Bike Renting Prediction

Dishank Bari

Email: dishankbari@gmail.com

Content

Chapter 1: Introduction

1.1 Problem Statement

The objective of this project is to predict of bike rental count on daily based on the environmental and seasonal condition or settings. The given dataset has 731 observations and 16 variables in that 15 predictors and 1 target variable. The 15 predictors are the various environmental and seasonal factors like season, humidity, temperature etc. So we need to build a production-ready prediction model that predict the estimate count or demand of bikes on a particular day based on environmental and season condition.

1.2 Dataset

Our task is to build a production ready regression model that predict the number of bikes used on particular day depending on the environmental and seasonal condition.

The given dataset consist of 731 observations (rows) and 16 variables (columns) in that 15 predictors and 1 target variable. This dataset is recorded over a period of two years between 2011 and 2012. All the variables names and description is mentioned in below table:

Table 1: Description of Variables

Variable Names	Description
instant	Record Index
dteday	Date
season	Season (1: Spring, 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 form Holiday Schedule)
weekday	Day of the Week

workingday	If day is neither weekend or holiday is 1, otherwise 0 (yes: 1, No: 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 in temperature in Celsius. The values are divided via (t – t_min)/(t_max – t_min) t_min = -8, t_max = +39 (only on hourly scale)
atemp	Normalized in temperature in Celsius. The values are divided via (t – t_min)/(t_max – t_min) t_min = -16, t_max = +50 (only on hourly scale)
hum	Normalized humidity. The values are divided to 100 (max)
windspeed	Normalized windpspeed. The values are divided by 67 (max)
casual	count of casual users
registered	count of registered users
cnt	count of total rental bikes including both casual and registered

The given dataset consist of 7 continuous and 8 categorical variables. Let's see the some of the sample data of given dataset

Table 2: Sample Data

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	2011-01-01	1	0	1	0	6	0	2
2	2011-01-02	1	0	1	0	0	0	2
3	2011-01-03	1	0	1	0	1	1	1
4	2011-01-04	1	0	1	0	2	1	1
5	2011-01-05	1	0	1	0	3	1	1

temp	atemp	hum	windspeed	casual	registered	cnt
0.33167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349
0.200000	0.212122	0.590435	0.160296	108	1454	1562
0.226957	0.229270	0.436957	0.186900	82	1518	1600

As you can see in the above table there are 15 predictors and 1 target variables. Now we put the complete list with which help us to predict the number of bikes used on particular day based on environmental and seasonal condition.

Chapter 2: Methodology

The solution of this problem statement is mainly divided into three parts. First was the Pre-Processing Analysis (i.e. Missing Value Analysis, Outliers Analysis, & Correlation Analysis). Second Was the Exploratory Data Analysis (Univariate, Bivariate & Feature Scaling) and last part was Modeling.

During first & second part we implement the various step like missing value analysis, outlier analysis, correlation analysis, univariate & bivariate analysis and feature scaling, etc. were performed. After that dataset is split into train and test dataset. As Target Variable 'cnt' is continuous numerical variable, so the problem is come under the regression model. For that reason, we used Linear Regression, Decision Tree Regression, Random Forest Regression and KNN Regression Algorithms model. Also we compare the performance of that model by implementing model on test dataset. All 4 algorithms were implement in R as well as in Python.

Understanding the dataset

After loading the dataset in both R and Python, we know check the number of observations and variables in it, found that there are 731 observations and 15 predictor variables and 1 target variable.

In dataset, some of the variables are not useful for further analysis for that reason we going to dropped that variables:

Dropping variables which are not required:

instant: index number, which is not useful in analysis

dteday: all the required parameters are already extracted from this variables like yr, mnth, weekday. So now this variable is not useful.

After dropping these two variables our dataset consist of 731 observations and 14 variables in that 13 predictors and 1 target variables.

2.1 Pre-Processing

2.1.1 Missing Value Analysis

The first and most important step in pre-processing analysis is Missing Value Analysis. This analysis is performed on the given dataset. We found that there is no missing values present in given dataset. Let's check the below image for missing value count for each variables.

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

Figure1: Missing Values per variables

2.1.2 Outlier Analysis

After missing value analysis, the foremost important pre-processing analysis is outlier analysis. Some of the observations are inconsistent with rest of the dataset are called outliers. This is due to poor data quality measurements, manual error and others.

In this data manipulation i.e. outlier analysis, we need to find out the inconsistent data from the given dataset. If inconsistent data is present in dataset will affect our further analysis that why we need to deal with them here.

Here, we used a most famous and well known outlier analysis method called Boxplot Method. First we checked the outliers for our Target Variable "cnt".

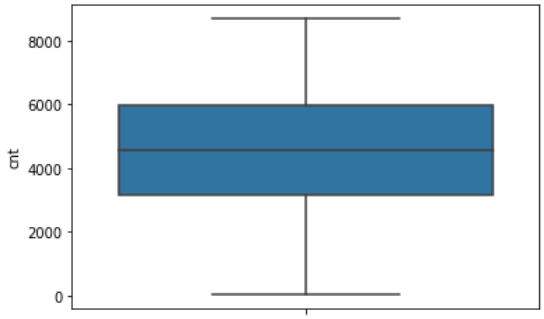


Figure2: Boxplot of Target Variable “cnt”

From the above boxplot of Target Variable “cnt”, it is evident that there is no outliers present in Target Variables “cnt”

Now, let’s check the outliers of predictor variables including “temp”, “atemp”, “hum”, “windspeed”, “casual”, and “registered” variables.

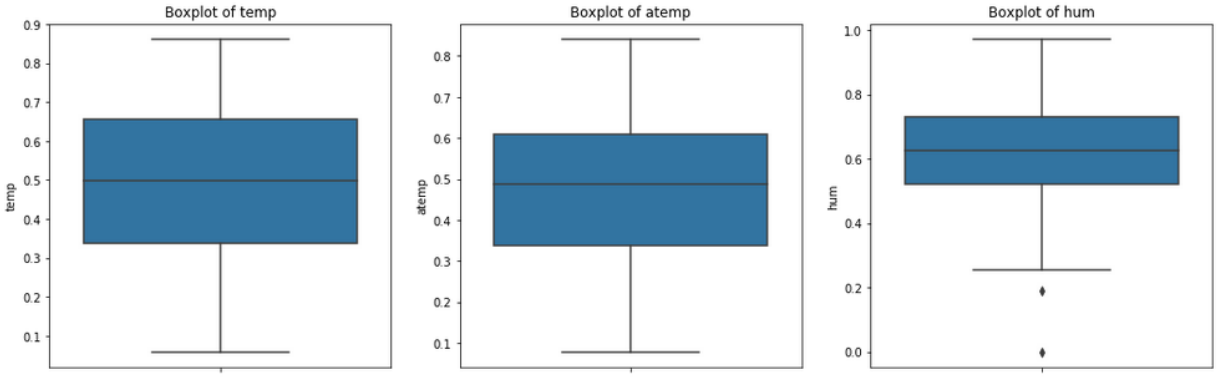


Figure3: Boxplot of temp, atemp and hum variables

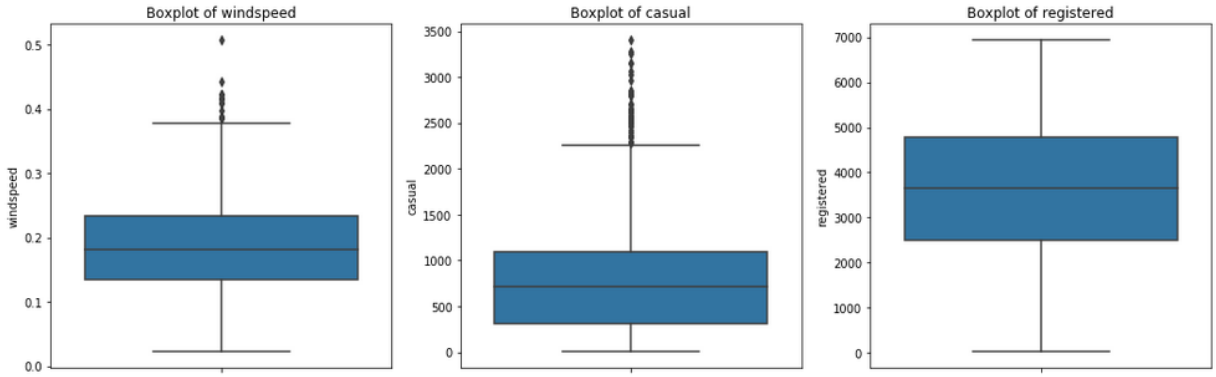


Figure4: Boxplot of windspeed, casual and registered variables

As the above image shows there are some outliers present in “hum”, “windspeed” and “casual” variables.

Here, we need to detect and delete the outliers that are present in “hum”, “windspeed” and “casual” variables using boxplot outlier removal method.

2.1.3 Correlation Analysis

Correlation Analysis is used to measure a linear relationship between two continuous variables. This analysis gives us an idea about the association between two continuous numerical variables. It is also used to check the muliticolinearity among predictors. Muliticolinearity is the condition when one predictor can be used to predict other. It is exist whenever two or more predictors in regression model are moderately or highly correlated.

Here we generate a correlation matrix and correlation plot

	temp	atemp	hum	windspeed	casual	registered	cnt
temp	1.000000	0.991483	0.122486	-0.139599	0.595525	0.545120	0.629031
atemp	0.991483	1.000000	0.135356	-0.167087	0.593962	0.547850	0.630906
hum	0.122486	0.135356	1.000000	-0.206719	-0.096350	-0.113078	-0.122854
windspeed	-0.139599	-0.167087	-0.206719	1.000000	-0.184026	-0.212375	-0.231596
casual	0.595525	0.593962	-0.096350	-0.184026	1.000000	0.427474	0.642890
registered	0.545120	0.547850	-0.113078	-0.212375	0.427474	1.000000	0.967266
cnt	0.629031	0.630906	-0.122854	-0.231596	0.642890	0.967266	1.000000

Figure5: correlation matrix of numerical variables

Important things to know for correlation matrix

1: highly positively correlated variables

-1: highly negatively correlated variables

0: no correlation between two variables

Variable acceptance and rejection based on following criteria:

- Correlation between two variables ranges from 0.7 to 1 and -1 to -0.7 then these variables, will not help to predict the final results, so you can drop any of the variable.
- Other than this accept these variables for model building.

