Prediction of Bike Rental Count

-By Rakshith R

19-08-2020

Contents

1. Introduction.....03
1.1 Problem Statement.....03
1.2 Data.....03
1.3 Exploratory Data Analysis.....04
1.4 Data Understanding.....05

2. Methodology.....08
2.1 Pre-Processing.....08
2.1.1 Missing Value analysis.....08
2.1.2 Outlier Analysis.....09
2.1.3 Feature Selection.....10
2.1.4 Feature Scaling 11
2.2 Model Development.....12
2.2.1 Model Selection.....12
2.2.1.1 Decision Tree.....12
2.2.1.2 Random Forest.....13
2.2.1.3 Linear Regression.....13

3. Evaluation of the Model.....14
3.1 Error Metrics.....14
3.1.1 MAPE.....15
3.1.2 Accuracy.....15
3.1.3 R Square.....15
3.2 Conclusion.....15

4. Appendix A – Python code.....16
5. Appendix B – R code.....31

Chapter 1: Introduction

1.1 Problem Statement

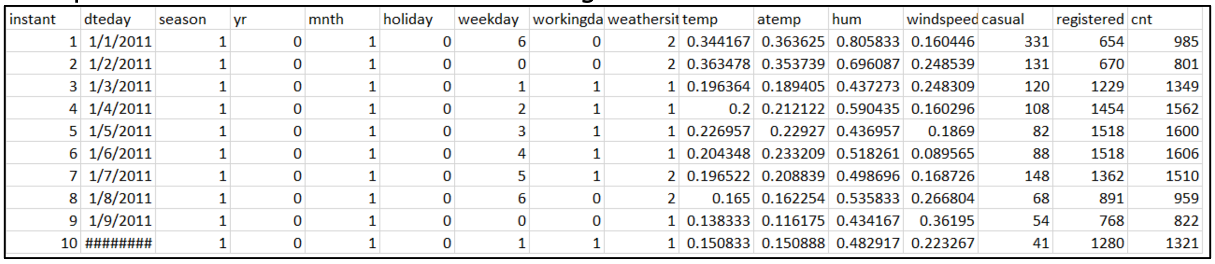
The project is about a bike rental company who has its historical data. The goal is to build a models which predicts the count of bike rentals based on the seasonal and environmental settings. These predicted values would help the business to meet the demand on those particular days by being prepared for high demand of bikes during peak periods.

1.2 DATA

The given dataset contains 16 variables and 731 observations. The "cnt" is the target variable and remaining all other variables being independent variables. The objective being to develop a model which can determine the count for future test cases. And this model can be developed by the help of given data.

The details of data attributes in the dataset are as follows:-
instant: Record index
dteday: Date
season: Season (1: springer, 2: summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted from Holiday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted from Freemeteo)
1: Clear, Few clouds, partly cloudy, 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_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

A snapshot of the data is mentioned following.



instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	1/4/2011	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
7	1/7/2011	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
8	1/8/2011	1	0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891	959
9	1/9/2011	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768	822
10	########	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

As you can see in the table below we have the following 13 variables, using which we have to correctly predict the count of bikes:

Sl.No	Variables
1	Instant
2	Dteday
3	Season
4	Yr
5	Month
6	Holiday
7	Weekday
8	Workingday
9	Weathersit
10	Temp
11	Atemp
12	Hum
13	windspeed

1.3 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach used to analysing data sets to summarize the main characteristics. In the given data set there are 16 variables and data types of all variables are object, float64 or int64.

```
instant          int64
dteday          object
season           int64
yr               int64
mnth             int64
holiday          int64
weekday          int64
workingday       int64
weathersit       int64
temp           float64
atemp          float64
hum            float64
windspeed      float64
casual           int64
registered       int64
cnt              int64
dtype: object
```

There are 731 observations and 16 columns in our data set. From EDA we have observed that there are 9 categorical variable and 7 continuous variable in nature.

From EDA we have checked the number of unique values in each variables.

```
instant        731
dteday         731
season           4
yr               2
mnth            12
holiday          2
```

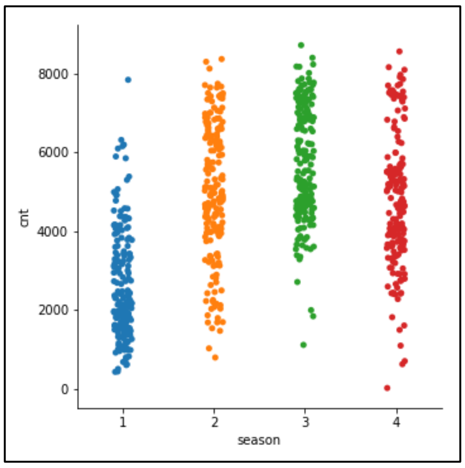
```
weekday          7
workingday       2
weathersit       3
temp           499
atemp          690
hum            595
windspeed      650
casual         606
registered     679
cnt            696
dtype: int64
```

During EDA we observed that few of variables are not important for proceed further as these are irrelevant variable in our dataset. Hence we drop them before processing the data.
We dropped variable 'instant' as it is just an index in our dataset. Similarly we dropped 'dteday' variable since our output is not Time-Series data analysis. Also, 'casual' and 'registered' variables can be removed, as these two sums to dependent variable 'count' and which is what we need to predict.

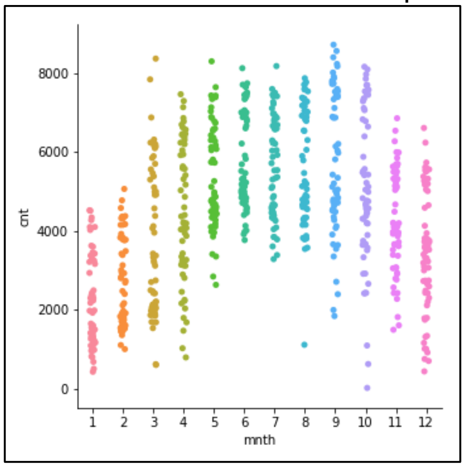
1.4 Data Understanding

For better understanding of data, here we have plotted some visualization for the variables.
Data Understand is a process wherein we get know our data in a better way by the help of visual representations and come up with initial ideas for developing our model. Here, the specific variables are plotted with respect to the target variable. In some cases two variables are compared, whereas in some cases three variables are plotted together for our better understanding and visualization.

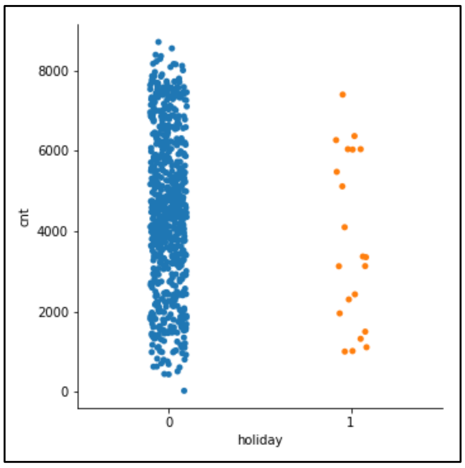
1. From season plot, we can see that season 2, 3 and 4 have more bike count as compare to season 1.



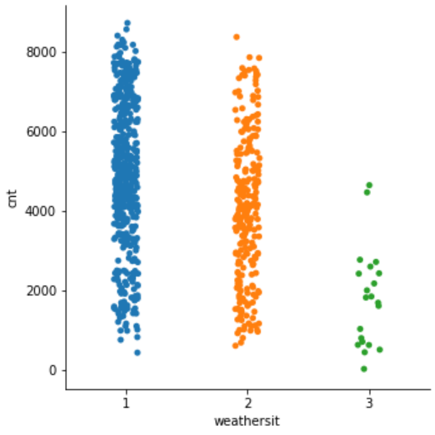
2. Below plot is for month wise count of bikes, so this tells us that the bike counts are higher between month 3 to month 10 as compare to other months



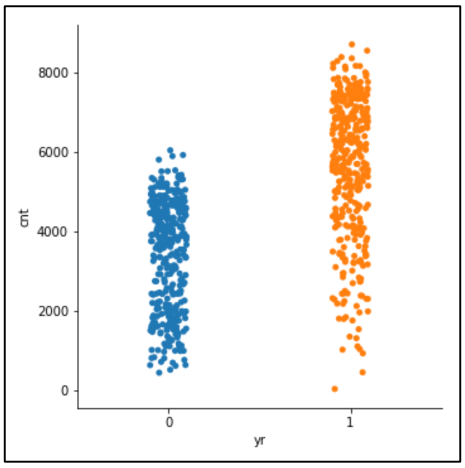
3. Below Plot is between holiday and count, from this plot we can clearly say on holidays the count is higher when compared non-holidays



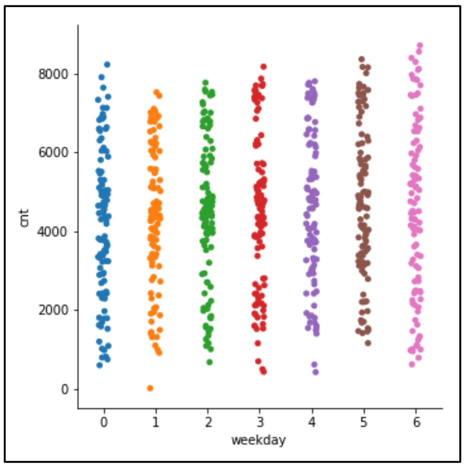
4. In weather-1 the count of bikes is good as compare to other weather



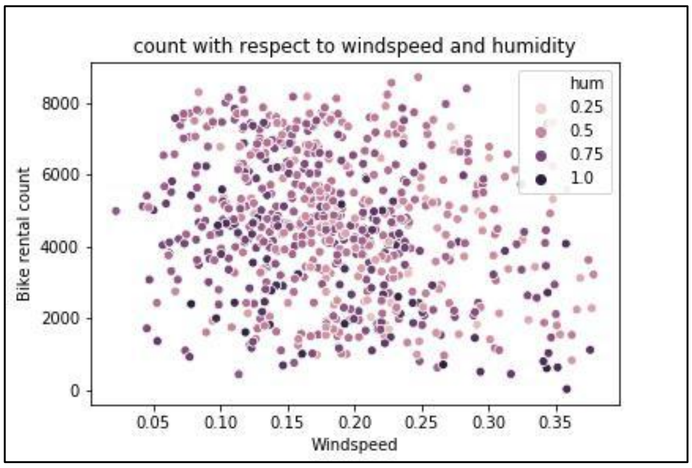
5. Here, it is found that in Year 1 has high count than 0



6. Here, it is observed that in weekdays, 0 and 6 i.e. Monday to Saturday the count is highest.



7. The below plot is of Windspeed and Humidity vs count. Here, it is found that in count vs windspeed and humidity, Count is High in ranges of windspeed 0.10 to 0.25 and humidity 0.5 to 0.75



Chapter 2: Methodology

Methodology mainly consists of following processes,

1. Pre-processing:
   It includes missing value analysis, outlier analysis, feature selection and feature scaling.
2. Model development:
   It includes identifying suitable Machine learning Algorithms and applying those algorithms in our given dataset.

