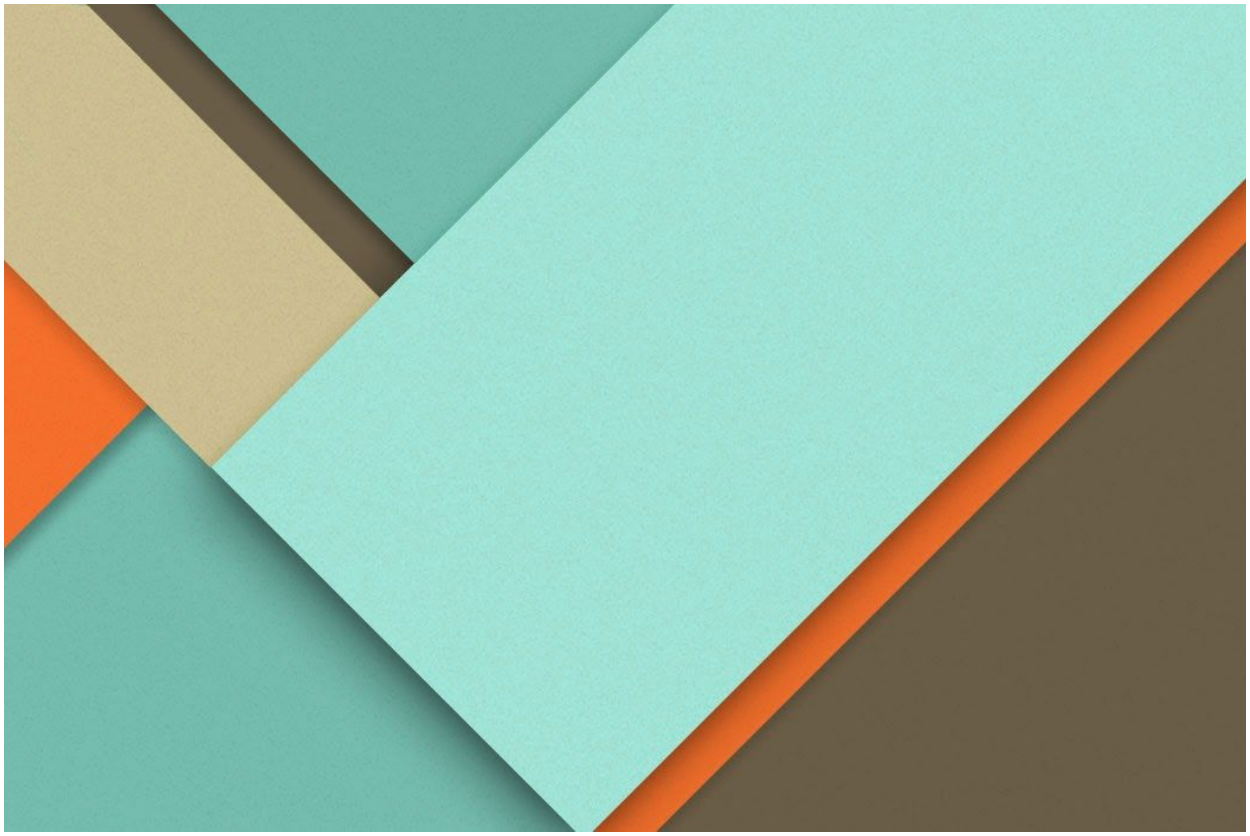


Bike Renting

22.08.2020

RAGHUNANDAN PAREEK

Contents

CHAPTER 1
INTRODUCTION

1.1 Problem Statement

Bike sharing systems are a new generation of traditional bike rentals where the whole process from membership, rental and return back has become automatic. Through these systems, users are able to easily rent a bike from a particular position and return back at another position. Apart from interesting real world applications of bike sharing systems, the characteristics of data being generated by these systems make them attractive for the research. Opposed to other transport services such as bus or subway, the duration of travel, departure and arrival position is explicitly recorded in these systems. This feature turns the bike sharing system into a virtual sensor network that can be used for sensing mobility in the city. Hence, it is expected that most of the important events in the city could be detected via monitoring these data.The objective of this Case is to predict daily bike rental count based on the environmental and seasonal settings.

1.2 Data

In our project, it is our task to build a model which can predict daily bike rental count based on the environmental and seasonal settings.