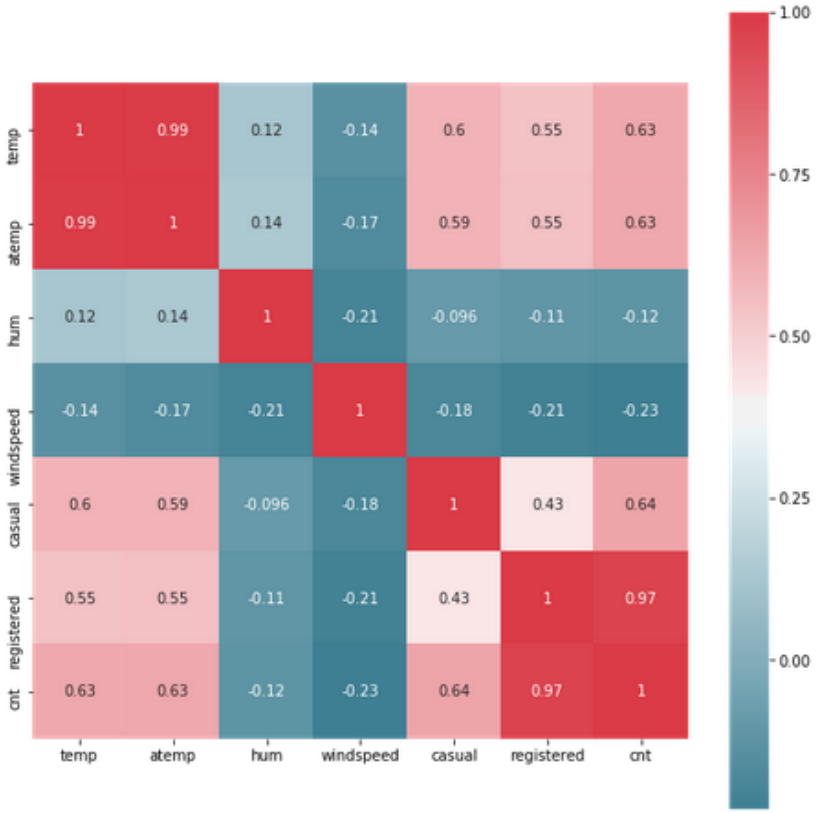


Figure6: correlation matrix & graph of numerical variables

In our case, we see that “registered” and “casual” were highly correlated with each other and our Target Variable “cnt” which is right because their sum is equal to “cnt” as mentioned in problem statement. So here we ignore these two variables and check the correlation between remaining variables.

From Correlation Matrix, it revealed that,

1) temp (temperature) and atemp (ambient temperature) are highly positively correlated. One of them should be removed before modeling step.

2) “cnt” has a positive and strong relationship with temp and atemp which is logical. People tends to rent more bikes when temperature is higher.
3) “cnt” has negative relationship with hum (humidity) and windspeed. People tends to rent less bike when there is more humidity and wind speed.

Here we drop the atemp variable as it highly positively correlated with temp variable.

2.2 Exploratory Data Analysis

2.2.1 Bivariate Analysis

In bivariate analysis, we check the relationship between our target variables and predictor variables. First we will check this for continuous variables.

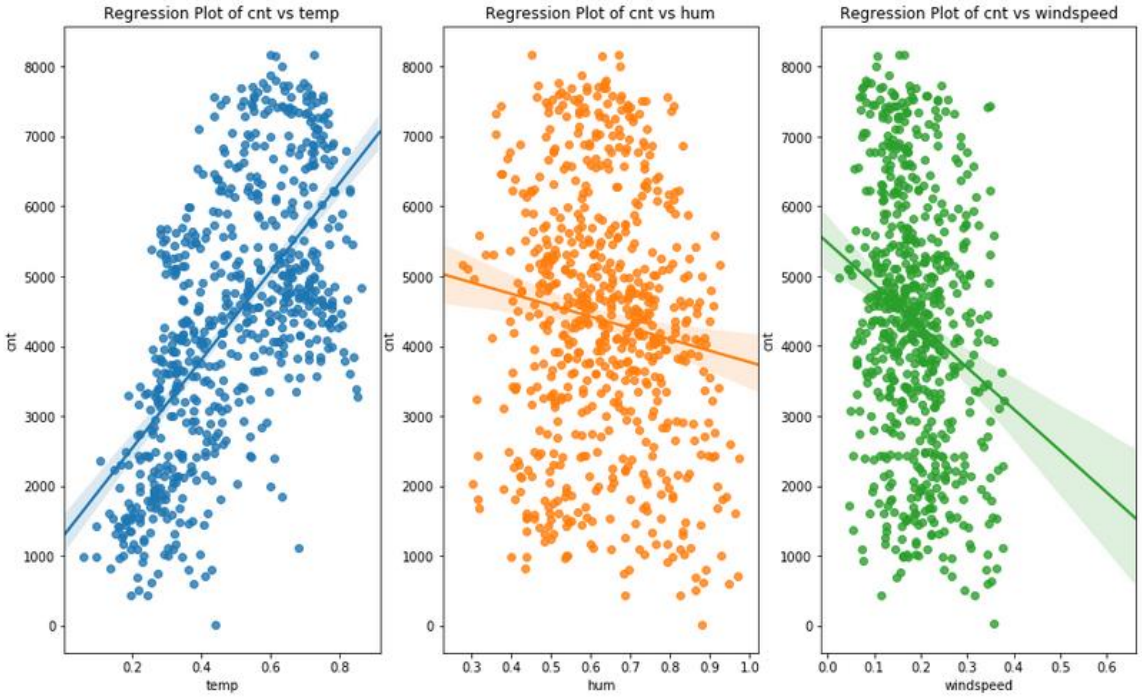


Figure7: Regression plot of Target Variables “cnt” vs Predictors temp, hum, & windspeed

From above regression scatter plot, we can say that,

1) ‘cnt’ and ‘temp’ have a positive and strong relationship. It means that when temperature increases the bike demand also increases.
2) ‘hum’ (humidity) has negative relationship with target variable ‘cnt’. It means if humidity decreases then bike demand increases.

3) ‘windspeed’ and ‘cnt’ have negative linear relationship. It means that decrease in windspeed will increase bike demand.

Now, let’s see check the relationship between target variable and categorical variables using boxplot.

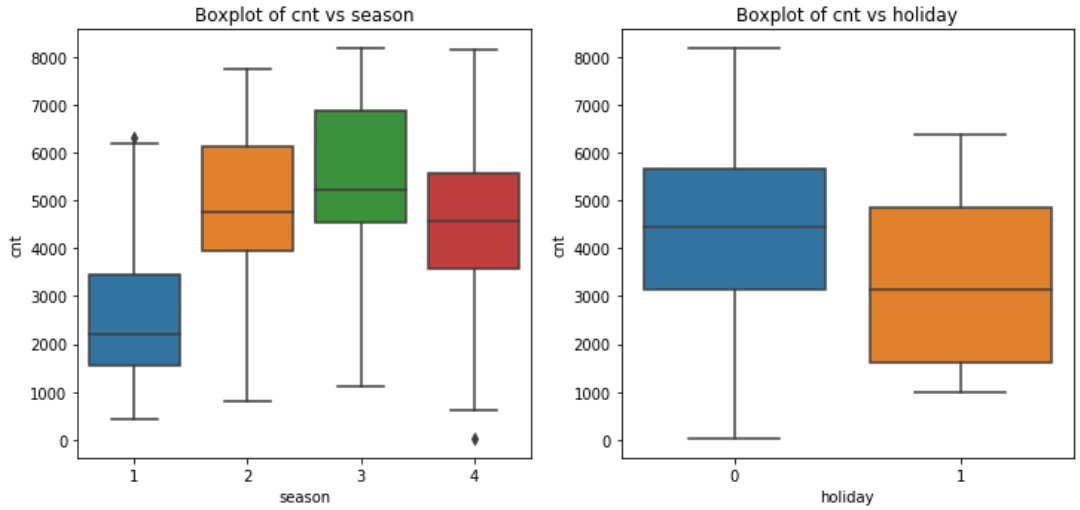


Figure8: Boxplot of cnt vs season and boxplot of cnt vs holiday

The boxplot of cnt vs season:

1) The cnt is lowest for spring season and highest for fall season.
2) There is not much different between cnt for summer and fall.

Boxplot for cnt vs holiday, it shows that less number of bikes rent on holiday.

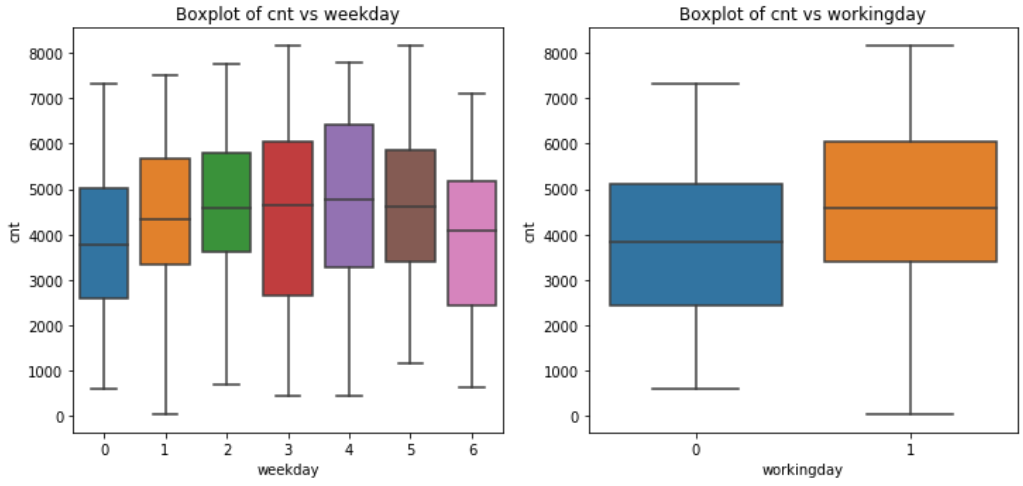


Figure9: Boxplot of cnt vs weekday and boxplot of cnt vs workingday

The Boxplot of cnt vs weekday, there are nearly similar on all weekdays, but higher on Friday.

Boxplot of cnt vs workingday, the more number of bikes are used on Workingday.