2.1 Pre-Processing

Data pre-processing is the first stage of any type of project. In this stage we get the feel of the data. A predictive model requires that we look at the data before we start to create a model. We do this by looking at plots of independent variables vs target variables. If the data is messy, we try to improve it by sorting deleting extra rows and columns. This stage is called as Exploratory Data Analysis. This stage generally involves data cleaning, merging, sorting, looking for outlier analysis, looking for missing values in the data, Imputing missing values if found by various methods such as mean, median, mode, KNN imputation, etc.

Further we will look into what pre-processing steps do this project was involved in.

2.1.1 Missing value Analysis

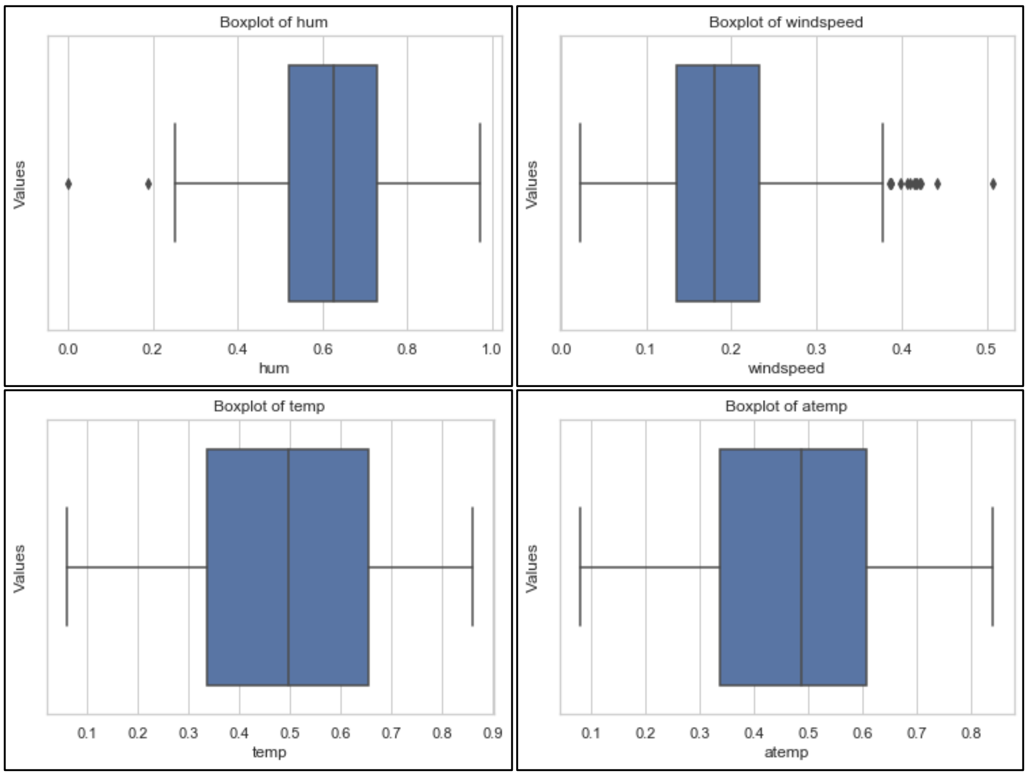
In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. If a columns has more than 30% of data as missing value either we ignore the entire column or we ignore those observations.
In this step we look for missing values in the dataset like empty row column cell which was left after removing special characters and punctuation marks or because of reasons like, incomplete submission, wrong input, manual error etc. Some missing values are in form of NA or Missing values left behind after outlier analysis; missing values can be in any form. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.
In the given data there is no any missing value. So we do not need to impute missing values.
Below table illustrate that there is no missing value present in the data.

```
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

2.1.2 Outlier Analysis

One of the other steps of pre-processing is to check the presence of outliers. Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error, poor quality of data and it is correct but exceptional data. This will create an error in predicting the target variables and can hamper our data model.. Here to check the outlier in our dataset, we used a classic approach to visualize outliers, which is Boxplot Method.

Below is the plots of box plots of variables,



In this project, outliers are found in only two variables this are Humidity and windspeed. Dots outside the quartile ranges are outliers.
Outliers can be removed using the Boxplot stats method, wherein the Inter Quartile Range (IQR) is calculated and the minimum and maximum value are calculated for the variables. Any value ranging outside the minimum and maximum value are discarded.