Table1.1 : Bike sharing Dataset(Columns:1-8)

Date	Season	Year	Month	Holiday	Weekday	Working Day	Weather Situation
2011-01-01	1	0	1	0	6	0	2
2011-01-02	1	0	1	0	0	0	2
2011-01-03	1	0	1	0	1	1	1
2011-01-04	1	0	1	0	2	1	1
2011-01-05	1	0	1	0	3	1	1

Table1.2 : Bike sharing Dataset(Columns:9-15)

Temperature	Feeling Temperature	Humidity	Windspeed	Casual	Registered	Total Count
0.344167	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
0.2	0.212122	0.590435	0.160296	108	1454	1562
0.226957	0.22927	0.436957	0.1869	82	1518	1600

From the table below we have the following 15 variables,using which we will predict the bike rental count:

Table1.3 : Predictor Variables

S.NO	Predictor
1	Date
2	Season
3	Year
4	Month
5	Holiday
6	Weekday
7	Working Day
8	Weather Situation

9	Temperature
10	Temperature Feeling
11	Humidity
12	Windspeed
13	Casual
14	Registered
15	Total Count

CHAPTER 2

METHODOLOGY

2.1 Exploratory Data Analysis

Exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics, often with visual methods. Exploratory data analysis is one of the most important steps in data mining in order to know features of data. It involves loading the dataset,data cleaning,normality test,typecasting of attributes, missing value analysis, outlier analysis, Attributes distributions and trends.So, we have to clean the data otherwise it will affect the performance of the model. Now we are going to explain one by one as follows.

2.1.1 Missing Value Analysis

Missing Values occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the datassing Values occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Codes to find missing values are as below

Python

```
Missing_Values = pd.DataFrame(df.isna().sum(),columns=['No Of Missing Values'])
Missing_Values
```

R

```
missing_val<-data.frame(apply(df,2,function(x){sum(is.na(x))}))
names(missing_val)[1]='missing_val'
Missing_val
```

Missing Values in Dataset

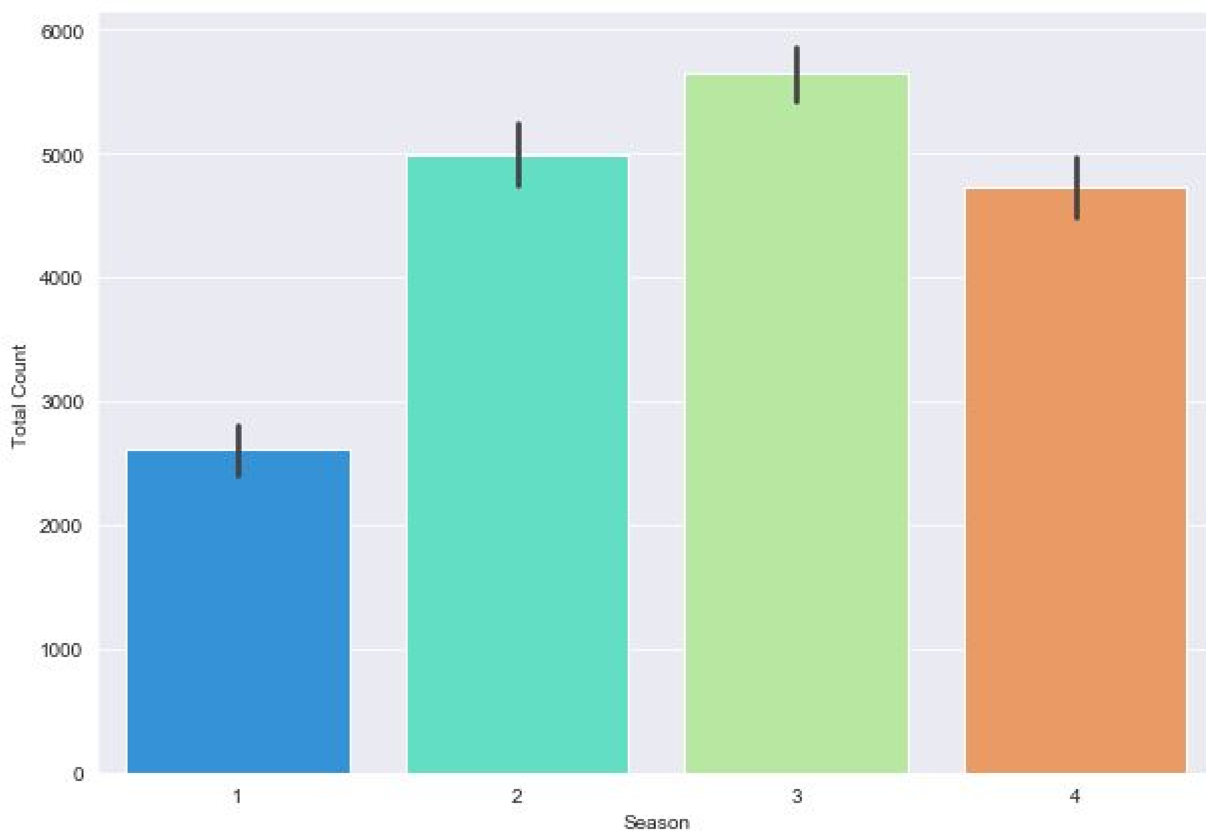
Columns	No Of Missing Values
Date	0
Season	0
Year	0
Month	0
Holiday	0
Weekday	0
Working Day	0
Weather Situation	0
Temperature	0
Temperature Feeling	0
Humidity	0
Windspeed	0
Casual	0
Registered	0
Total Count	0