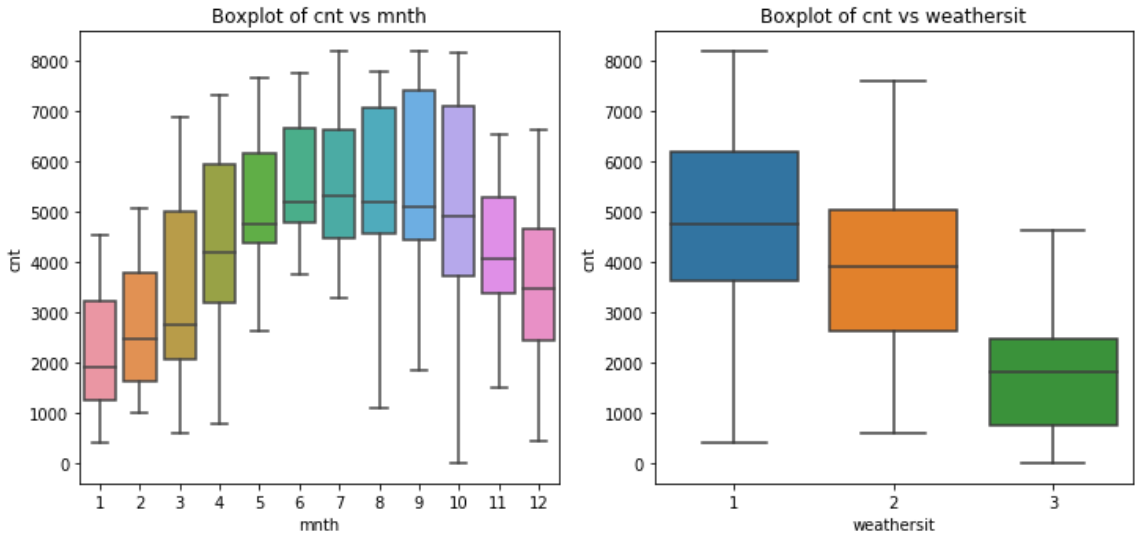


Figure10: Boxplot of cnt vs mnth and boxplot of cnt vs weathersit

The boxplot of cnt vs month, more bikes rent in July month.

Boxplot of cnt vs weathersit,

1) as per the graph more number of bikes are used when weather condition is Clear, Few clouds, Partly cloudy, Partly cloudy
2) and less number of bikes are used when weather condition is Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
3) No bikes where used when weather condition is Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Summary of Bivariate Analysis:

- More number of bikes are rent in good weather condition and less in bad weather condition
- More number of bikes are rented on weekdays.

2.2.2 Feature Scaling

Data Normalization is the process of rescaling one or more variables to the range of [0, 1], means smallest value of each variable is 0 and highest is 1. This is technique is a good to use when we know that our data distribution is not Gaussian.

As we know, the given dataset numerical variables are already normalized. To clarify using visualization techniques.

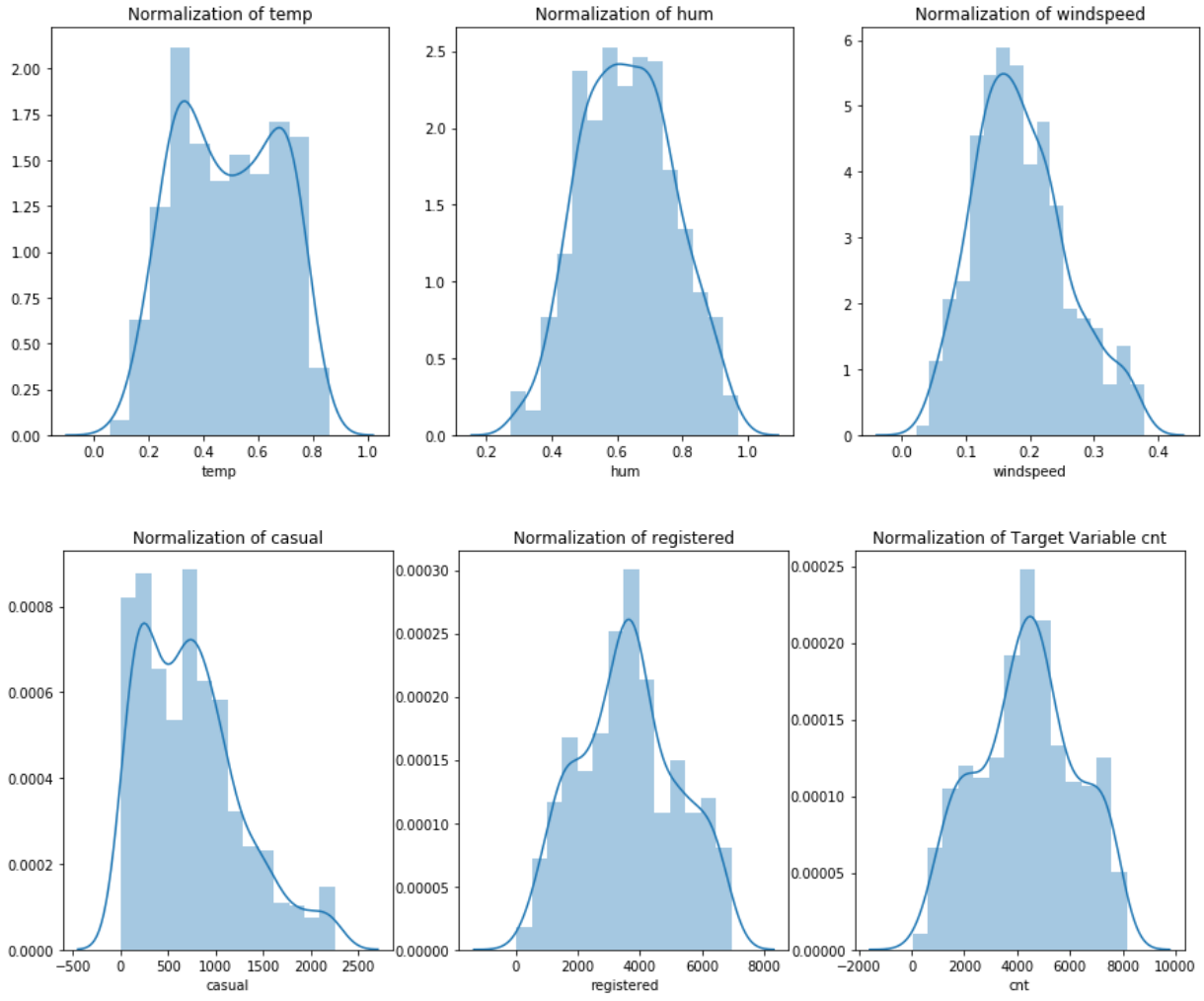


Figure11: Normalized distribution of predictor numerical variables and target variable

As we see from above graph, our given data is in proper scaling form. All the numerical predictor variables are normalized. Also, the target variable 'cnt' is close to normal distribution.

2.3 Modeling

After pre-processing and exploratory data analysis, the data was divided into training and test dataset with 80% and 20% ratio. Now it's time to implement various machine learning algorithms on our dataset.

In machine learning there is two main types:

- **Supervised Machine Learning:** knowledge of output. Target Variable is fix
- **Unsupervised Machine Learning:** No knowledge of Output. Self-Guided Learning Algorithms.

2.3.1 Model Selection

Selecting model is main Part of Modelling, We have various model algorithms some of the basic algorithms are:

- **Linear Regression :** Best suitable for Regression Model
- **Logistic Regression:** Suitable for Classification Model
- **Decision Tree:** Best suitable for Regression & Classification model
- **Random Forest:** Mostly used for Classification model analysis but can be used for Regression model
- **KNN algorithms:** Can be used for Regression and Classification model
- **Naive Bayes:** used for Classification Model

In our case of Bike Renting Prediction, we have to predict the number of bikes rent in single day based on environmental and seasonal conditions. The Target Variable 'cnt' is continuous in nature means we are dealing with regression problem.

That why, we are considering following machine learning algorithms

- Linear Regression
- Decision Tree Regression
- Random Forest Regression
- KNN Regression Algorithms.

2.3.2 Linear Regression Model

Linear Regression model is used to check the linear relationship between target variable and predictors.

Steps in Linear Regression Model:

1) Data is divided into train and test data in 80-20 ratio
2) Linear Regression is trained on train dataset.
3) Predict the output for test dataset by implementing trained linear regression model.
4) MAE, MSE, MAPE, and RMSE are used to check the performance of the model

Python Implementation

OLS Regression Results

Dep. Variable:	y	**R-squared:**	0.967
Model:	OLS	**Adj. R-squared:**	0.966
Method:	Least Squares	**F-statistic:**	1533.
Date:	Sun, 25 Aug 2019	**Prob (F-statistic):**	0.00
Time:	20:11:36	**Log-Likelihood:**	-4402.5
No. Observations:	538	**AIC:**	8825.
Df Residuals:	528	**BIC:**	8868.
Df Model:	10		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
season	536.8620	62.087	8.647	0.000	414.894	658.830
yr	2016.5647	75.024	26.879	0.000	1869.184	2163.946
mnth	-34.2763	19.569	-1.752	0.080	-72.719	4.166
holiday	-295.6698	246.102	-1.201	0.230	-779.128	187.789
weekday	82.2860	19.176	4.291	0.000	44.615	119.957
workingday	468.9121	87.145	5.381	0.000	297.719	640.105
weathersit	-774.2040	93.662	-8.266	0.000	-958.200	-590.208
temp	5225.0709	218.615	23.901	0.000	4795.609	5654.533
hum	421.6321	304.044	1.387	0.166	-175.652	1018.916
windspeed	-483.9963	449.417	-1.077	0.282	-1366.861	398.868

Omnibus:	88.484	**Durbin-Watson:**	2.001
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	184.950
Skew:	-0.904	**Prob(JB):**	6.90e-41
Kurtosis:	5.232	**Cond. No.**	103.

Figure12: summary of Linear Regression Model

Few things we learn from this output:

- season, yr, weekday, workingday, weathersit, temp have small p-values, whereas mnth, holiday, hum, windspeed have a larger p-values

- Here we reject the null-hypothesis for season, yr, weekday, workingday, weathersit, temp
    - There is assicoation between these variables and Target Variable cnt

- Fail to reject the null hypothesis for mnth, holiday, hum, windspeed
    - There is no association between these variables and Target Variable cnt

R-squared (0.967) means this model provides best fit for the given data

2.3.3 Decision Tree Regression Model

In decision tree regression algorithms the predictions is based on branching series of Boolean tests. There are number of different types of decision trees that can be used in Machine Learning Algorithms. Decision Tree is a rule. Each branch connects nodes with "and" and multiple braches are connected by "or".