```
season         0
yr             0
mnth           0
holiday        0
weekday        0
workingday     0
weathersit     0
temp           0
atemp          0
hum            2
windspeed     13
cnt            0
dtype: int64
```

We can observe from above table that there are 13 outliers present in 'windspeed' variable an d 2 in 'humidity' variable. In this project, I used median method to impute the outliers in wind speed and humidity variables.

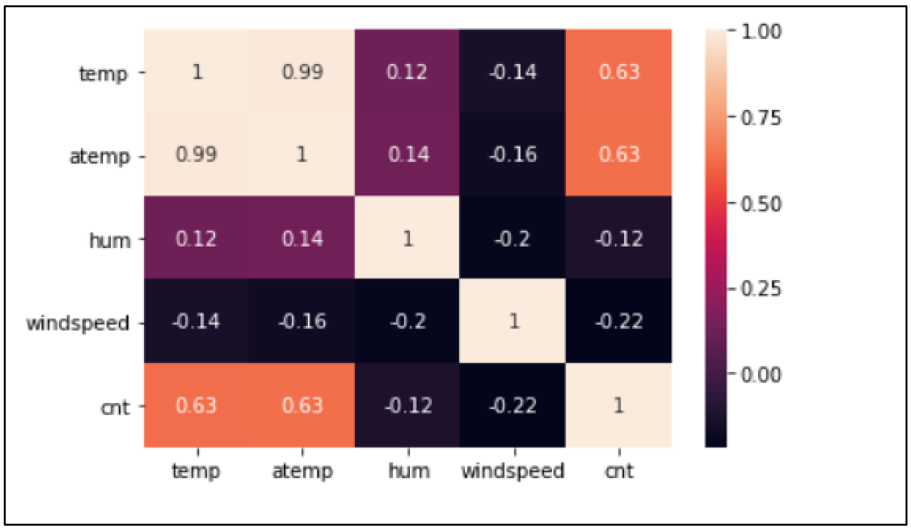
2.1.3 Feature Selection

Feature Selection is used to reduce the complexity of a model and make it easier to interpret. It even reduces overfitting. Sometimes in our data, all the variables may not be accurate enough or required to predict the target variable, in such cases we need to analyse our data, understand the data and select the variables which are most useful for our model. In such cases we apply feature selection. Feature selection helps in reducing computational time of model.

In this project we have selected Correlation Analysis for numerical variable and ANOVA (Analysis of variance) for categorical variables to check if there is collinearity among the variable. And if there is any collinearity it's better to drop such variables, else this redundant variables may hamper the accuracy of model.

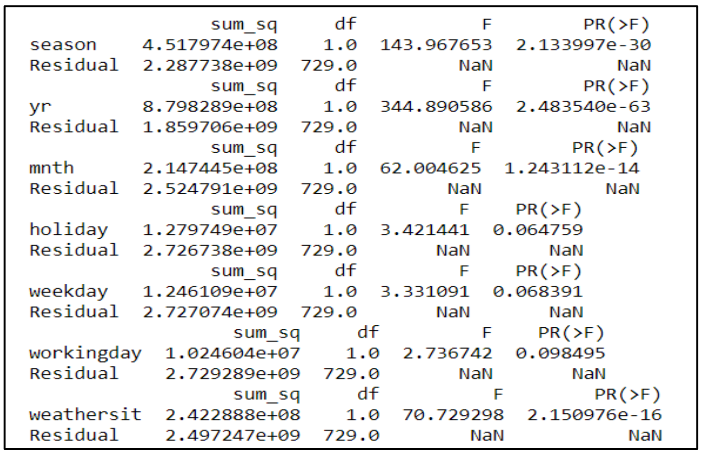
Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multi-collinearity between variables. The highly collinear variables are dropped and then the model is executed.

➢ Correlation Analysis for Numerical Variables.



We observe that 'temp' and 'atemp' variables has high correlation (>0.9) with each other. So, in further processes we will drop 'atemp' as it is similar to 'temp' variable.

➢ ANOVA Test for Categorical Variables



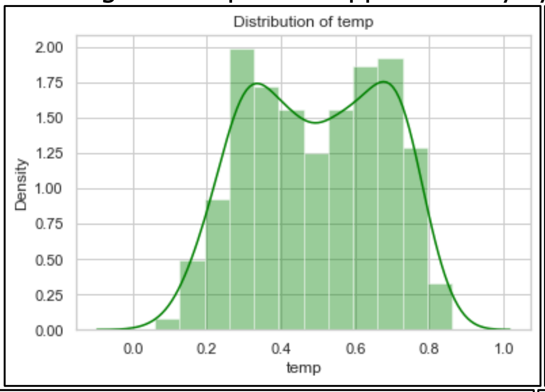
```
                sum_sq     df           F        PR(>F)
season    4.517974e+08    1.0  143.967653  2.133997e-30
Residual  2.287738e+09  729.0         NaN           NaN
                sum_sq     df           F        PR(>F)
yr        8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  729.0         NaN           NaN
                sum_sq     df          F        PR(>F)
mnth      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)
weathersit  2.422888e+08    1.0  70.729298  2.150976e-16
Residual    2.497247e+09  729.0        NaN           NaN
```

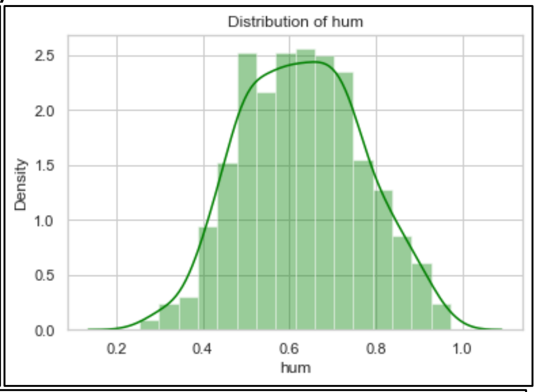
In ANOVA analysis, it is found that the in categorical variables 'holiday', 'weekday' and 'working day' have the p value >0.05, so null hypothesis is accepted (i.e. these variables have no dependency over target variable). Therefore we will be excluded them before further modelling.

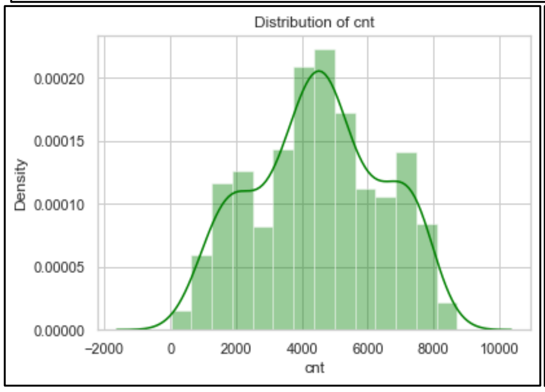
2.1.4 Features Scaling

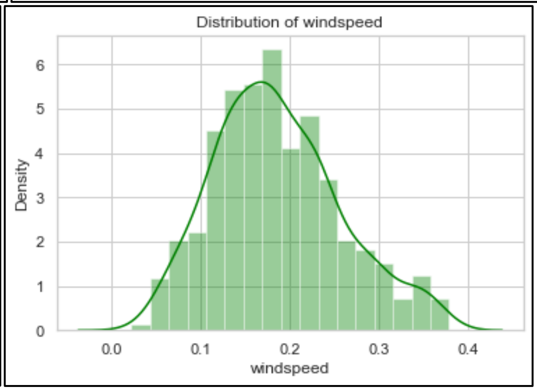
Feature scaling is a method used to standardize the range of independent variables or features of data. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. When you normalize data you eliminate the units of measurement for the data, enabling you to more easily compare data from different places. Some of the more common ways to normalize data is by transforming data using a z-score or t-score. This is usually called as standardization.

In Feature Scaling ranges of variables are normalized or standardized, such that variables can be compared with same range. In this project, since in given dataset for the continuous variables is already normalized and found to be approximately symmetric, feature scaling is not required. Following are the plots of approximately symmetric data visuals.









	season	yr	mnth	weathersit	temp	hum	windspeed	cnt
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	0.629354	0.186257	4504.348837
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156	1937.211452
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.254167	0.022392	22.000000
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.522291	0.134950	3152.000000
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.627500	0.178802	4548.000000
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.229786	5956.000000
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108	8714.000000

2.2 Model Development

After cleaning the data by removing outliers and missing values and pre-processing the data, next step would be Model Development. Now we are having data ready to be implemented for developing a model. There are number of models and machine learning algorithms that can used to develop model. It includes decision tree, random forest, KNN, Naïve Bayes, Linear regression, Logistic Regression etc. So, before implementing any model we should choose precisely our model. So, the first step in the Model Development is selection of model.

2.2.1 Model Selection

First step in selecting the suitable machine learning algorithm for a problem statement is to categorize, analysing and understanding the data. There are many categories in which a problem may lie like forecasting, classification, optimisation, unsupervised learning etc. If the output of the model is a number, it's a regression problem. If the output of the model is a class, it's a classification problem. If the output of the model is a set of input groups, it's a clustering problem.

Choosing the right machine learning algorithm depends on several factors, including, but not limited to: data size, quality and diversity, as well as what answers businesses want to derive from that data.

The process of selecting suitable model depends on our goal and the problem statement. In this project the goal is to build a models which will predict the count of bike rentals based on the seasonal and environmental settings. Thus, the problem statement is an identified as regression problem and it will fall under the category of forecasting where we need to forecast a numeric data or continuous variable for the target.

The dependent variable in our model is a continuous variable i.e., Total Count of bike rentals. Therefore the models that we choose in this project are Decision Tree, Random Forest and Linear Regression

2.2.1.1 Decision tree

Decision Tree is a supervised learning predictive model which is used to predict the data for classification and regression. It uses a set of binary rules in order to calculate the target value/dependent variable. It uses a tree-like model of decisions. A decision tree can be used to visually and also explicitly represent decision making. It accepts both continuous and categorical variables.

Decision trees are divided into three main parts this are:

- Root Node : performs the first split
- Terminal Nodes : that predict the outcome which are also called leaf nodes
- Branches: arrows connecting nodes which shows the flow from root to other leaves.

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". It provides its output in the form of rule, which can easily understood by a non-technical person also.

2.2.1.2 Random Forest

The next model to be followed in this project is 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 in order 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. Random decision forests correct for decision tree's habit of overfitting to their training set.

2.2.1.3 Linear Regression

The next method in the process is linear regression. Linear Regression is one of the statistical method of prediction. It is most common predictive analysis algorithm. It is used to predict the value of variable Y based on one or more input predictor variables X. The goal of this method is to establish a linear relationship between the predictor variables and the response variable. Such that, we can use this formula to estimate the value of the response Y, when only the predictors (X- Values) are known.

Further we will evaluate the developed models under various error metrics and select the best suitable model in order to predict the target variable with less error.

CHAPTER 3: EVALUATION OF THE MODEL

Once the models are developed for predicting the target variable, next step is evaluate the models and identify which one is most suitable for deployment. To evaluate the model, error metrics are used. In this project, we will be using MAPE, R Square and Accuracy as error metrics

3.1 Error Metrics

- **MAE (Mean Absolute Error)**
  It is one of the error measures that is used to calculate the predictive performance of the model. It is the sum of calculated errors.

- **MAPE (Mean Absolute Percent Error)**
  MAPE is a measure of prediction accuracy of a forecasting method. It is the measures the size of the error in percentage terms. It is calculated as the average of the unsigned percentage error.
  MAPE is calculated using below expression,

$$\left(MAPE = \frac{1}{n}\sum\frac{|Actual - \ Forecast|}{|Actual|}\right) * 100$$

- **Accuracy**
  It is the ratio of number of correct predictions to the total number of predictions made.

  Accuracy= number of correct predictions / Total predictions made

  It can also be calculated from MAPE as
  Accuracy = 1- MAPE

- **R Square**
  R Square is another metric that helps us to know the Correlation between original and predicted values. R-squared is basically explains the degree to which input variable explain the variation of the output. In simple words R-squared explains how much variance of dependent variable explained by the independent variable. It is a measure of goodness of fit in regression line. Value of R-squared ranges between 0-1, where 0 means independent variable is unable to explain the target variable and 1 means the target variable is completely explained by the independent variable.

$$RSME = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,i})^2}{n}}$$

Lower values of MAPE and higher value of R-Squared Value indicate better fit of model.

3.1.1 MAPE (Mean Absolute Percent Error)

We evaluated the above metrics in both R and python and the values are compiled below,

Method	In R	In Python
Decision Tree	26.4225	36.94
Random Forest	19.3210	20.71
Linear Regression	21.5679	18.80

The model which has lowest MAPE should be chosen as a suitable Model. Here, from R we can observe that Random Forest as a better model, whereas from Python we observe that Linear Regression as a better model. So following this we can conclude that Both Random Forest and Linear Regression can be used as model for this data, if you evaluate on the basis of MAPE. But we need more error metrics to cross check this. So, we go for R Square which is a better error metric.

3.1.2 ACCURACY

We evaluated the above metrics in both R and python and the values are compiled below,

Method	In R	In Python
Decision Tree	73.57	63.05
Random Forest	80.67	79.29
Linear Regression	78.43	81.20

The models with high accuracy is chosen as suitable model. As, Accuracy is based on MAPE percentage, here also it is found that both Random Forest and Linear Regression are good models for the given data set.

3.1.3 R Square

We evaluated the above metrics in both R and python and the values are compiled below,

Method	In R	In Python
Decision Tree	76.12	65.45
Random Forest	86.85	88.41
Linear Regression	81.91	84.36

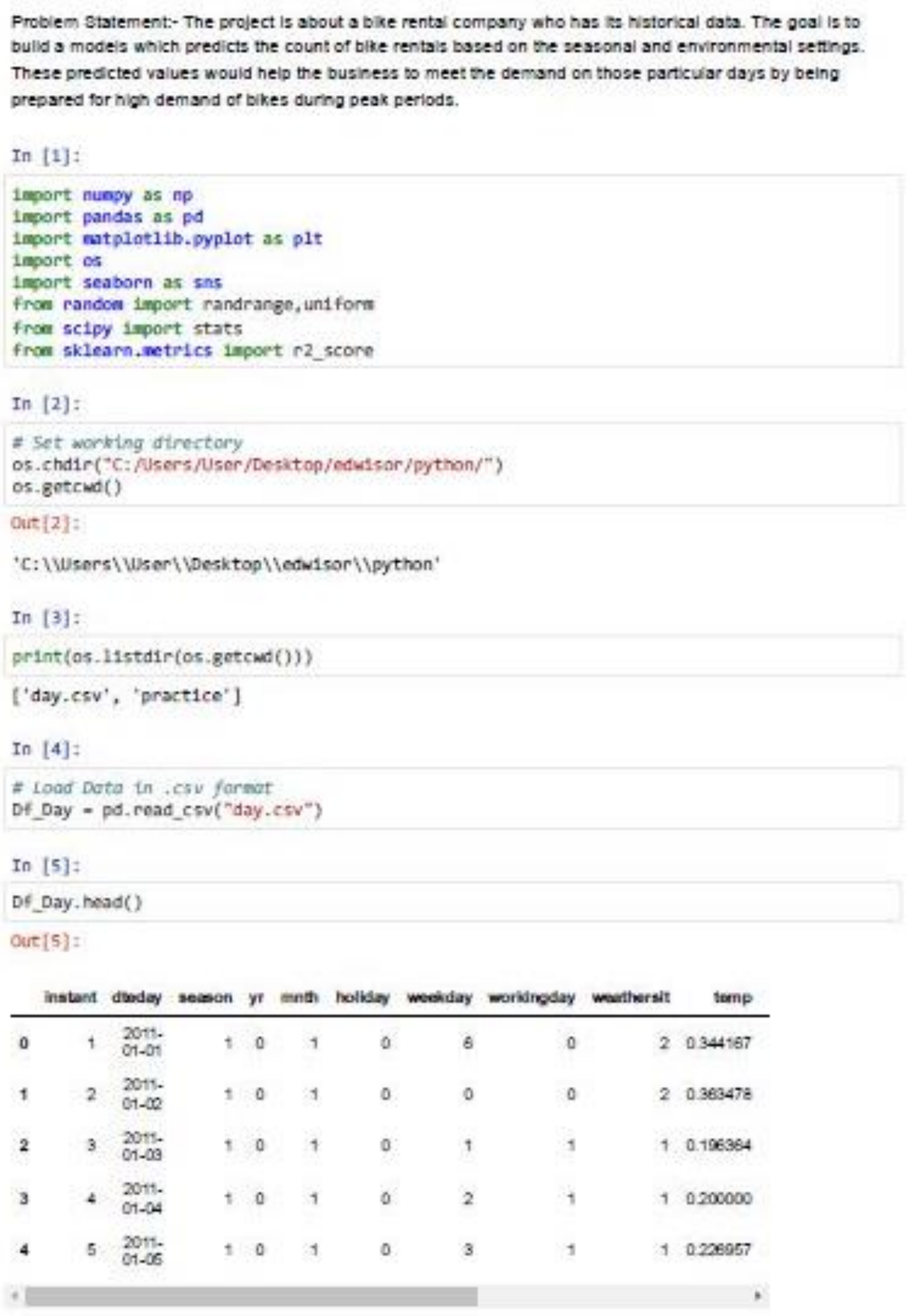
The model which has highest R Square value should be chosen as a suitable Model. R Square is identified as a better error metric to evaluate models. When we observe the values from above table, we choose the model with highest R Square as a suitable Model. Here, from both R and Python it is found that Random Forest is a best fit model for the given data.

3.2 Conclusion

Upon evaluating the above error metrics, we can conclude that **Random Forest** is the better model for our analysis. Hence we choose Random Forest as the model for prediction of bike rental count.

APPENDIX A – Python Code

PYTHON CODE



Problem Statement:- The project is about a bike rental company who has its historical data. The goal is to build a models which predicts the count of bike rentals based on the seasonal and environmental settings. These predicted values would help the business to meet the demand on those particular days by being prepared for high demand of bikes during peak periods.

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import os
import seaborn as sns
from random import randrange,uniform
from scipy import stats
from sklearn.metrics import r2_score
```