2.1.2 Attributes Distributions and Trends

2.1.2.1 Season Wise Distribution of Total Count of Bike Rent

Here, Season 1 -> Spring, Season2 -> Summer,

Season 3 -> Fall , Season 4 -> Winter

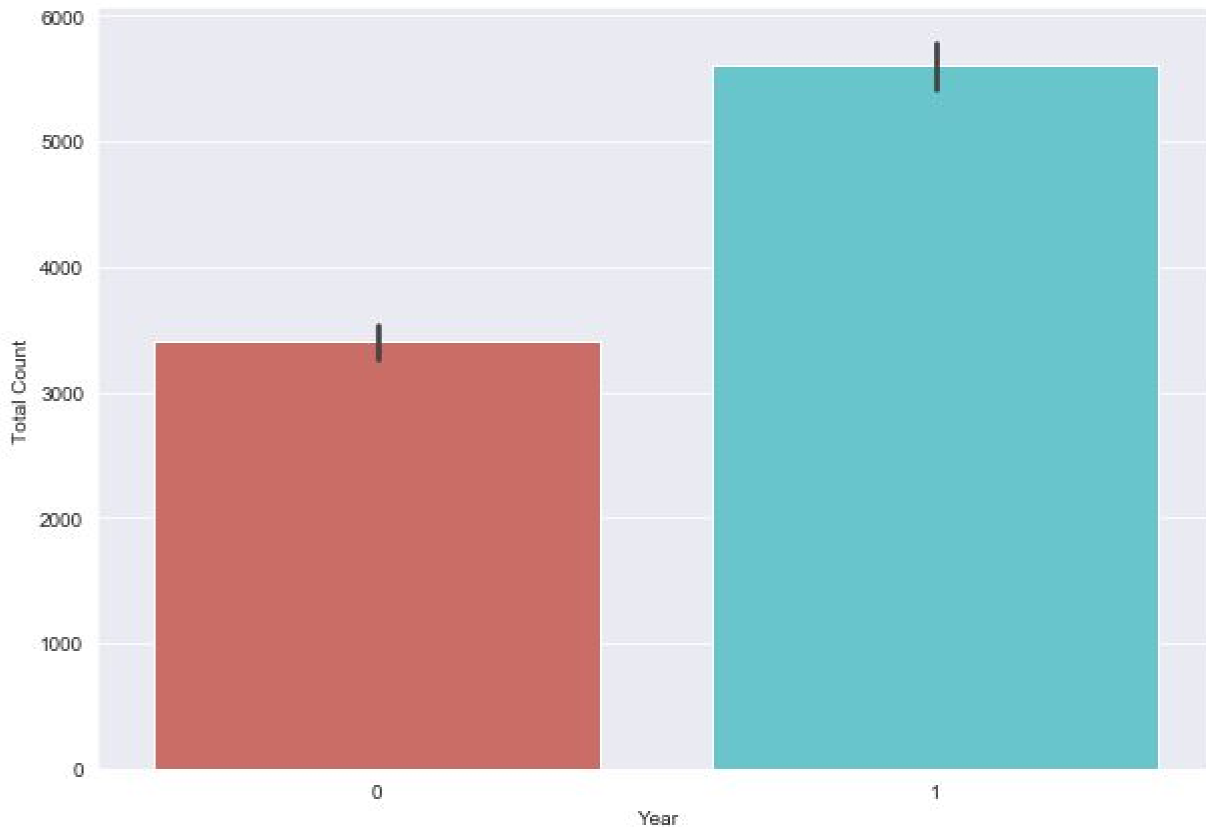


From the above plot it can be observed that bike rentals peak in Fall season and bottoms in Spring. After the Fall season it bike rental increases and peaks in Spring