Steps in Decision Tree Regression Model:

1) Data is divided into train and test data in 80-20 ratio
2) Decision Tree Regression is trained on train dataset.
3) Predict the output for test dataset by implementing trained decision tree regression model.
4) MAE, MSE, MAPE, and RMSE are used to check the performance of the model

2.3.4 Random Forest Regression Model

Random Forest is an ensemble that consists of many decision trees. The method combines Breiman's "bagging" idea and random selection of features. This algorithms can be used for regression problem

Steps in Random Forest Regression Model:

1) Data is divided into train and test data in 80-20 ratio
2) Random Forest Regression is trained on train dataset.
3) Predict the output for test dataset by implementing trained random forest regression model.
4) MAE, MSE, MAPE, and RMSE are used to check the performance of the model

2.3.5 KNN Regression Algorithms

K stands for K-Nearest Neighbors. It is the simple algorithms that scores all variable cases and classifies new cases based on similarity measures. It is the lazy learning algorithms

Steps in KNN Regression Model:

1) Data is divided into train and test data in 80-20 ratio
2) KNN Regression is trained on train dataset.
3) Predict the output for test dataset by implementing trained KNN forest regression model.
4) MAE, MSE, MAPE, and RMSE are used to check the performance of the model

Chapter 3: Conclusion

3.1 Model Evaluation

The Regression Model is evaluated by Error Metrics of Regression. There are various types of error measures to evaluate the model.

Mean Absolute Error (MAE): is the mean of the absolute value of the errors: In [0,∞), the smaller the better

Mean Squared Error (MSE): is the mean of the squared errors: In [0,∞), the smaller the better

Mean Absolute Percent Error (MAPE): is the mean of the absolute percent value of the errors: In [0,1), the smaller the better

Root Mean Squared Error (RMSE) :is the square root of the mean of the squared errors: In [0,∞), the smaller the better.

- MAE gives less weight to outliers means it is not sensitive to outliers.
- MAPE is similar to MAE, but normalized the true obeservations. When true observation is zero then this metric will be problematic
- MSE is a combination measurement of bias and variance of predictions. It is more popular.
- RSME is square Root of MSE, Root square is taken to make the units of the error be the same as the units of the target. This measure gives more weight to large deviations such as outliers, since large differences squared become larger and small (smaller than 1) differences squared become smaller.

Selection: Out-off these 4 error metrics, MSE and RMSE are mainly used for Time-Series dataset. As we know, current working data is not a time dependent or time-series data.

For that Reason the Model Evaluation is based on **MAPE Error Metrics**

3.2 Performance Measures

3.2.1 R Implementation

For measuring the performance of various regression model, we considering MAPE Error Metrics using the metric package in R. Now we compare the MAPE for regression model

Table 3: Comparing MAPE Error Metrics in Various Regression Algorithms in R

Model	MAPE
Linear Regression	0.1684646
Decision Tree Regression	0.2316771
Random Forest Regression	0.1258508
KNN Regression Algorithms	0.2524476

From the above table, we see that Random Forest Regression performing better than other regression algorithms in MAPE error metrics.

3.2.2 Python Implementation

In python, MAPE error metric was calculated by creating MAPE function. Here no pre-built package is available.

Table 4: Comparing MAPE Error Metrics in Various Regression Algorithms in Python

Model	MAPE
Linear Regression	0.17278869
Decision Tree Regression	0.26101719
Random Forest Regression	0.12905417
KNN Regression Algorithms	0.18841675

As we see, Random Forest Regression performing better than other regression algorithms.

3.3 Model Selection

From the error metric we can see that Random Forest Regression is performing better than other regression algorithms in both R and Python. The result we get for Random Forest Regression is somewhat similar in both R and Python. But R implementation is performing better than python for linear, decision tree and random tree regression algorithms.

The final model selection, we will go with the Random Forest Regression Model to get more precise predictions as they have low MAPE error metrics.