In [2]:

```
# Set working directory
os.chdir("C:/Users/User/Desktop/edwisor/python/")
os.getcwd()
```

Out[2]:

```
'C:\\Users\\User\\Desktop\\edwisor\\python'
```

In [3]:

```
print(os.listdir(os.getcwd()))
```

```
['day.csv', 'practice']
```

In [4]:

```
# Load Data in .csv format
Df_Day = pd.read_csv("day.csv")
```

In [5]:

```
Df_Day.head()
```

Out[5]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957

EXPLORATORY DATA ANALYSIS

In [6]:

```
# To check the data Types of Varaibles

Df_Day.dtypes
```

Out[6]:

```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

In [7]:

```
#Shape of the data
Df_Day.shape
```

Out[7]:

```
(731, 16)
```

The dataset contains 731 observations and 16 attributes.

In [8]:

```
# To get columns names

Df_Day.columns
```

Out[8]:

```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

In [9]:

```
# To check the unique values which present in each variable

Df_Day.nunique()
```

Out[9]:

```
instant       731
dteday        731
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          499
atemp         690
hum           595
windspeed     650
casual        606
registered    679
cnt           696
dtype: int64
```

In [10]:

```
##Dropping the variables which are not necessary for our model
#variable "instant" can be dropped as it simply represents the index number
# casual and registered variables can be removed, as these two sums to dependent variab
le count and which is what we need to predict
# Variable "dteday" can be ignored as the output is not based on time series analysis

Df_Day = Df_Day.drop(Df_Day.columns[[0, 1, 13, 14]], axis = "columns")

Df_Day.shape
```

Out[10]:

```
(731, 12)
```

In [11]:

```
#Classifying into numeric and categorical variables and saving those in a specific arra
y

numeric_var = ['temp', 'atemp', 'hum', 'windspeed', 'cnt']

categorical_var = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weather
sit']
```

DATA PRE PROCESSING

Missing Value Analysis

In [12]:

```
#sum of the missing values

Df_Day.isnull().sum()
```

Out[12]:

```
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

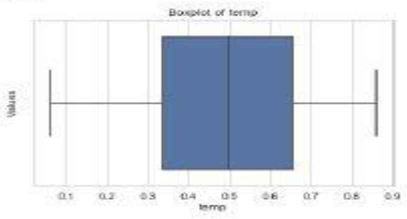
No missing values found

Outlier Analysis

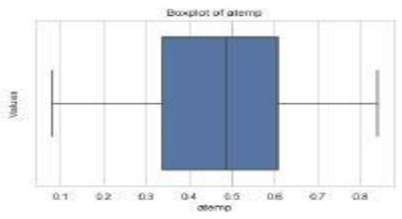
In [13]:

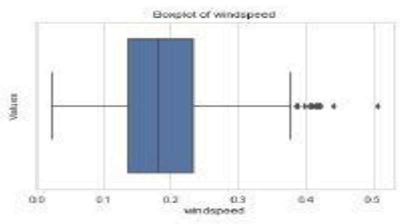
```
for i in numeric_var:
    print(i)
    sns.set(style="whitegrid")
    sns.boxplot(y = Df_Day[i], orient="h")
    plt.xlabel(i)
    plt.ylabel("Values")
    plt.title("Boxplot of " + i)
    plt.show()
```