after Spring it starts decreasing in Winter and bottoms out in Winter

2.1.2.2 Year Wise Distribution of Total Count of Bike Rent

Here, Year 0 -> 2011,

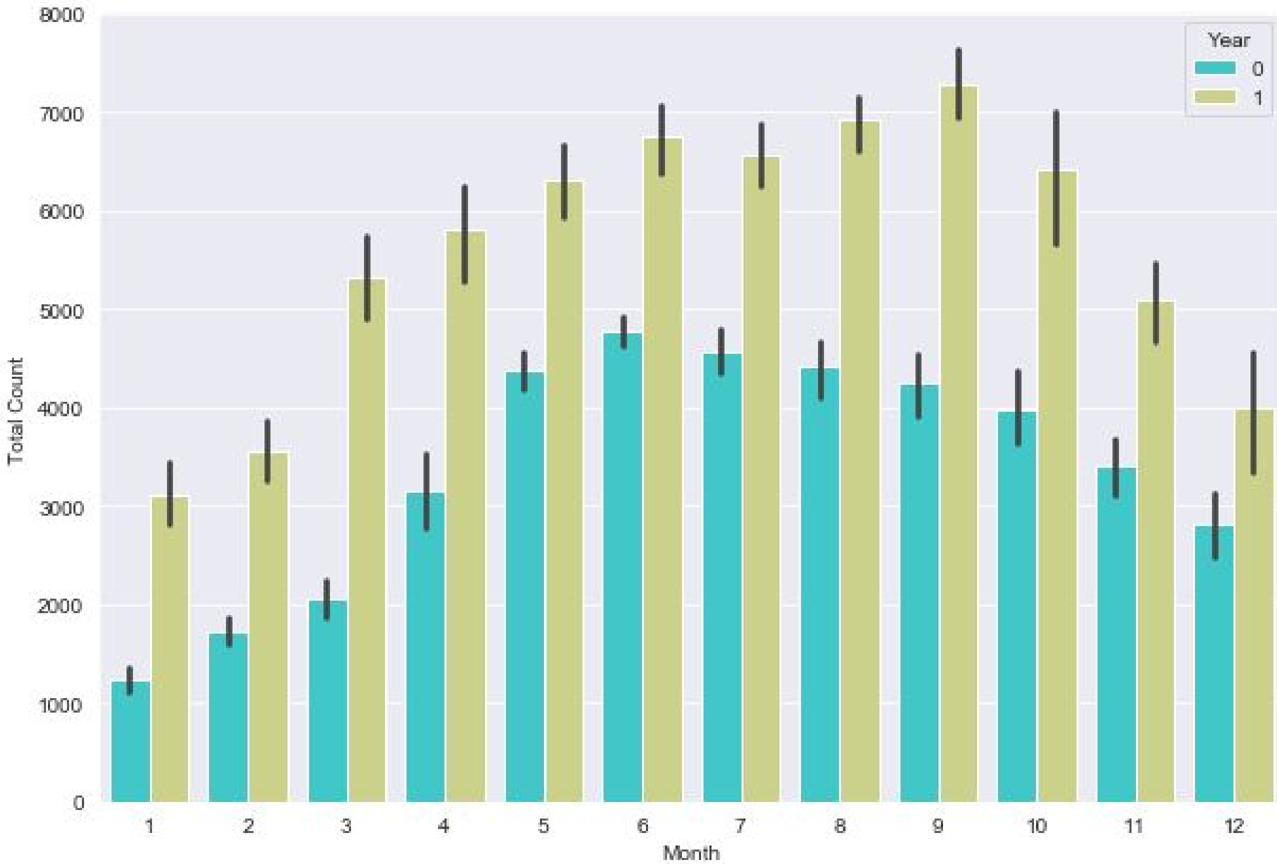
Year 1-> 2012



From the above plot it can be observed that bike rentals were lower in 2011 and it is increasing in 2012. Bike Rental is increasing year on year wise and an upward trend can be observed