Appendix

R Implementation graph plot

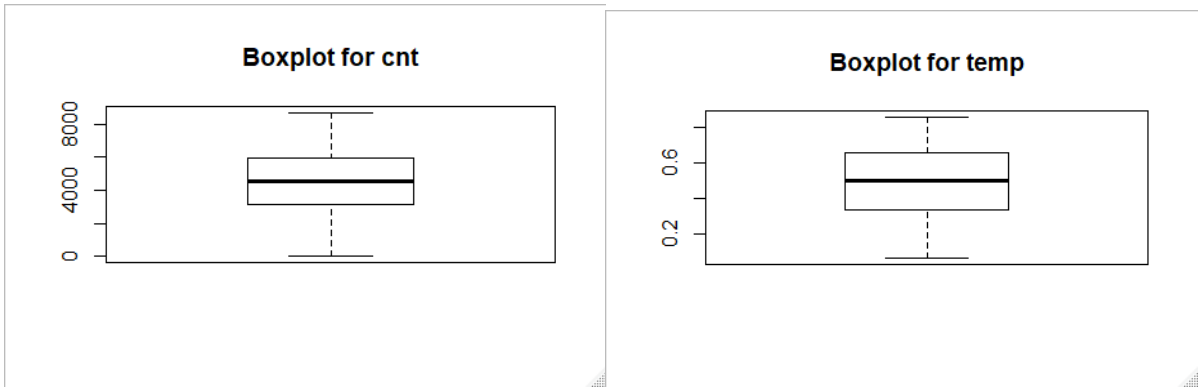


Figure13: Boxplot of cnt and boxplot of temp

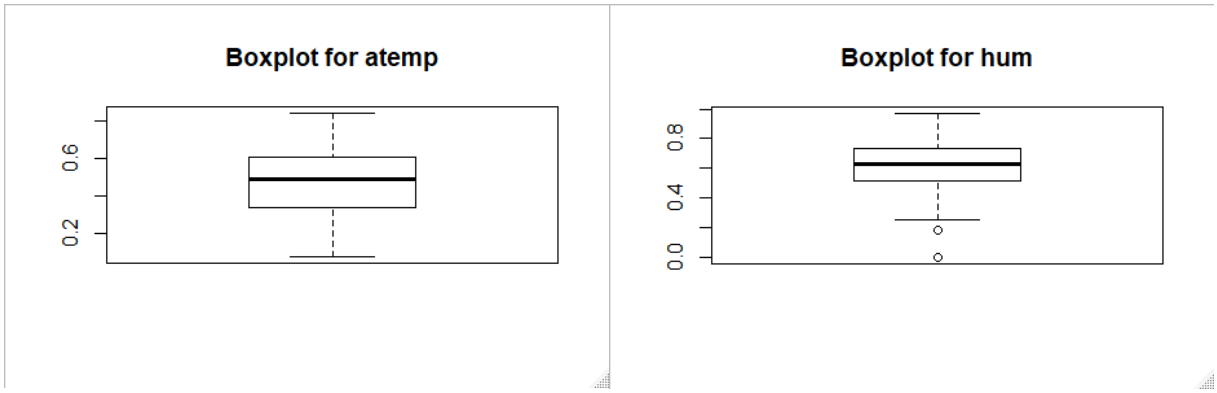


Figure14: Boxplot of atemp and boxplot of hum

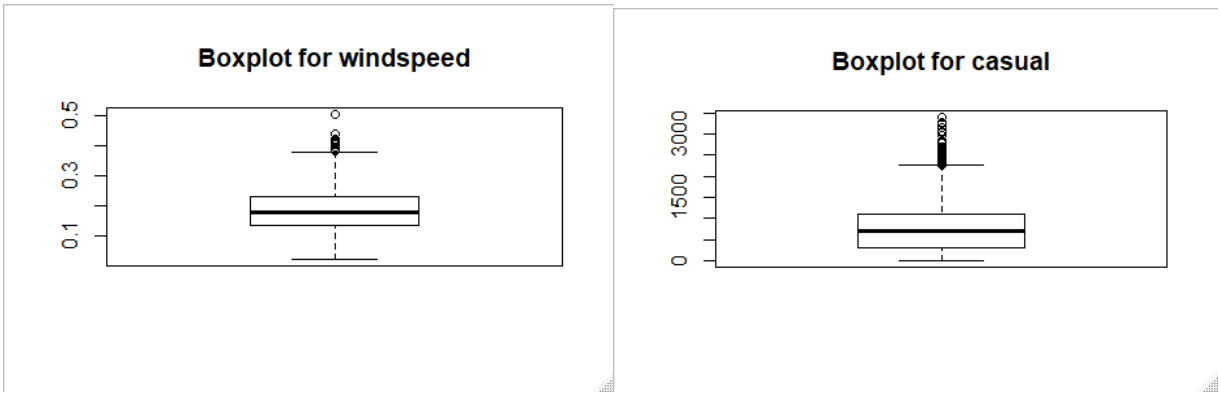


Figure15: Boxplot of windspeed and boxplot of casual

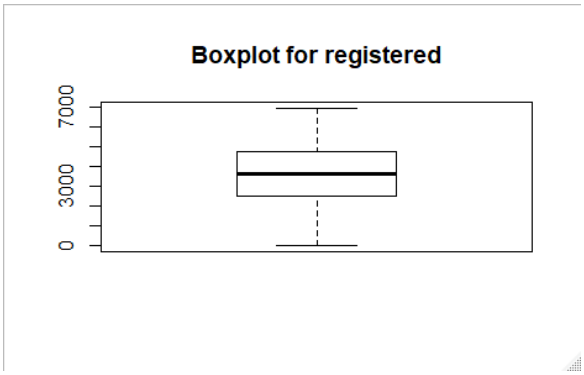


Figure16: Boxplot of cnt registered

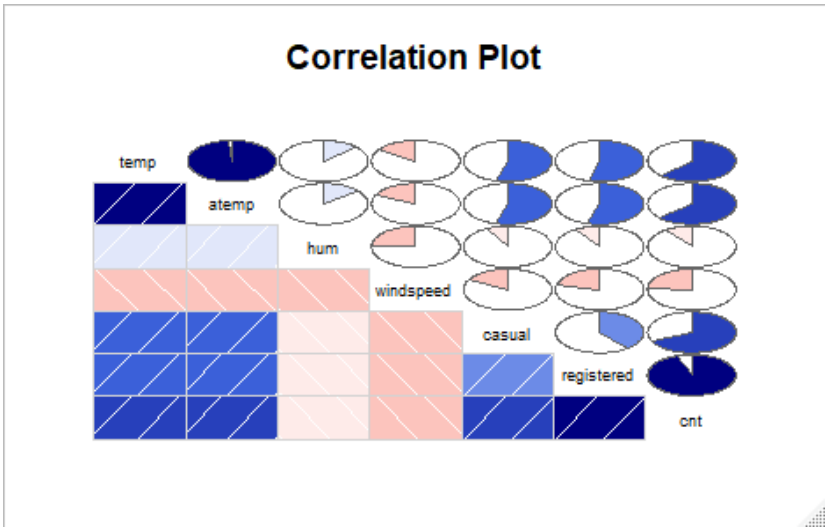


Figure17: Correlation graph in R

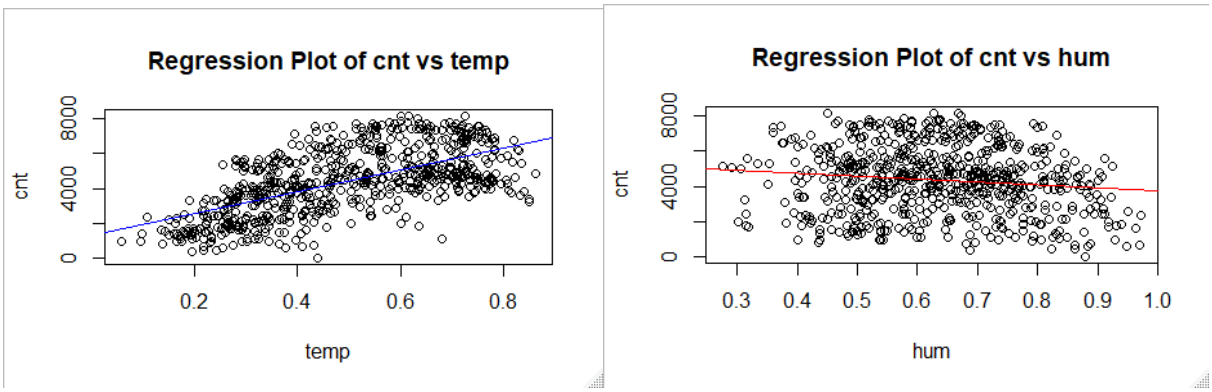


Figure18: Regression plot of cnt vs temp and regression plot of cnt vs hum

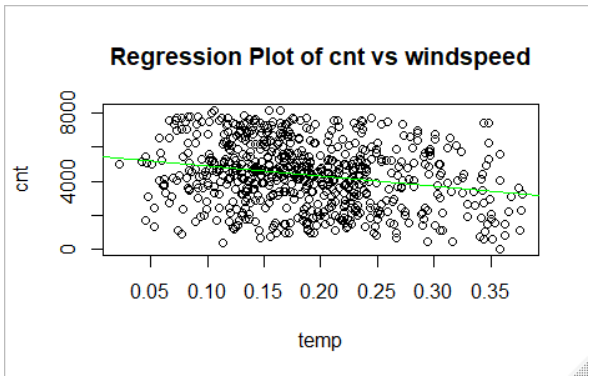
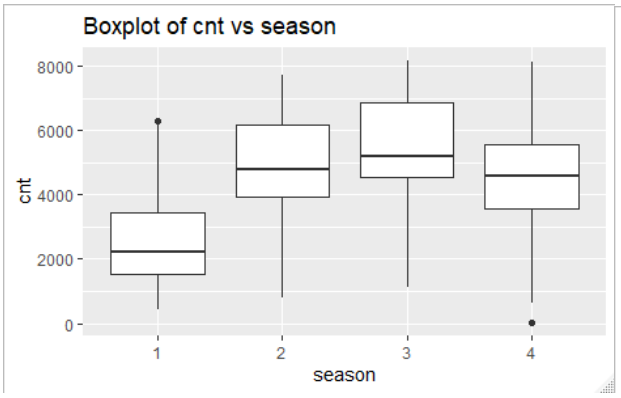
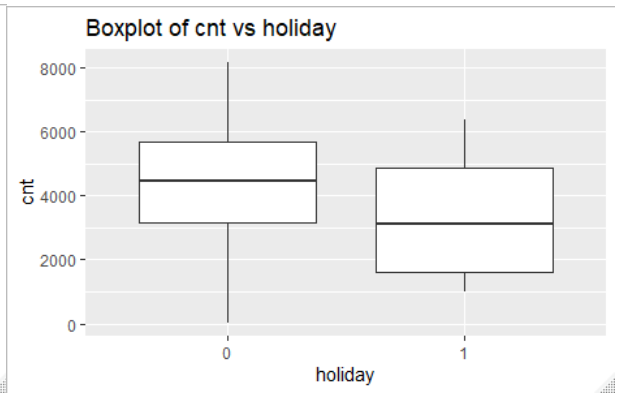
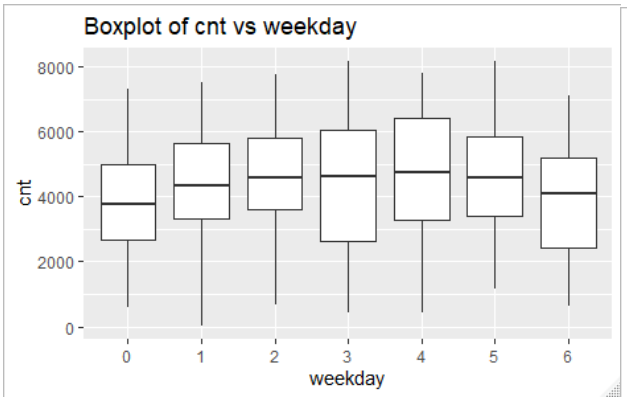
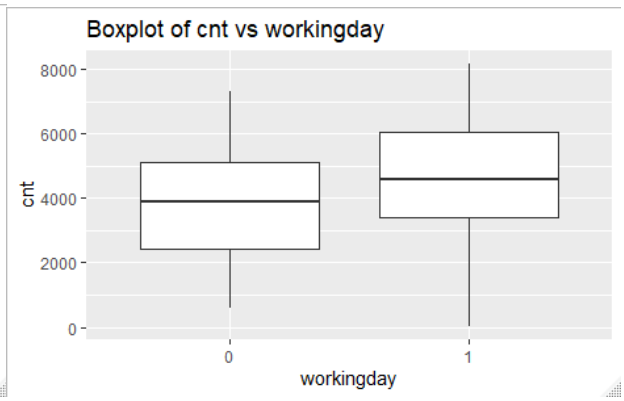