```
temp
```

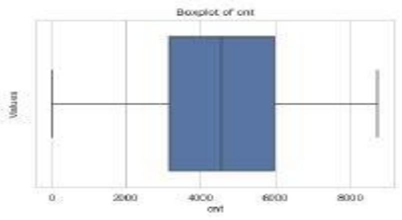


```
atemp
```

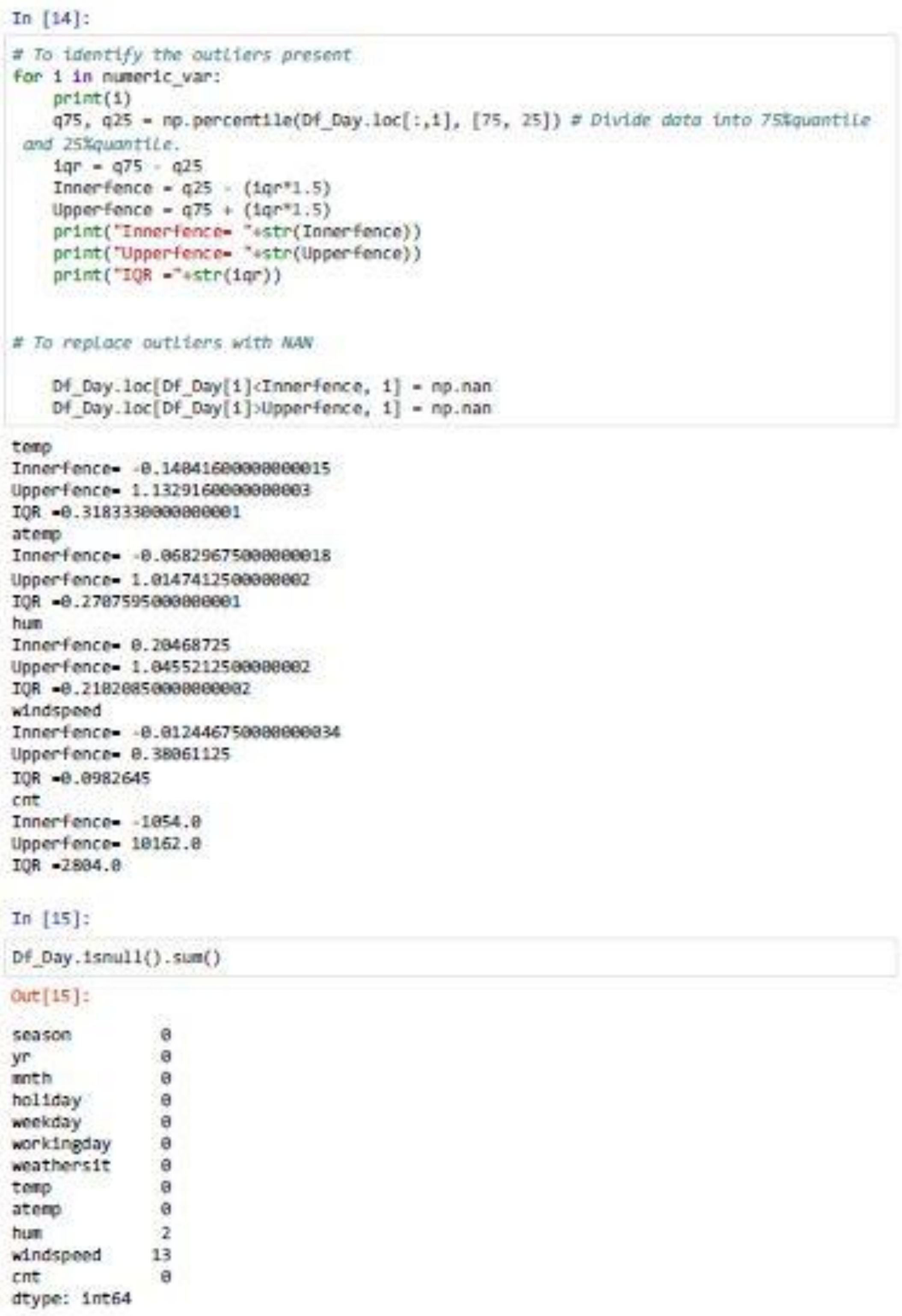




```
cnt
```



outliers are found in windspeed and humidity variables.



In [14]:

```
# To identify the outliers present
for i in numeric_var:
    print(i)
    q75, q25 = np.percentile(Df_Day.loc[:,i], [75, 25]) # Divide data into 75%quantile
 and 25%quantile.
    iqr = q75 - q25
    Innerfence = q25 - (iqr*1.5)
    Upperfence = q75 + (iqr*1.5)
    print("Innerfence= "+str(Innerfence))
    print("Upperfence= "+str(Upperfence))
    print("IQR ="+str(iqr))


# To replace outliers with NAN

    Df_Day.loc[Df_Day[i]<Innerfence, i] = np.nan
    Df_Day.loc[Df_Day[i]>Upperfence, i] = np.nan
```

```
temp
Innerfence= -0.14041600000000015
Upperfence= 1.1329160000000003
IQR =0.3183330000000001
atemp
Innerfence= -0.06829675000000018
Upperfence= 1.0147412500000002
IQR =0.2707595000000001
hum
Innerfence= 0.20468725
Upperfence= 1.0455212500000002
IQR =0.2102085000000002
windspeed
Innerfence= -0.012446750000000034
Upperfence= 0.38061125
IQR =0.0982645
cnt
Innerfence= -1054.0
Upperfence= 10162.0
IQR =2804.0
```

In [15]:

```
Df_Day.isnull().sum()
```

Out[15]:

```
season         0
yr             0
mnth           0
holiday        0
weekday        0
workingday     0
weathersit     0
temp           0
atemp          0
hum            2
windspeed     13
cnt            0
dtype: int64
```

Total 15 outliers found, out of which 13 are present in windspeed and remaining 2 in humidity variable.

```
In [16]:
```

```
#  Now we impute the values, by the help of median method.

Df_Day['hum'] = Df_Day['hum'].fillna(Df_Day['hum'].median())
Df_Day['windspeed'] = Df_Day['windspeed'].fillna(Df_Day['windspeed'].median())
```

```
In [17]:
```

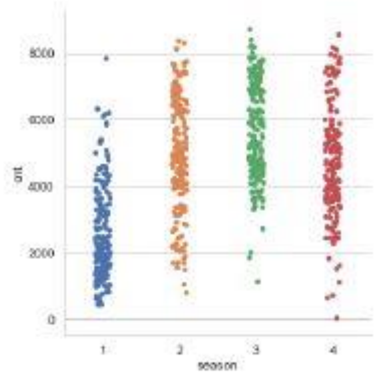
```
# To check the imputation result

Df_Day.isnull().sum()
```

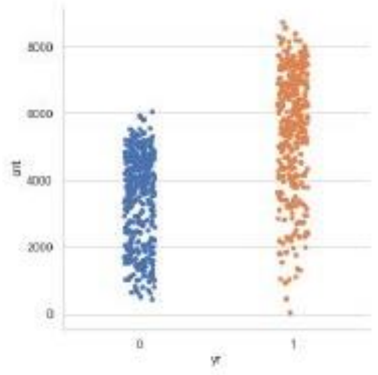
```
Out[17]:

season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

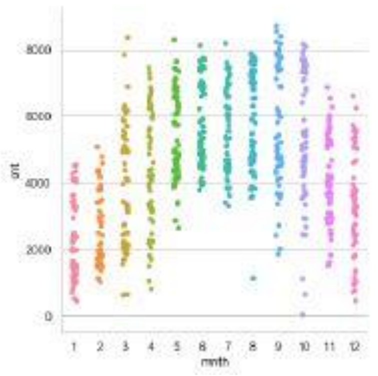
DATA UNDERSTANDING

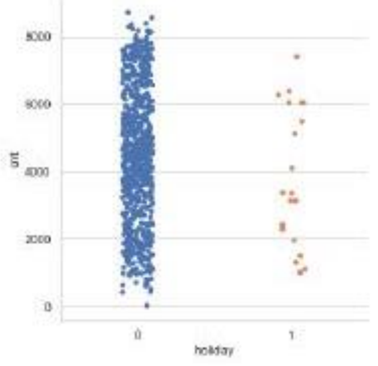
```
In [18]:
```

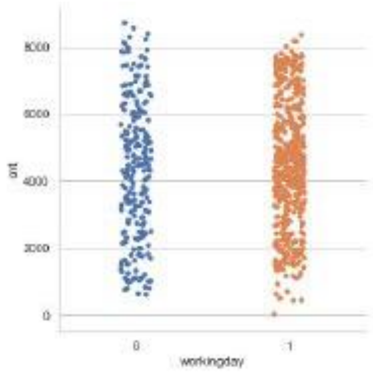
```
for i in categorical_var:
    sns.catplot(x = i, y = "cnt", data=Df_Day)
```

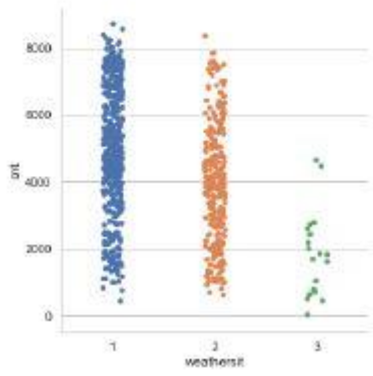










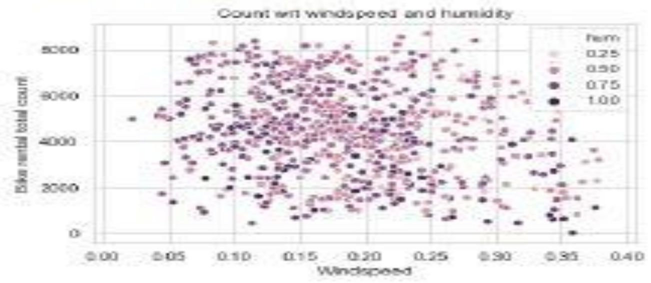


In [19]:

```
scatter_plot1 = sns.scatterplot(x="windspeed", y="cnt", hue="hum", data= Df_Day)

plt.title("Count wrt windspeed and humidity")
plt.ylabel("Bike rental total count")
plt.xlabel("Windspeed")
```

Out[19]:

```
Text(0.5, 0, 'Windspeed')
```



From the above scatter plot it is observed that in Count v/s windspeed and humidity, Count is High in ranges, windspeed 0.10 to 0.25 and humidity 0.5 to 0.75

FEATURE SELECTION

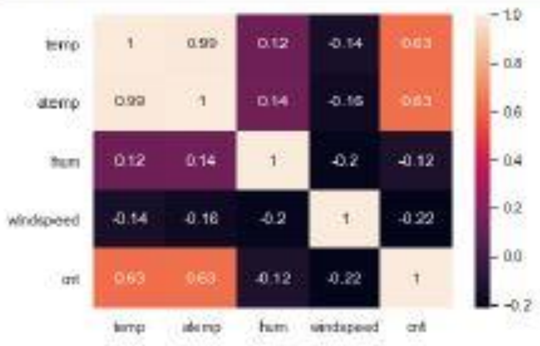
In [20]:

```
# Correlation Analysis to find varaibles which can be excluded

Df_Day_cor = Df_Day.loc[:, numeric_var]
correlation_result = Df_Day_cor.corr()
print(correlation_result)
```

```
               temp     atemp       hum  windspeed       cnt
temp       1.000000  0.991702  0.123723  -0.138937  0.627494
atemp      0.991702  1.000000  0.137312  -0.164157  0.631066
hum        0.123723  0.137312  1.000000  -0.200237 -0.121454
windspeed -0.138937 -0.164157 -0.200237   1.000000 -0.215203
cnt        0.627494  0.631066 -0.121454  -0.215203  1.000000
```

In [21]:

```
heatmap = sns.heatmap(correlation_result, annot=True)
```



It is found that temperature and atemp are highly correlated with each other.