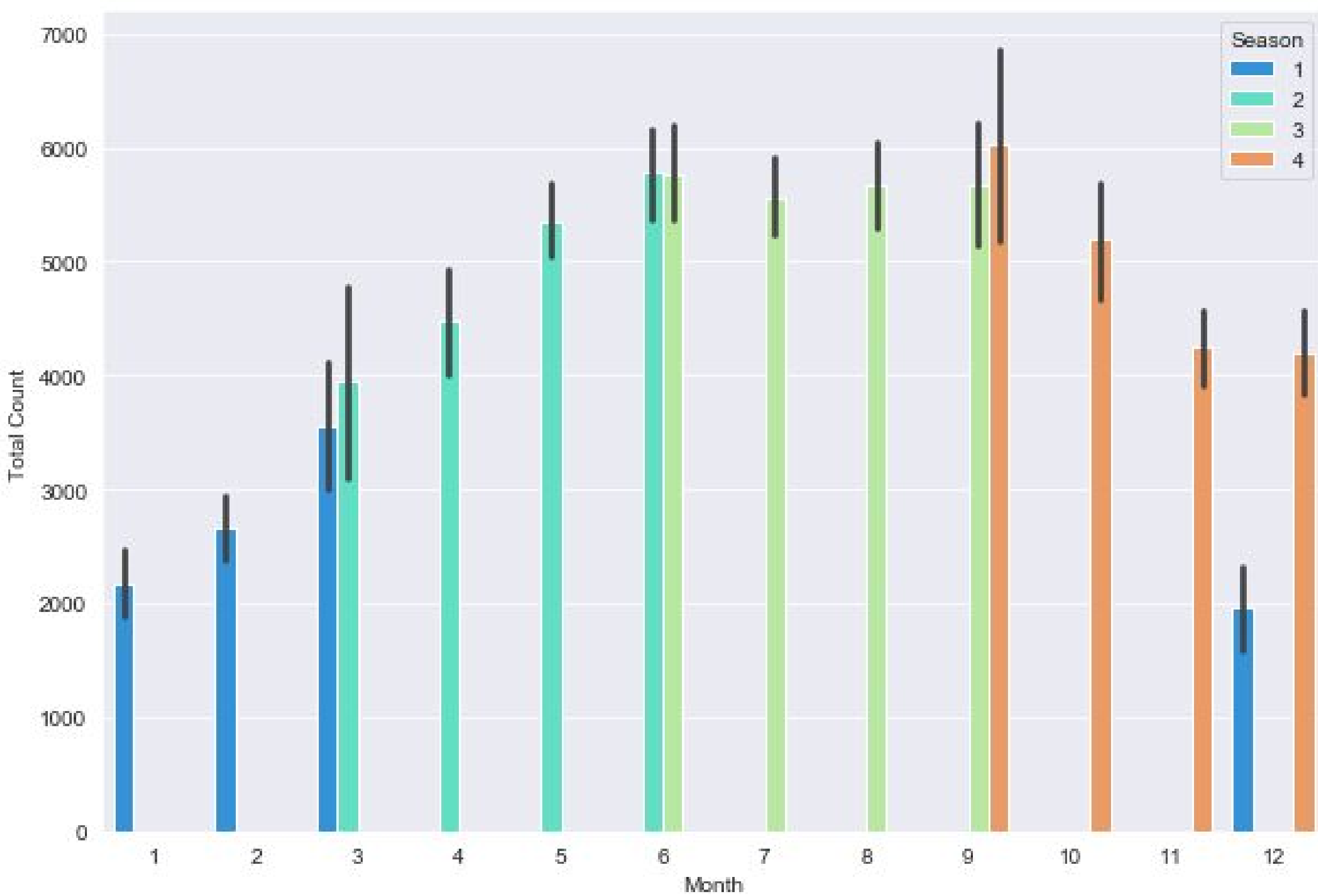
2.1.2.3 Month Wise Distribution of Total Count of Bike Rent

Here, 1 -> January, 2 -> February, 3 -> March, 4 -> April, 5 -> May, 6-> June, 7 -> July, 8 -> August, 9 -> September, 10 -> October, 11 -> November, 12 -> December



From the above plot it can be observed that bike rentals peak in May to September. After September it bike rental declines and bottoms out in January after January it starts increasing till September.