Figure19: Regression plot of cnt vs windspeed









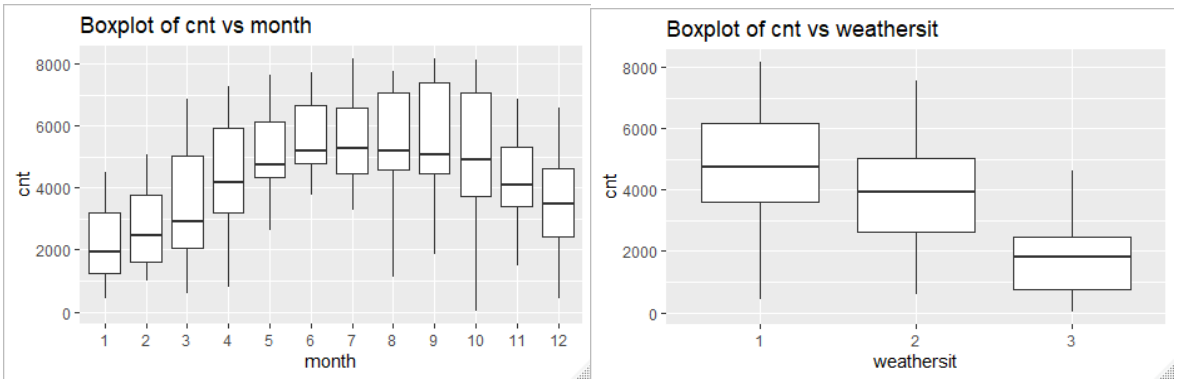


Figure20: Bivariate Boxplot of Target Variables vs Categorical Variables

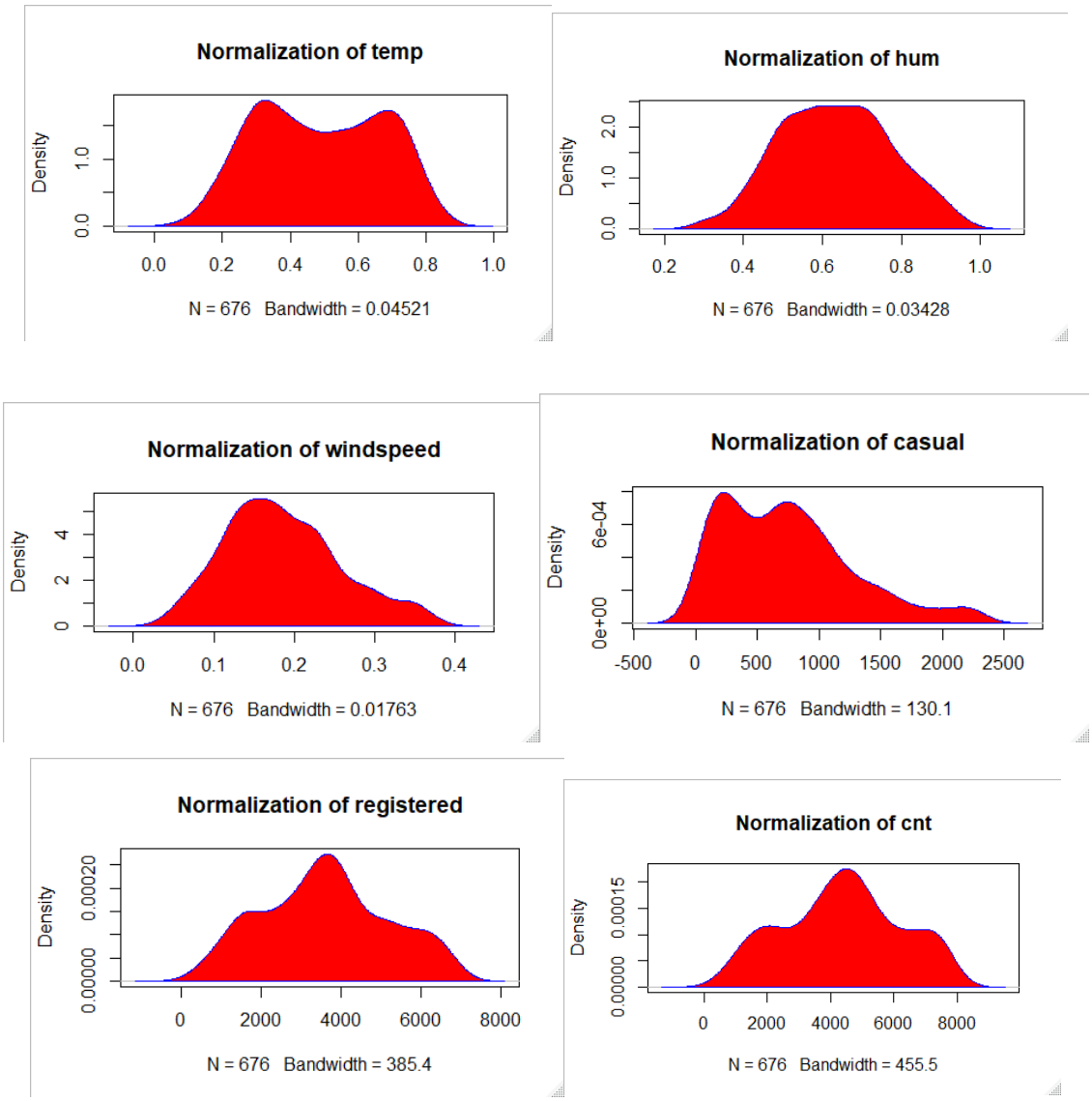


Figure21: Normalization Graph

Code

R Implementation Code

```
#remove all program/objects from RAM
rm(list=ls())

#set Working directory
setwd("D:/Bike Renting")

#cheking current working directory
getwd()

#before starting this project with dataset.
#Initially I open given file in excel and just look the data available
#starting the project We need to load some libraries to deal with this data
install.packages(c("lattice", "dplyr", "plyr", "ggplot2", "corrgram"))
install.packages("contrib.url")

#install required libraries
library("dplyr")
library("plyr")
library("ggplot2")
library("corrgram")
library("lattice")

#load required dataset
data = read.csv("day.csv", header = T)

############################################################################
#########################
# Understanding the data
############################################################################
########################
#after loading the dataset
#let's check the number of variables and obseravtions in dataset
dim(data)
#we see that there are 731 observations(Rows) and 16 variables(Columns) in given dataset.

#checking first few rows of data
head(data,10)
#in this "cnt" is our Target Variable and others are predictor variables
```

```
37.
38. #check the summary of the data
39. summary(data)
40.
41. #it shows that varable like "yr", "mnth", "holiday", "weekday", "workingday", "weathershit"
42. #are categorical variables and already encoded.
43.
44. #numerical variables like "temp", "atemp", "hum", windspeed" are already converted into
45. #noramlized form as that mentioned into problem statement.
46.
47. #cheking data structure
48. str(data)
49.
50. #as categorical variables shows the int datatypes. So we need to convert them into proper datatype
51. #i.e. factor variable
52.
53. #changing dtype of categotical variables
54. data$season = as.factor(data$season)
55. data$yr = as.factor(data$yr)
56. data$mnth =  as.factor(data$mnth)
57. data$holiday =  as.factor(data$holiday)
58. data$weekday =  as.factor(data$weekday)
59. data$workingday =  as.factor(data$workingday)
60. data$weathersit = as.factor(data$weathersit)
61.
62. #checking data structure again
63. str(data)
64.
65. #in this dataset, some of the variables are not useful for further analysis for that reason
66. #we going to dropped that variables
67. ######### Dropping Variables which are not required
68. ############## instant: index number which is not useful in analysis
69. ############## dteday: all required parameters are already extracted from this variable such as
70. ######################## yr, mth, weekday. So this variable is not useful
71.
72. #dropping instant & dteday variables
73. data = select(data, -c(instant, dteday))
74.
75. #check dataset after dropping variables
76. dim(data)
77. head(data,10)
78.
```

```
79. ##############################################################################
    ##########################
80. #Missing Value Analysis
81. ##############################################################################
    ##########################
82. #checking number of missing values in dataset
83. sum(is.na(data))
84.
85. #checking variables with missing values count
86. sapply(data, function(x) sum(is.na(x)))
87.
88. #there is no missing values in dataset
89.
90. ##############################################################################
    ##########################
91. #Outlier Analysis
92. ##############################################################################
    ##########################
93. #here, we use Boxplot Method to visualize the ouliers in our dataset
94.
95. #plotting boxplot for "cnt" variable
96. boxplot(data$cnt, data=data, main = "Boxplot for cnt")
97.
98. #after checking the Boxplot for cnt, it is evident that there is no outlier present in cnt
99.
100. #let's check the outliers of predictor variables such as temp, atemp, hum, winspeed, casual, registered
101.
102. #plotting boxplot for "temp" variable
103. boxplot(data$temp, data=data, main = "Boxplot for temp")
104. #no outliers in temp variable
105.
106. #plotting boxplot for "atemp" variable
107. boxplot(data$atemp, data=data, main = "Boxplot for atemp")
108. #no outliers in atemp variable
109.
110. #plotting boxplot for "hum" variable
111. boxplot(data$hum, data=data, main = "Boxplot for hum")
112. #outliers preset in hum variable, that we will remove after checking all variables ouliter
113.
114. #plotting boxplot for "windspeed" variable
115. boxplot(data$windspeed, data=data, main = "Boxplot for windspeed")
116. #outliers preset in windspeed variable, that we will remove after checking all variables ouliter
```

```
117.
118. #plotting boxplot for "causal" variable
119. boxplot(data$casual, data=data, main = "Boxplot for casual")
120. #outliers preset in casual variable, that we will remove after checking all variables ouliter
121.
122. #plotting boxplot for "registered" variable
123. boxplot(data$registered, data=data, main = "Boxplot for registered")
124. #no outliers in registered variable
125.
126. #as we see there are some outliers present in "hum", "windspeed" and "casual" variables
127. #varaibles hum, windspeed, casual has ouliers and that have to be remove by
128. #outlier removal method
129.
130. #make copy of data if anything wrong goes
131. df = data
132. data =df
133. ########### Treating Outliers ##############################
134. #findout the numeric variables in dataset
135. numeric_index = sapply(data, is.numeric)
136.
137. #prepare the numeric dataset
138. numeric_data = data[, numeric_index]
139.
140. cnames = colnames(numeric_data)
141.
142. #loop to remove outliers from all variables
143. for(i in cnames){
144.   qnt = quantile(data[i], probs = c(.75, .25), na.rm = T)
145.   iqr = qnt[1] - qnt[2]
146.   min = qnt[2] - (iqr*1.5)
147.   max = qnt[1] + (iqr*1.5)
148.   print(min)
149.   print(max)
150.   data = subset(data, data[i]>min)
151.   data = subset(data, data[i]<max)
152. }
153.
154. #there are 55 observations are dropped in outliers
155.
156. ##############################################################################
     ########################
157. #correlation Analysis
```

```
158. ##########################################################################
     ########################
159. #Correlation Analysis: Here we generating correlation matrix to understand
160. #how the each variable related with each other. In that we plotting correlation matrix
161. #and generate plot using corrgram library for better understanding
162.
163. #this metrix wil be plot only using numeric data for that we provide numeric data
164. corr = cor(numeric_data)
165. print(corr)
166.
167. #plotting correlation plot using corrgram library
168. corrgram(numeric_data, order = F, upper.panel = panel.pie, text.panel = panel.txt, main = "Correlation Plot
     ")
169.
170. #The above correlation analysis shows that,
171. #temp and atemp are highly correlated
172. #temp and atemp have positive and strong correlation with cnt
173. #hum and windspeed have negative and weak correlation with cnt
174.
175. #dropping atemp variable from a dataset
176. data = select(data, -c(atemp))
177.
178. ##########################################################################
     ########################
179. # Exploratory Data Analysis
180. ##########################################################################
     ########################
181.
182. ################################ Bivariate Analysis ##############################
     ###############
183. #finding the relationship between Numerical variables "temp", "hum", "windspeed" with
184. #target variable cnt
185.
186. #relation between cnt vs temp
187. attach(data)
188. plot(temp, cnt, main="Regression Plot of cnt vs temp",
189.      xlab="temp", ylab="cnt")
190. abline(lm(cnt~temp), col="blue") # regression line (y~x)
191.
192. #relation between cnt vs hum
193. plot(hum, cnt, main="Regression Plot of cnt vs hum",
194.      xlab="hum", ylab="cnt")
```

```
abline(lm(cnt~hum), col="red") # regression line (y~x)

#relation between cnt vs windspeed
plot(windspeed, cnt, main="Regression Plot of cnt vs windspeed",
    xlab="temp", ylab="cnt")
            abline(lm(cnt~windspeed), col="green") # regression line (y~x)

            #after plotting above graph, we see that
### cnt has positive linear relationship with temp
            ### on the otherside, cnt has negative linear relationship with windspeed
### but hum(humidity) has a little neagive relationship with cnt

#now let's find the relationship between categorical variables and Target Variables "cnt"
            #categorical variables are "season", "holiday", "weekday", "Workingday", "weathershit", "month"

#plotting boxplot for cnt vs season variable
ggplot(data, aes(x=season, y=cnt)) +
  geom_boxplot()+xlab("season")+ylab("cnt")+ggtitle("Boxplot of cnt vs season")
##cnt is very low in Spring Season and cnt is large in Fall Season

#plotting boxplot of cnt vs holiday
ggplot(data, aes(x=holiday, y=cnt)) +
  geom_boxplot()+xlab("holiday")+ylab("cnt")+ggtitle("Boxplot of cnt vs holiday")
##cnt is more on weekday i.e. no holiday

            #plotting boxplot of cnt vs weekday
ggplot(data, aes(x=weekday, y=cnt)) +
  geom_boxplot()+xlab("weekday")+ylab("cnt")+ggtitle("Boxplot of cnt vs weekday")
#as per the graph more number of bikes are used on Friday

#plotting boxplot of cnt vs workingday
ggplot(data, aes(x=workingday, y=cnt)) +
  geom_boxplot()+xlab("workingday")+ylab("cnt")+ggtitle("Boxplot of cnt vs workingday")
#as per the graph more number of bikes are used on Workingday,
#this conclusion we already get from boxplot of cnt vs holiday

#plotting boxplot of cnt vs month
ggplot(data, aes(x=mnth, y=cnt)) +
  geom_boxplot()+xlab("month")+ylab("cnt")+ggtitle("Boxplot of cnt vs month")
#as per the graph more number of bikes are used in July Month of year

#plotting boxplot of cnt vs weathersit
```

```
ggplot(data, aes(x=weathersit, y=cnt)) +
 geom_boxplot()+xlab("weathersit")+ylab("cnt")+ggtitle("Boxplot of cnt vs weathersit")
#1 >>as per the graph more number of bikes are used when weather condition is Clear, Few clouds,
            #Party cloudy, Partly cloudy
#2 >> and less number of bikes are used when weather condition is
#Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
#3 >> No bikes where used when weather condition is
#Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

##########Summary##############
## cnt is maximum in good weather condition and minimum in bad weather condition
## more number of bikes are rented on weekdays.

#############################################################################
#########################
#Feature Scalling
#############################################################################
#########################
#as we know that, most of given variables are already Normalised
#But let's clarify that using Visualization techniques

#check whether variable "temp" is normal or not
plot(density(data$temp), main="Normalization of temp")
polygon(density(data$temp), col="red", border="blue")

            #check whether variable "hum" is normal or not
plot(density(data$hum), main="Normalization of hum")
polygon(density(data$hum), col="red", border="blue")

#check whether variable "windspeed" is normal or not
plot(density(data$windspeed), main="Normalization of windspeed")
polygon(density(data$windspeed), col="red", border="blue")

#check whether variable "casual" is normal or not
plot(density(data$casual), main="Normalization of casual")
polygon(density(data$casual), col="red", border="blue")

#checking whether varianle "registered" is normal or not
plot(density(data$registered), main="Normalization of registered")
polygon(density(data$registered), col="red", border="blue")

#checking whether variable "cnt" is normal or not
```

```
plot(density(data$cnt), main="Normalization of cnt")
polygon(density(data$cnt), col="red", border="blue")

        #As we can see, our data is in proper scalling form. Numerical Predictor Variables are Normalized
#Also our Target Variable 'cnt' is also close to Normal Distribution

#############################################################################
#Modeling & Prediction
#############################################################################
##Data Cleaning is done! Now lets bulid the model and predict the results

        #In machine Learning there is Two main types:

        #Supervised Machine Learning : knowledge of output. Target Variable is fix
#Unsupervised Machine Learning: No knowledge of Output. Self Guided Learnig Algorithms.

#Selecting model is main Part of Modelling, We have various model algorithms some of
#the basic algorithms are:

#Linear Regression : Best suitable for Regression Model
#Logistic Regression: Suitable for Classification Model
#Decision Tree: Best suitable for Regression & Classification model
#Random Forest: Mostly used for Classification model analysis but can be use for Regression model
        #KNN algorithms: Can be used for Regression and Classification model
#Naive Bayes: used for Classification Model

#in our given dataset, Target Variable "cnt" is Numerical Continuous Variable. So we are dealing
#with Regression Model Analysis.

        #for that reason we are considering following algorithms
#1 >> Linear Regression
        #2 >> Decision Tree
        #3 >> Random Forest
#4 >> KNN Algorithms

#before moving further let's droped the casual and registered variables because there sum is
#equal to target variable i.e. "cnt"
data = select(data, -c(casual, registered))

#Now to predict the model, to simulate a train and test set, we are going to split randomly
```

```
#this train dataset into 80% train and 20% test.

df1 = data
#Random Sample indexes
train_index = sample(1:nrow(data), 0.8*nrow(data))
test_index = setdiff(1:nrow(data), train_index)

#Build train_X, train_y, test_X, test_y
train_X = data[train_index, -11]
train_y = data[train_index, "cnt"]

test_X = data[test_index, -11]
test_y = data[test_index, "cnt"]

################################################################################
########################
#Linear Regression Model
################################################################################
########################
install.packages("usdm")
library(usdm)
model_LR = lm(train_y ~ ., data=train_X)

summary(model_LR)

#predict the output for test_X dataset
predict_LR = predict(model_LR, test_X)

#A few things, we learn from this output
#season, yr, weekday, workingday, weathersit, temp have small p-values,
#where as mnth, holiday, hum, windspeed have a larger p-values
#Here we reject the null-hypothesis for season, yr, weekday, workingday, weathersit, temp
#There is assicoation between these variables and Target Variable cnt

#Fail to reject the null hypothesis for mnth, holiday, hum, windspeed
#There is no association between these variables and Target Variable cnt

#R-squared (0.8652) means this model provides best fit for the given data
#but Selecting the model with the highest R-squared is not a
#reliable approach for choosing the best linear model.

#####Better Solution is############
```

```
#Do model Evaluation based on the Error Metrics for Regression:

#For classification problems, we have only used classification accuracy as our evaluation metric.
        #But here we used Error Metrics to evaluate the model

#Mean Absolute Error (MAE): is the mean of the absolute value of the errors:
#In [0,???), the smaller the better

#Mean Squared Error (MSE): is the mean of the squared errors: In [0,???), the smaller the better

#Mean Absolute Percent Error (MAPE): is the mean of the absolute percent value of the errors:
#In [0,1), the smaller the better

#Root Mean Squared Error (RMSE) :is the square root of the mean of the squared errors:
 #In [0,???), the smaller the better

#Let's calculate these by hand, to get an intuitive sense for the results:
install.packages("Metrics")

library(Metrics)

# calculate MAE, MSE, MAPE, RMSE
mae(test_y, predict_LR) #503.6793
        mse(test_y, predict_LR) #473119.3
mape(test_y, predict_LR) #0.1684646
rmse(test_y, predict_LR) #687.8367

####
#MAE gives less weight to outliers means it is not sensitive to outliers.
#MAPE is similar to MAE, but normalized the true obeservations. When true observation is zero
#then this metric will be problematic
        #MSE is a combination measurement of bias and variance of predictions. It is more popular.
#RSME is square Root of MSE, Root square is taken to make the units of the error be the same as
        #the units of the target. This measure gives more weight to large deviations such as outliers,
 #since large differences squared become larger and small (smaller than 1)
#differences squared become smaller.

#Selection: Outoff these 4 error metrics, MSE and RMSE are mainly used for Time-Series dataset.
#As I know, current working data is not a time dependend or time-series data.

#for that Reason the Model Evaluation is based on MAPE Error Metrics