In [22]:

```
#Anova test to find varaibles which can be dropped

import statsmodels.api as sm
from statsmodels.formula.api import ols

for i in categorical_var:
    mod = ols('cnt' + '~' + i, data = Df_Day).fit()
    anova_table = sm.stats.anova_lm(mod, typ = 2)
    print(anova_table)
```

```
                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)
yr        8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  729.0         NaN           NaN
                sum_sq     df          F        PR(>F)
mnth      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)
weathersit  2.422888e+08    1.0  70.729298  2.150976e-16
Residual    2.497247e+09  729.0        NaN           NaN
```

It is found that holiday, weekday and workingday has p value > 0.05, by which, we accept null hypothesis.

In [23]:

```
#Dimension Reduction. Drop atemp, holiday, weekday and working day

Df_Day = Df_Day.drop(['atemp', 'holiday', 'weekday', 'workingday'], axis = "columns")
Df_Day.shape
```

Out[23]:

```
(731, 8)
```

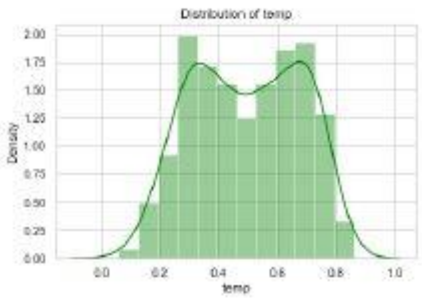
In [24]:

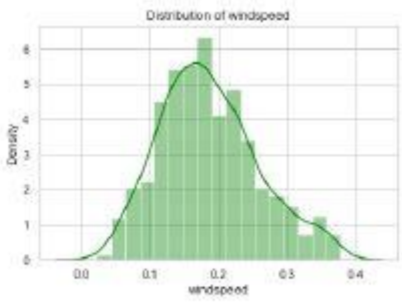
```
#Final Variables- the cleaned data

numeric_var = ["temp","hum","windspeed","cnt"]    # numeric variables

categorical_var = ["season", "yr", "mnth", "weathersit"]    # categorical variables
```

FEATURE SCALING

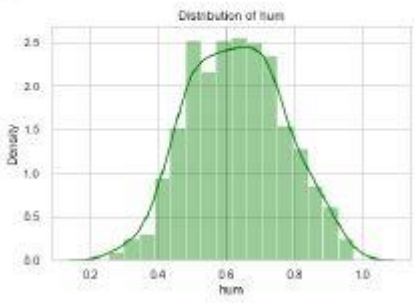
```
In [25]:
# To check whether the variables are normally distributed
for i in numeric_var:
    print(i)
    sns.distplot(Df_Day[i], bins = 'auto', color = 'green')
    plt.title("Distribution of "+i)
    plt.ylabel("Density")
    plt.show()
```

```
temp
```

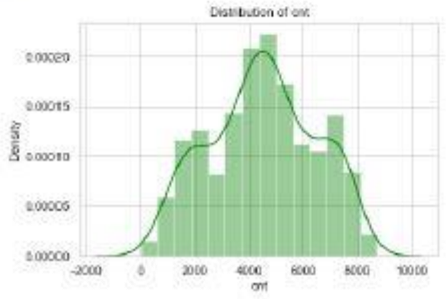




```
hum
```



```
cnt
```



Distributions are approximately symmetric

```
In [26]:
Df_Day.describe()
Out[26]:
```

	season	yr	mnth	weathersit	temp	hum	windspeed
count	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	0.629354	0.186257
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.254167	0.022392
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.522291	0.134950
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.627500	0.178802
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.229786
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108

MODEL DEVELOPMENT

In [27]:

```
df = Df_Day.copy()
Df_Day = df.copy()
```

In [28]:

```
# Creating dummy variables

Df_Day = pd.get_dummies(Df_Day, columns = categorical_var)

Df_Day.shape
```

Out[28]:

```
(731, 25)
```

In [29]:

```
Df_Day.head()
```

Out[29]:

	temp	hum	windspeed	cnt	season_1	season_2	season_3	season_4	yr_0
0	0.344167	0.805833	0.160446	985.0	1	0	0	0	1
1	0.363478	0.696087	0.248539	801.0	1	0	0	0	1
2	0.196364	0.437273	0.248309	1349.0	1	0	0	0	1
3	0.200000	0.590435	0.160296	1562.0	1	0	0	0	1
4	0.226957	0.436957	0.186900	1600.0	1	0	0	0	1

5 rows X 25 columns

In [30]:

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from scipy.stats.stats import pearsonr
```

In [31]:

```
#define the Error Metrics.

def MAPE(y_actual, y_predicted):
    MAPE = np.mean(np.abs(y_actual-y_predicted)/y_actual)*100
    return MAPE

def Rsquare(y_actual, y_predicted):
    Rsquare = np.corrcoef(y_actual,y_predicted)**2
    return Rsquare
```

In [32]:

```
#predictors and target variables

X = Df_Day.drop(['cnt'], axis = "columns")
y = Df_Day['cnt']
```

In [33]:

```
#divide the data into train and test

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2, random_state=0)
```

DECISION TREE

In [34]:

```
from sklearn.tree import DecisionTreeRegressor
DTModel = DecisionTreeRegressor(max_depth=2).fit(X_train,y_train)

# Prediction

DTTest = DTModel.predict(X_test)

# MAPE
DTMape_Test = MAPE(y_test, DTTest)


# Rsquare

DTR2_Test = Rsquare(y_test, DTTest)

DTR2_Test1 = DTR2_Test.ravel()

DTR2_Test2 = float(DTR2_Test1[1])


print("MAPE ="+str(DTMape_Test))
print("Accuracy =" + str(100 - DTMape_Test))
print("Rsquare ="+str(DTR2_Test2))
```

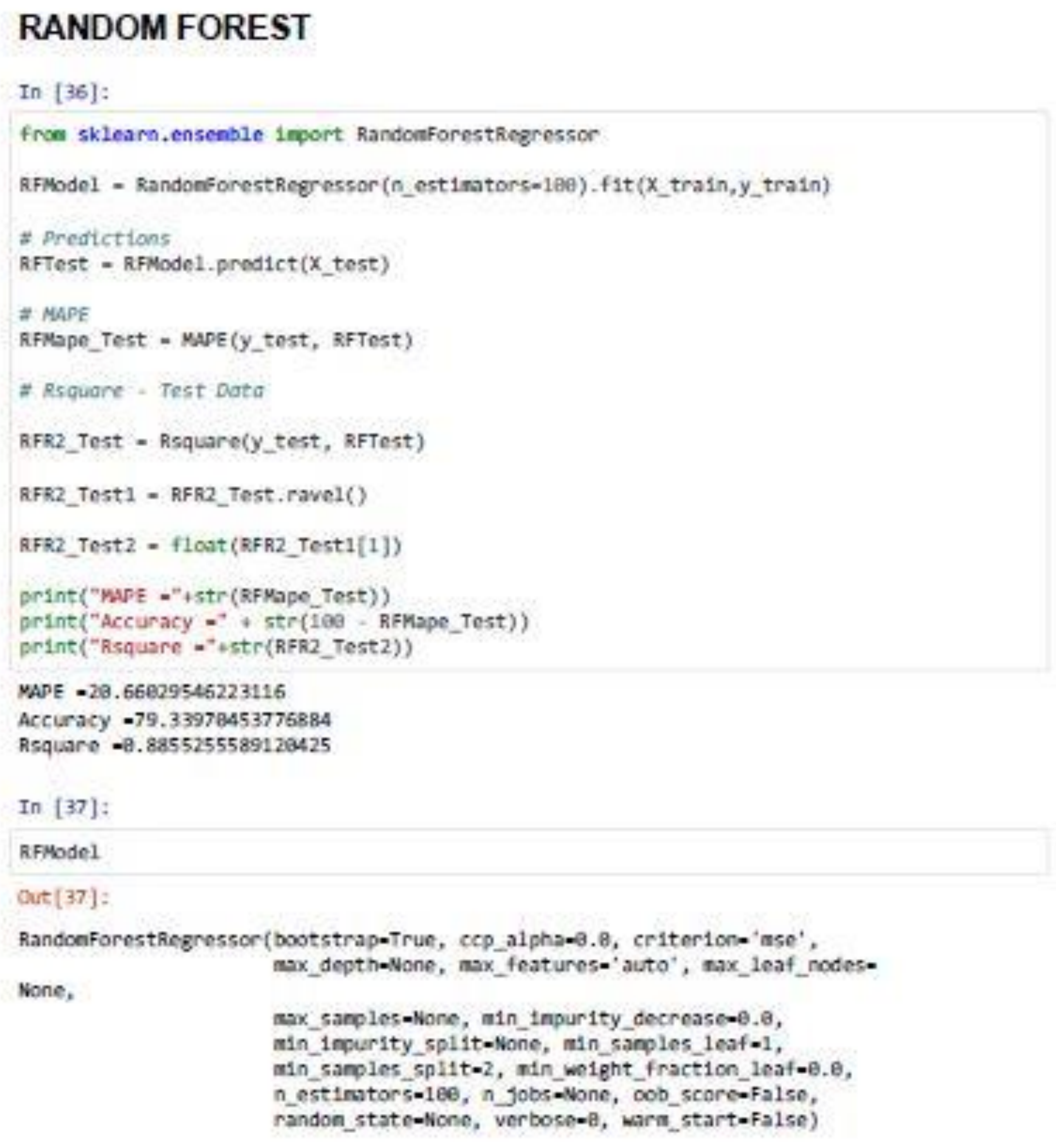
```
MAPE =36.94809301452646
Accuracy =63.05190698547354
Rsquare =0.6544606873373328
```

In [35]:

```
DTModel
```

Out[35]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=2,
                      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='deprecated',
                      random_state=None, splitter='best')
```



RANDOM FOREST

In [36]:

```
from sklearn.ensemble import RandomForestRegressor

RFModel = RandomForestRegressor(n_estimators=100).fit(X_train,y_train)

# Predictions
RFTest = RFModel.predict(X_test)

# MAPE
RFMape_Test = MAPE(y_test, RFTest)

# Rsquare - Test Data

RFR2_Test = Rsquare(y_test, RFTest)

RFR2_Test1 = RFR2_Test.ravel()

RFR2_Test2 = float(RFR2_Test1[1])

print("MAPE ="+str(RFMape_Test))
print("Accuracy =" + str(100 - RFMape_Test))
print("Rsquare ="+str(RFR2_Test2))
```

```
MAPE =20.66029546223116
Accuracy =79.33970453776884
Rsquare =0.8855255589120425
```

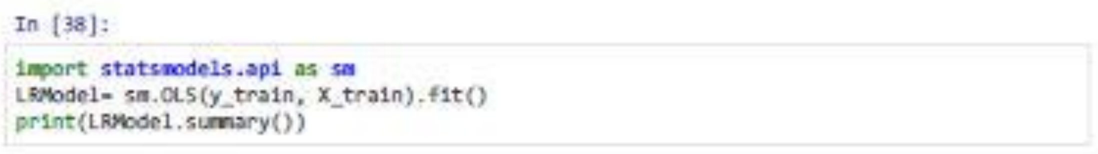
In [37]:

```
RFModel
```

Out[37]:

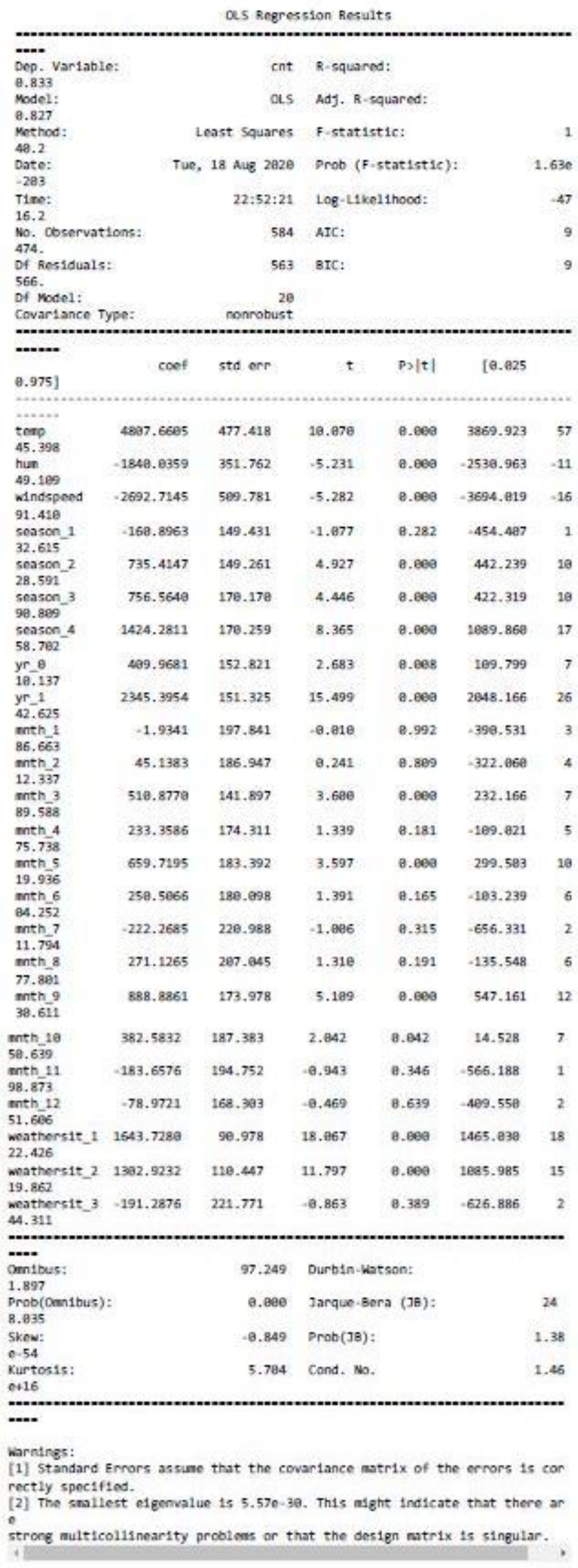
```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=
None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

LINEAR REGRESSION MODEL

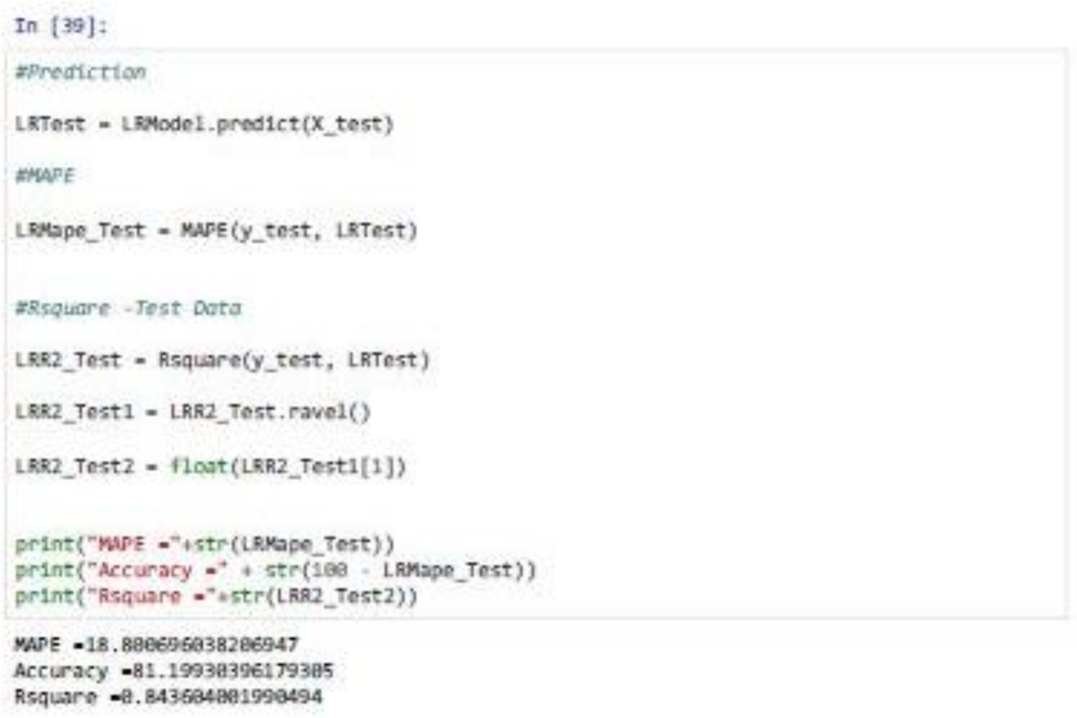


In [38]:

```
import statsmodels.api as sm
LRModel= sm.OLS(y_train, X_train).fit()
print(LRModel.summary())
```



```
                            OLS Regression Results
==============================================================================
====
Dep. Variable:                    cnt   R-squared:
0.833
Model:                            OLS   Adj. R-squared:
0.827
Method:                 Least Squares   F-statistic:                     1
40.2
Date:                Tue, 18 Aug 2020   Prob (F-statistic):          1.63e
-203
Time:                        22:52:21   Log-Likelihood:                -47
16.2
No. Observations:                 584   AIC:                             9
474.
Df Residuals:                     563   BIC:                             9
566.
Df Model:                          20
Covariance Type:            nonrobust
==============================================================================
======
                  coef    std err          t      P>|t|      [0.025
0.975]
------------------------------------------------------------------------------
------
temp         4807.6605    477.418     10.070      0.000    3869.923     57
45.398
hum         -1840.0359    351.762     -5.231      0.000   -2530.963    -11
49.109
windspeed   -2692.7145    509.781     -5.282      0.000   -3694.019    -16
91.410
season_1     -160.8963    149.431     -1.077      0.282    -454.407      1
32.615
season_2      735.4147    149.261      4.927      0.000     442.239     10
28.591
season_3      756.5640    170.170      4.446      0.000     422.319     10
90.809
season_4     1424.2811    170.259      8.365      0.000    1089.860     17
58.702
yr_0          409.9681    152.821      2.683      0.008     109.799      7
10.137
yr_1         2345.3954    151.325     15.499      0.000    2048.166     26
42.625
mnth_1         -1.9341    197.841     -0.010      0.992    -390.531      3
86.663
mnth_2         45.1383    186.947      0.241      0.809    -322.060      4
12.337
mnth_3        510.8770    141.897      3.600      0.000     232.166      7
89.588
mnth_4        233.3586    174.311      1.339      0.181    -109.021      5
75.738
mnth_5        659.7195    183.392      3.597      0.000     299.583     10
19.936
mnth_6        250.5066    180.098      1.391      0.165    -103.239      6
04.252
mnth_7       -222.2685    220.988     -1.006      0.315    -656.331      2
11.794
mnth_8        271.1265    207.045      1.310      0.191    -135.548      6
77.801
mnth_9        888.8861    173.978      5.109      0.000     547.161     12
30.611
mnth_10       382.5832    187.383      2.042      0.042      14.528      7
50.639
mnth_11      -183.6576    194.752     -0.943      0.346    -566.188      1
98.873
mnth_12       -78.9721    168.303     -0.469      0.639    -409.550      2
51.606
weathersit_1 1643.7280     90.978     18.067      0.000    1465.030     18
22.426
weathersit_2 1302.9232    110.447     11.797      0.000    1085.985     15
19.862
weathersit_3 -191.2876    221.771     -0.863      0.389    -626.886      2
44.311
==============================================================================
====
Omnibus:                       97.249   Durbin-Watson:
1.897
Prob(Omnibus):                  0.000   Jarque-Bera (JB):               24
8.035
Skew:                          -0.849   Prob(JB):                     1.38
e-54
Kurtosis:                       5.704   Cond. No.                     1.46
e+16
==============================================================================
====