From the above plot it can be observed that bike rentals were lower in 2011 and it is increasing in 2012. Bike Rental is increasing month on month basis and an upward trend can be observed

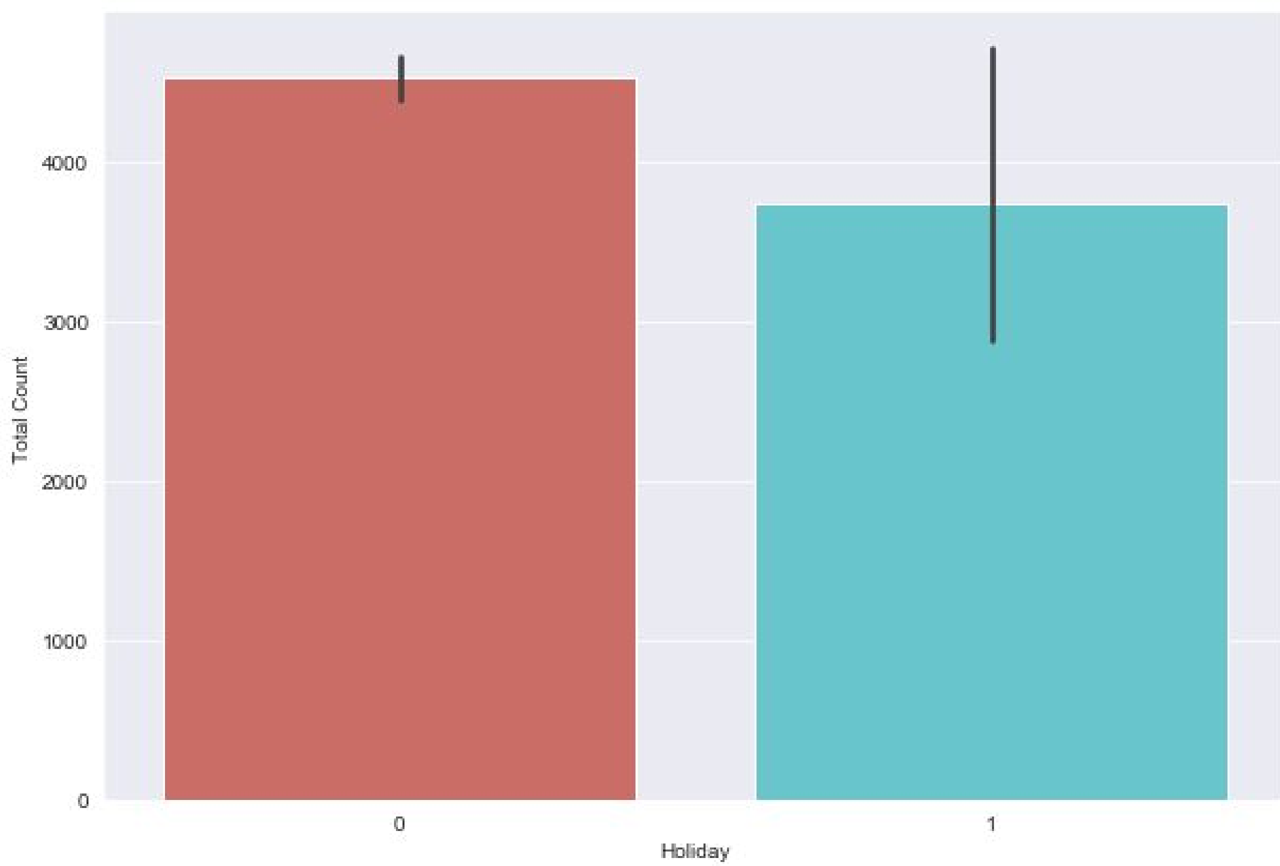


From the above plot it can be observed that bike rentals peak in the June and bottoms in January of Spring Season. After January bike rental increases till September and after September it starts decreasing till January.