```

```
399. ##########################################################################
     ######################
400.             #Decision Tree
401. ##########################################################################
     ######################
402.             #load libraries
403. library(rpart)
404.             library(MASS)
405.
406.             #use rpart for Decision Tree regression
407. model_DT = rpart(train_y ~ ., data=train_X, method = "anova")
408.
409.             summary(model_DT)
410.
411. #predict the output for test_X dataset
412. predict_DT = predict(model_DT, test_X)
413.
414. # calculate MAE, MSE, MAPE, RMSE
415. mae(test_y, predict_DT) #652.6066
416. mse(test_y, predict_DT) #722906.5
417. mape(test_y, predict_DT) #0.2316771
418. rmse(test_y, predict_DT) #850.2391
419.
420.             ##########################################################################
     ############################
421. #Random Forest
422. ##########################################################################
     ######################
423. #load libraries
424. library(randomForest)
425.
426. #use randomForest for Random Forest regression
427. model_RF = randomForest(train_y ~ ., data=train_X, importance = TRUE, ntree = 50)
428.
429. summary(model_RF)
430.
431. #predict the output for test_X dataset
432. predict_RF = predict(model_RF, test_X)
433.
434. # calculate MAE, MSE, MAPE, RMSE
435. mae(test_y, predict_RF) #295.5393
436. mse(test_y, predict_RF) #207524.9
```

```
mape(test_y, predict_RF) #0.1258508
rmse(test_y, predict_RF) #455.549

          ######################################################################
#############################
#KNN Algorithms
############################################################################
#######################
#load library
library(caret)

#use knnreg for KNN regression algorithms
model_KNN = knnreg(train_y ~.,data=train_X, k=5)

summary(model_KNN)

#predict the output for test_X dataset
predict_KNN = predict(model_KNN, test_X)

# calculate MAE, MSE, MAPE, RMSE
mae(test_y, predict_KNN) #735.0471
mse(test_y, predict_KNN) #866782.7
mape(test_y, predict_KNN) #0.2524476
rmse(test_y, predict_KNN) #931.0117

          ######################################################################
##############################
#Selecting Best Fit model for Future Analysis
############################################################################
#######################

#We are cosidering the MAPE for model evaluation becasue,
#it calculate average absolute percent error for
#each time period minus actual values divided by actual values.

#Reason we already discussed, let's explain again:

####Selection: Outoff these 4 error metrics,
#MSE and RMSE are mainly used for Time-Series dataset. As we know,
#current working data is not a time dependend or time-series data.

#Random Forest Model has smallest error metrics i.e.
```

```
# MAPE = 0.1258508

#So, for further analysis We are selecting Random Forest Model.
```