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is cor
rectly specified.
[2] The smallest eigenvalue is 5.57e-30. This might indicate that there ar
e
strong multicollinearity problems or that the design matrix is singular.
```



```
In [39]:

#Prediction

LRTest = LRModel.predict(X_test)

#MAPE

LRMape_Test = MAPE(y_test, LRTest)


#Rsquare -Test Data

LRR2_Test = Rsquare(y_test, LRTest)

LRR2_Test1 = LRR2_Test.ravel()

LRR2_Test2 = float(LRR2_Test1[1])


print("MAPE ="+str(LRMape_Test))
print("Accuracy =" + str(100 - LRMape_Test))
print("Rsquare ="+str(LRR2_Test2))
```

```
MAPE =18.800696038206947
Accuracy =81.19930396179305
Rsquare =0.843604001990494
```

APPENDIX B – R Code

R CODE

```
#Clean the environment
rm(list=ls())

#Set Working Directory
setwd("C:/Users/User/Desktop/edwisor/New folder")

#get Working directory
getwd()

#Load the librarires
            c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071",
"information",
      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')


install.packages(libraries)
lapply(X = libraries,require, character.only = TRUE)
rm(libraries)

#load Bike rental data in R

df_day= read.csv(file = "day.csv", header = T)

########################################EXPLORATARY DATA ANALYSIS########################################
# Summarizing  data

#Class of data
class(df_day)

#Verify first five rows of data

head(df_day)

#Dimensions of data
dim(df_day)

#Column names

names(df_day)

#Structure of variables
str(df_day)

#Verify  summary of data
summary(df_day)

##Dropping the variables which are not necessary for our model

#variable "instant" can be dropped as it simply represents the index number
# casual and registered variables can be removed, as these two sums to dependent variable count and which is what we
need to predict
# Variable "dteday" can be ignored as the output is not based on time series analysis

df_day = subset(df_day, select = -c(instant, dteday, casual, registered))

#Dimensions of data after dropping
dim(df_day)
names(df_day)

#Classifying into numeric and categorical variables and saving those in a specific array

numeric_var = c('temp', 'atemp', 'hum', 'windspeed', 'cnt')

categorical_var = c('season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit')

####Missing Value analysis
summary(is.na(df_day))
sum(is.na(df_day))

#No missing values in dataset

####Outlier Analysis

df = df_day
df day = df
```

```
# #Check for outliers in data using boxplot method
library(ggplot2)

for (i in 1:length(numeric_var))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (numeric_var[i]), x = "cnt"), data = subset(df_day))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=numeric_var[i],x="count")+
           ggtitle(paste("Box plot of count versus",numeric_var[i])))
}

## Plotting of the plots together
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5, ncol=2)

#outliers are found in windspeed and humidity variables.

#To replace outliers with NA

for(i in numeric_var){
  print(i)
  outlier_number = df_day[,i][df_day[,i] %in% boxplot.stats(df_day[,i])$out]
  print(length(outlier_number))
  df_day[,i][df_day[,i] %in% outlier_number] = NA
}

sum(is.na(df_day))


#Impute the NA values with KNN

library(DMwR)
library(rpart)



  df_day = knnImputation(df_day, k = 5)

  sum(is.na(df_day))

  #### Data Understanding
  # In order to plot some graphs,install few libraries

  library(ggplot2)
  library(gplots)
  library(scales)
  library(psych)


  # Barplot with x axis being season and y axis being count
  ggplot(df_day, aes(x = 'season', y= 'cnt')) +
    geom_bar(stat = "identity", fill = "blue")+
    labs(title = "Count of bikes rented wrt season", x = "Seasons", y = "cnt")+
    theme(panel.background = element_rect("white"))+
    theme(plot.title = element_text(face = "bold"))
  #We can observe from the cat plots that In Season 2, 3 and 4 has the highest count compared to season 1

  # Barplot with x axis being year and y axis being count
  ggplot(df_day, aes(x = df_day$yr, y = df_day$cnt))+
    geom_bar(stat = "identity", fill = "red")+
    labs(title = "Count of bikes rented wrt year", x = "yr", y = "cnt")+
    theme(panel.background = element_rect("white"))+
    theme(plot.title = element_text(face = "bold"))
  #In Year 1 has high count than 0 (0= 2011, 1= 2012)


  # Barplot with x axis being weekday and y axis being count
  ggplot(df_day, aes(x = df_day$weekday, y = df_day$cnt))+
    geom_bar(stat = "identity", fill = "navyblue")+
    labs(title = "Count of bikes rented wrt days", x = "Days of week", y = "count")+
    theme(panel.background = element_rect("white"))+
    theme(plot.title = element_text(face = "bold"))
```

```
#In weekdays, 0 and 6 has the highest count

####Feature Selection
df2 = df_day
df_day = df2

# Correlation Analysis to find varaibles which can be excluded
library(corrgram)

corrgram(df_day[,numeric_var],order=FALSE,upper.panel = panel.pie,
         text.panel = panel.txt,
         main= "Correlation Analysis with numeric variables")
#It is found that temperature and atemp are highly correlated with each other.

#Anova test to find varaibles which can be dropped
for(i in categorical_var){
  print(i)
  Anova_test_result = summary(aov(formula = cnt~df_day[,i],df_day))
  print(Anova_test_result)
}
#It is found that holiday, weekday and workingday has p value > 0.05, by which, we accept null hypothesis.

#Dimension Reduction. Drop atemp, holiday, weekday and working day
df_day = subset(df_day, select=-c(atemp,holiday,weekday,workingday))

####Feature Scaling
#Final Variables- the cleaned data
numeric_var = c("temp","hum","windspeed","cnt")
catergorical_var = c("season", "yr", "mnth", "weathersit")

# To check whether the variables are normally distributed
# Skewness test

library(propagate)

for(i in numeric_var){
  print(i)

   skew = skewness(df_day[,i])
   print(skew)
 }
 # We can observe that dataset is approximately symmetric.


 # Summary of the variables to check normality

 for(i in numeric_var){
   print(summary(df_day[,i]))
 }

 #Data is found to be normalized, scaling not required

 # visualizing normality check

 hist(df_day$hum, col="Blue", xlab="Humidity", ylab="Frequency",
      main="Humidity Distribution")

 hist(df_day$temp, col="Navyblue", xlab="Temperature", ylab="Frequency",
      main="Temperature Distribution")

 hist(df_day$windspeed,col="Dark green",xlab="Windspeed",ylab="Frequency",
      main="Windspeed Distribution")

 #Distribution is approximately symmetric

 ###################MODELING ###########################


 library(DataCombine)
 rmExcept("df_day")


 df3 = df_day
 df_day = df3
```

```
##define the Error Metrics.

#MAPE

MAPE = function(y,y1){
  mean(abs((y-y1)/y))*100
}


#R Square

Rsquare = function(y,y1){
  cor(y,y1)^2
}


####creation of dummies

categorical_var = c("season","yr","mnth","weathersit")

library(dummies)

df_day = dummy.data.frame(df_day, categorical_var)

#Divide the data into train and test

set.seed(123)
train_index = sample(1:nrow(df_day),0.8*nrow(df_day))
train= df_day[train_index,]
test= df_day[-train_index,]


####check multicollinearity

numeric_var = c("temp","hum","windspeed", "cnt")

numeric_var2 = df_day[,numeric_var]


library(usdm)

vifcor(numeric_var2, th = 0.7)

#No collinearity problem observed.


########################DECISION TREE ##################################

library(rpart)

DTModel = rpart(cnt~., train, method = "anova" , minsplit=5)

# Prediction

DTTest = predict(DTModel, test[-25])

#summary
summary(DTModel)

#MAPE value

DTMape_Test = MAPE(test[,25], DTTest)
DTMape_Test   #26.42


#R Square value

DT_RSquare = Rsquare(test[,25], DTTest)
DT_RSquare  #0.7612


######################RANDOM FOREST##############################

library(randomForest)
set.seed(123)
```

```
RFModel = randomForest(cnt~., train, ntree = 500, importance = TRUE)

# Prediction

RFTest = predict(RFModel, test[-25])


# MAPE Value

RFMape_Test = MAPE(test[,25], RFTest)
RFMape_Test  #  19.32

#R Square value

RF_RSquare = Rsquare(test[,25], RFTest)
RF_RSquare   # 0.8685

######################LINEAR REGRESSION########################

LRModel = lm(cnt~., train)
#Summary
summary(LRModel)


# Predictions on test values

LRTest = predict(LRModel, test[-25])


#MAPE Value

LRMape_Test = MAPE(test[,25], LRTest)
LRMape_Test #  21.56


#R Square Value


 LR_RSquare = Rsquare(test[,25], LRTest)
 LR_RSquare  #  0.8191

 ########################Model Selection & Evaluation ########################

 print("MAPE Values")
 print(DTMape_Test)
 print(RFMape_Test)
 print(LRMape_Test)

 print("Accuracy")
 print(100 - DTMape_Test)
 print(100 - RFMape_Test)
 print(100 - LRMape_Test)


 print("R-Square Values")
 print(DT_RSquare)
 print(RF_RSquare)
 print(LR_RSquare)
```

Thank you