0	2302.0
35	2652.0	2311.0
72	2643.0	2368.0
74	3671.0	2395.0
44	2095.0	2424.0

4.1 Brief Insights about the Bike Rental Count prediction Project

From this bike rental count project, we analyzed and found below observations.

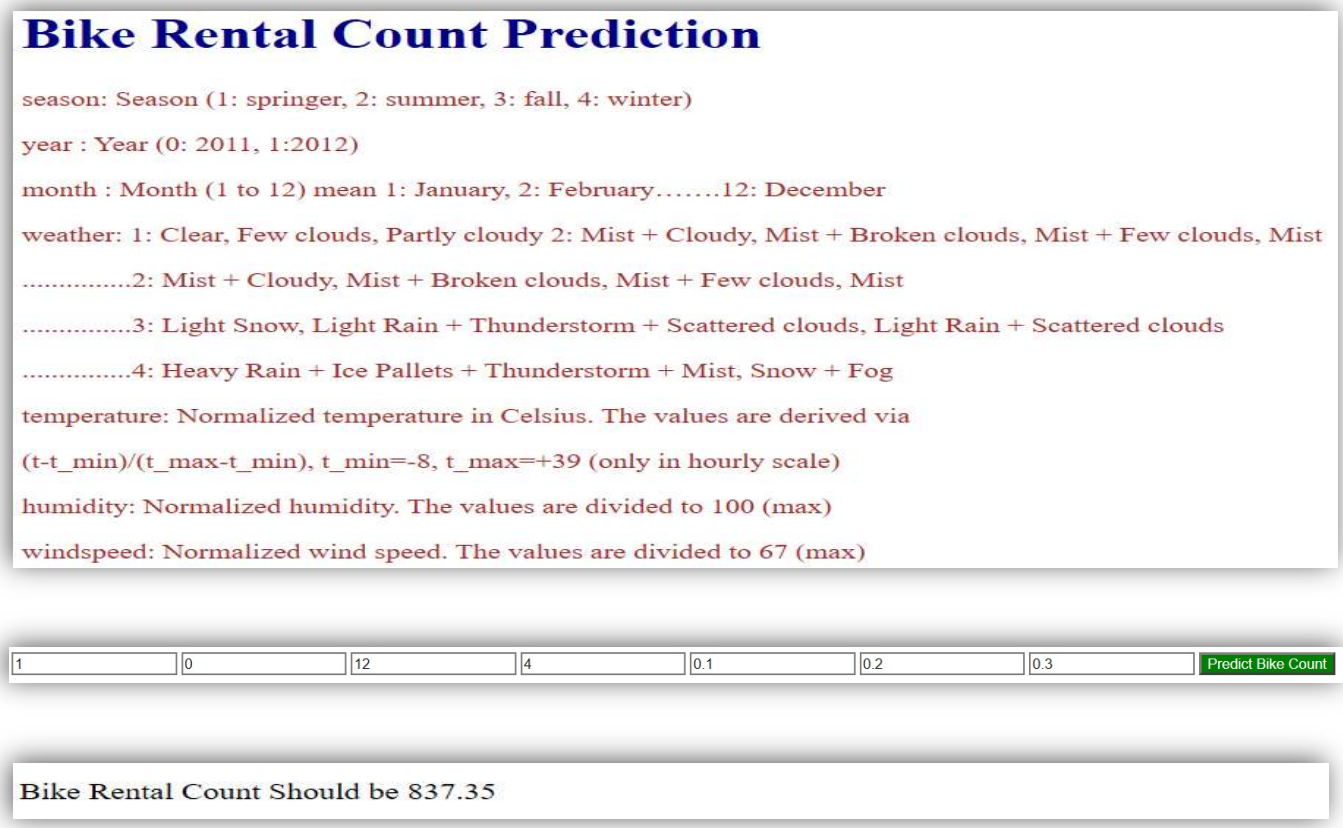
1) Temperature variable has major impact on bike rental count
2) In season 3 (Sept, Oct, Nov) there will be more bike rental counts
3) When the weather is 1 means when weather is Clear, Few clouds, Partly cloudy then there will be more bike rental counts
4) In the month of August bike rental count will be very high
5) Windspeed and humidity don't have any impact on bike rental count and these two Variables are negatively correlated with all variables
6) On Friday or fifth day of week bike rental count is more
7) Even during working days bike rental count is more

4.2 Model Deployed on cloud for Prediction:

Deployment of our machine learning model (Gradient Boosting) of bike rental count in python language using Flask at Heroku PAAS (Platform as a Service) environment.

Enter the Values in the space provided with set range values each field based on the input values model will predict the bike rental count. Global link to test the Bike Rental Count Prediction web APP
(Use Ctrl Key + click the link below to navigate it to web API)
https://bike-rental-count-api.herokuapp.com

Example Screenshot Shown Below:



Deployed code files like app.py, index.html, model.pkl file , requirements.txt files can be accessed on my Git hub repository as mentioned below
https://github.com/sanjaytallolli/Bike_Rent_Count_Deplyment

5 Appendix A – R and Python Scripts and Notebooks


Bike_Rental_Count_Python.py


Bike_Rental_Count_R.r


Bike_Rental_Count_Python - deploy.py


Bike_Rental_Count_Python.ipynb


Bike_Rental_Count_Rscript.ipynb


Bike_Rental_Count_Python - deploy.ipynb


Bike_Rental_Count_Python_Codefile.pdf


Bike Rental Problem Statement.pdf


Bike_Rental_Count_R_Codefile.pdf

5 Appendix B – References:

1. https://www.analyticsvidhya.com/blog/2017/09/machine-learning-models-as-apis-using-flask/
2. For Model deployment refer https://www.saedsayad.com/model_deployment.htm
3. https://www.analyticsvidhya.com/blog/2016/01/guide-data-exploration/
4. For Model evaluation refer https://www.saedsayad.com/model_evaluation.htm
5. https://www.youtube.com/watch?v=mrExsjcvF4o&list=PLZoTAELRMXVOAvUbePX1lTdxQR8EY35Z1&index=2 # Model Deployment Reference
6. https://trainingdatascience.com/workshops/histograms-and-skewed-data/
7. Reference Blog: https://machinelearningmastery.com/tune-machine-learning-algorithms-in-r/
8. For parameter Tuning https://uc-r.github.io/random_forestsp
9. http://www.knowru.com/blog/how-create-restful-api-for-machine-learning-credit-model-in-r/
10. https://github.com/trestletech/plumber