Python Implementation Code

```
#before starting this project with dataset. Initially I open given file in excel and just look the data available

#starting the project we need to load some libraries to deal with this data

import pandas as pd #for data processing & I/O operations
import numpy as np #for mathematical calculations
import seaborn as sns #for data visualization
import matplotlib.pyplot as plt #for plotting graphs

import sklearn #for machine learning algorithms

import os #for setting directory, I/O file operations

#setting working directories
os.chdir("D:\Bike Renting")

#checking the file directory
os.getcwd()

#load required dataset
data = pd.read_csv("day.csv")

#after loading dataset
#let's check the number of variables and observations in dataset
data.shape

#now explore dataset more
#checking few dataset
data.head()

#before moving further let's check the dtypes of each variables
data.dtypes

#converting to categorical variables

#so to convert muliple variables in one go, we need to create a loop function
```

```
37.
38. #create a variable and store all variables
39. cat_var = ["season", "yr", "mnth", "holiday", "weekday", "workingday", "weathersit"]
40.
41. for i in cat_var:
42.     data[i] = data[i].astype('category')
43.
44. #checking dtypes again
45. data.dtypes
46.
47. #dropping instant and dteday variables
48. data = data.drop(['instant', 'dteday'], axis = 1)
49.
50. #checking the dataset after dropping two variables
51. data.head()
52.
53. #checking the missing values using isnull function
54.
55. data.isnull().sum().sort_values(ascending= False)
56.
57. #check summary of the dataset
58. data.describe()
59.
60. #checking outliers in Target Variable "cnt" using boxplot method
61.
62. #plotting boxplot for cnt variable
63. sns.boxplot(data =data, y="cnt", orient ='v')
64.
65. #let's check the outliers of predictor variables such as temp, atemp, hum, windspeed, casual, registered
66.
67. fig, axes = plt.subplots(nrows=2, ncols= 3)
68.
69. fig.set_size_inches(18,12)
70.
71. #plotting boxplot of temp variable
72. sns.boxplot(data['temp'], orient ='v', ax=axes[0][0]).set_title("Boxplot of temp")
73.
74. #plotting boxplot of atemp variable
75. sns.boxplot(data['atemp'], orient ='v', ax=axes[0][1]).set_title("Boxplot of atemp")
76.
77. #plotting boxplot of hum variable
78. sns.boxplot(data['hum'], orient ='v', ax=axes[0][2]).set_title("Boxplot of hum")
```

```
79.
80. #plotting boxplot of windspeed variable
81. sns.boxplot(data['windspeed'], orient ='v', ax=axes[1][0]).set_title("Boxplot of windspeed")
82.
83. #plotting boxplot of casual variable
84. sns.boxplot(data['casual'], orient ='v', ax=axes[1][1]).set_title("Boxplot of casual")
85.
86. #plotting boxplot of registered variable
87. sns.boxplot(data['registered'], orient ='v', ax=axes[1][2]).set_title("Boxplot of registered")
88.
89. #as we see that there are some outliers value present in 'hum', 'windspeed', 'casual' variables.
90.
91. #before outlier removal lets findout the correlation analysis of these variables with target variables
92.
93. print(data['hum'].corr(data['cnt']))
94.
95. print(data['windspeed'].corr(data['cnt']))
96.
97. print(data['casual'].corr(data['cnt']))
98.
99. fig, (ax1,ax2,ax3) = plt.subplots(ncols= 3)
100.
101. fig.set_size_inches(18,6)
102.
103. #Correlation between 'hum' and 'cnt'  before  removal of  outliers
104. sns.regplot(x="hum", y="cnt", data=data, ax=ax1).set_title("Regression Plot of cnt vs hum")
105.
106. #Correlation between 'windspeed' and 'cnt'  before  removal of  outliers
107. sns.regplot(x="windspeed", y="cnt", data=data, ax=ax2).set_title("Regression Plot of cnt vs windspeed")
108.
109. #Correlation between 'casual' and 'cnt'  before  removal of  outliers
110. sns.regplot(x="casual", y="cnt", data=data, ax=ax3).set_title("Regression Plot of cnt vs casual")
111.
112. #make copy of dataset
113. df = data.copy()
114.
115. #Detect & Delete Outliers from the dataset
116.
117. cnames = ['casual','hum','windspeed']
118.
119. for i in cnames:
120.     q75, q25 = np.percentile(data.loc[:,i],[75,25])
```

```
    iqr = q75 - q25
    min = q25-(iqr*1.5)
    max = q75 +(iqr*1.5)
    print(min)
    print(max)
    data = data.drop(data[data.loc[:,i]< min].index)
    data = data.drop(data[data.loc[:,i]>max].index)

#check shape of dataset
data.shape #58 obseravtions are dropped in outliers

#generating correlation matrix
corr = data.corr()
corr

#plotting correlation matrix and heatmap using seaborn libraty
fig, ax = plt.subplots(figsize=(10,10))
sns.heatmap(corr,mask=np.zeros_like(corr, dtype=np.bool),cmap = sns.diverging_palette(220,10,as_cmap
    =True),square =True, annot=True, ax=ax)

#dropping atemp variable from a dataset
data = data.drop(['atemp'], axis = 1)

data.head(5)

#finding relationship between Numerical Variable 'temp', 'hum', 'windspeed' with target variable 'cnt'

fig, (ax1,ax2,ax3) = plt.subplots(ncols= 3)

fig.set_size_inches(15,9)

#relation between 'temp' and 'cnt'
sns.regplot(x="temp", y="cnt", data=data, ax=ax1).set_title("Regression Plot of cnt vs temp")

#relation between 'hum' and 'cnt'
sns.regplot(x="hum", y="cnt", data=data, ax=ax2).set_title("Regression Plot of cnt vs hum")

#relation between 'windspeed' and 'cnt'
sns.regplot(x="windspeed", y="cnt", data=data, ax=ax3).set_title("Regression Plot of cnt vs windspeed")

#now we find the relationship between categorical variables and Target Variable 'cnt'

```

```
# categorical variables are "Season", "holiday", "Weekday", "Workingday", "Weathersit", "month"
fig, axes = plt.subplots(nrows = 3, ncols=2)

fig.set_size_inches(12,18)

#plotting boxplot for cnt vs season variables
sns.boxplot(data =data, y="cnt", x="season", orient='v', ax=axes[0][0]).set_title("Boxplot of cnt vs season")

#plotting boxplot for cnt vs holiday variables
sns.boxplot(data =data, y="cnt", x="holiday", orient='v', ax=axes[0][1]).set_title("Boxplot of cnt vs holiday")

#plotting boxplot for cnt vs weekday variables
sns.boxplot(data =data, y="cnt", x="weekday", orient ='v', ax=axes[1][0]).set_title("Boxplot of cnt vs weekda
y")

#plotting boxplot for cnt vs workingday variables
sns.boxplot(data=data, y="cnt", x="workingday", orient='v', ax=axes[1][1]).set_title("Boxplot of cnt vs worki
ngday")

#plotting boxplot for cnt vs month variables
sns.boxplot(data=data, y="cnt", x="mnth", orient='v', ax=axes[2][0]).set_title("Boxplot of cnt vs mnth")

#plotting boxplot for cnt vs Weathershit variables
sns.boxplot(data=data, y="cnt", x="weathersit", orient='v', ax=axes[2][1]).set_title("Boxplot of cnt vs weath
ersit")

fig, axes = plt.subplots(nrows = 2, ncols = 3)

fig.set_size_inches(15,12)

#Check whether  variable 'temp'is normal or not
sns.distplot(data['temp'], ax=axes[0][0]).set_title("Normalization of temp")

#Check whether  variable 'hum'is normal or not
sns.distplot(data['hum'], ax=axes[0][1]).set_title("Normalization of hum")

#Check whether  variable 'windspeed'is normal or not
sns.distplot(data['windspeed'], ax=axes[0][2]).set_title("Normalization of windspeed")

#Check whether  variable 'casual'is normal or not
sns.distplot(data['casual'],ax=axes[1][0]).set_title("Normalization of casual")

```

```
#Check whether  variable 'registered'is normal or not
            sns.distplot(data['registered'], ax=axes[1][1]).set_title("Normalization of registered")

            #as well as Check whether  Target variable 'cnt'is normal or not
sns.distplot(data['cnt'], ax=axes[1][2]).set_title("Normalization of Target Variable cnt")

data.shape

            # before moving further
#let's droped the casual and registered variables because there sum is equal to target variable ie. 'cnt'

data = data.drop(['casual', 'registered'], axis =1)

#now let's define the feature matrix and response vector
X = data.drop('cnt', axis=1)
y = data.iloc[:,-1].values

#splitting X and y into training and testing dataset
#import sklearn train_test_split library
            from sklearn.model_selection import train_test_split

train_X, test_X, train_y, test_y =train_test_split(X,y,test_size=0.2)

#train the model using traing dataset
#import LinearRegression libraries from sklearn
from sklearn import linear_model
import statsmodels.api as sm

#train the model using training dataset
model_LR = sm.OLS(train_y, train_X.astype(float)).fit()

model_LR.summary()
#make the predictions by model
predict_LR = model_LR.predict(test_X)
def MAPE(true_y, pred_y):
    mape = np.mean(np.abs(true_y-pred_y)/true_y)
    return mape

#importing important error metrics libraries
            from sklearn.metrics import mean_absolute_error, mean_squared_error
# calculate MAE, MSE, MAPE, RMSE
print("MAE:",mean_absolute_error(test_y, predict_LR))
```

```
print("MSE:",mean_squared_error(test_y, predict_LR))
print("MAPE:",MAPE(test_y,predict_LR))
print("RMSE:",np.sqrt(mean_squared_error(test_y, predict_LR)))

#decision tree regression
#import DecisionTreeRegressor Analysis

from sklearn.tree import DecisionTreeRegressor
model_DT = DecisionTreeRegressor(max_depth = 2).fit(train_X,train_y)

#apply the model on test data
predict_DT = model_DT.predict(test_X)

# calculate MAE, MSE, MAPE, RMSE
print("MAE:",mean_absolute_error(test_y, predict_DT))
print("MSE:",mean_squared_error(test_y, predict_DT))
print("MAPE:",MAPE(test_y,predict_DT))
        print("RMSE:",np.sqrt(mean_squared_error(test_y, predict_DT)))

#Random forest analysis
#imnport RandomForestRegressor

from sklearn.ensemble import RandomForestRegressor

##create Random Forest object
model_RF = RandomForestRegressor(n_estimators = 50)

##train the model using training dataset
model_RF.fit(train_X, train_y)

#make the predictions by model
predict_RF = model_RF.predict(test_X)

# calculate MAE, MSE, MAPE, RMSE
print("MAE:",mean_absolute_error(test_y, predict_RF))
print("MSE:",mean_squared_error(test_y, predict_RF))
print("MAPE:",MAPE(test_y,predict_RF))
        print("RMSE:",np.sqrt(mean_squared_error(test_y, predict_RF)))

#KNN implementation
from sklearn.neighbors import KNeighborsRegressor
```

```
model_KNN = KNeighborsRegressor(n_neighbors =5).fit(train_X, train_y)
predict_KNN = model_KNN.predict(test_X)

            # calculate MAE, MSE, MAPE, RMSE
print("MAE:",mean_absolute_error(test_y, predict_KNN))
            print("MSE:",mean_squared_error(test_y, predict_KNN))
print("MAPE:",MAPE(test_y,predict_KNN))
print("RMSE:",np.sqrt(mean_squared_error(test_y, predict_KNN)))

#Random Forest Model has smallest error metrics i.e.

#   MAPE = 0.1290541

#So, for further analysis We are selecting Random Forest Model.
```