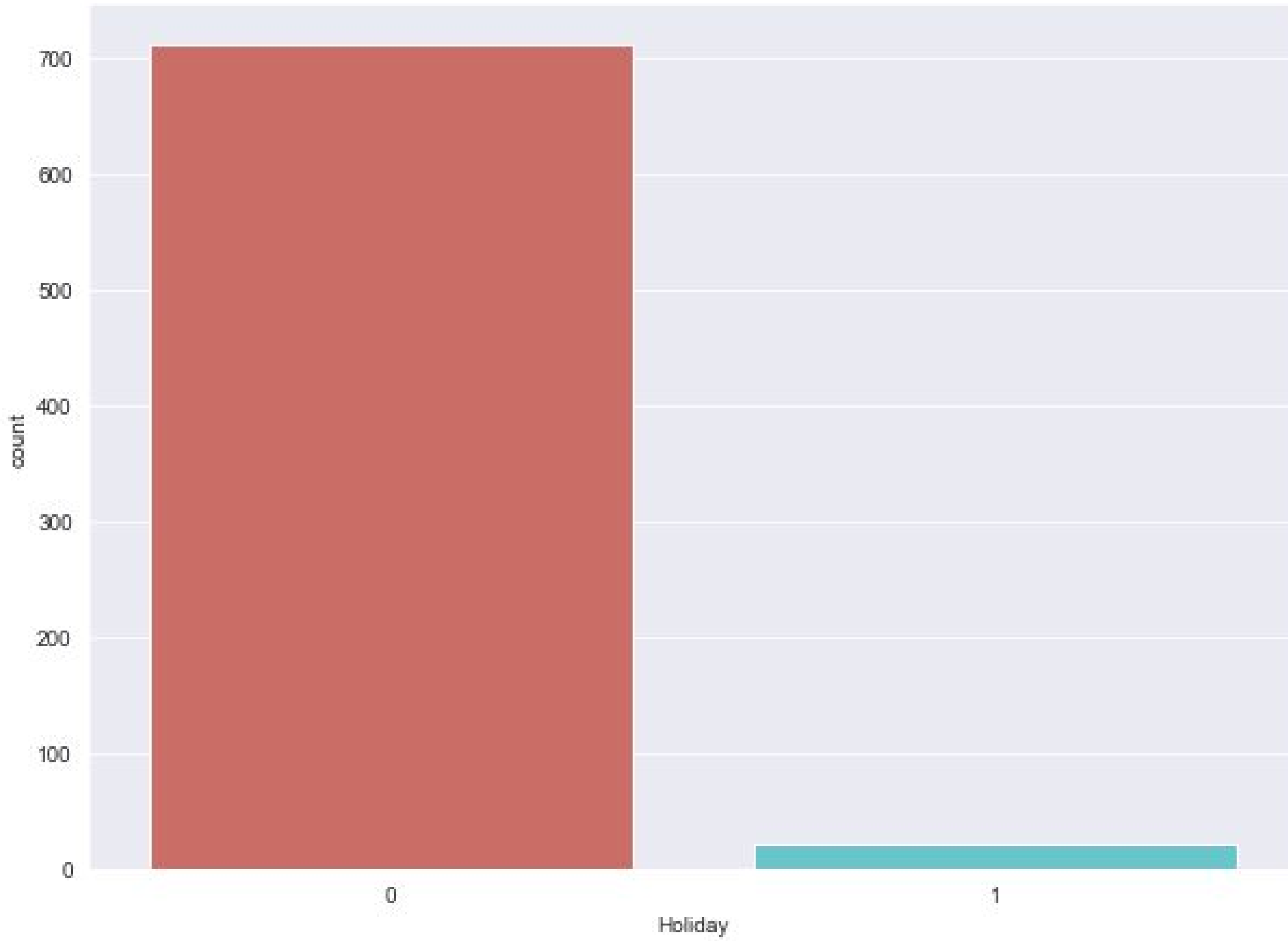
From the above plot it can be observed that bike rentals peak in Fall season and bottoms in Spring. After the Fall season it bike rental increases and peaks in Spring after Spring it starts decreasing in Winter and bottoms out in Winter.

2.1.2.4 Holiday Wise Distribution of Total Count of Bike Rent

Here, Holiday 0 -> No Holiday,   Holiday 1-> Holiday



Holiday vs Total Count



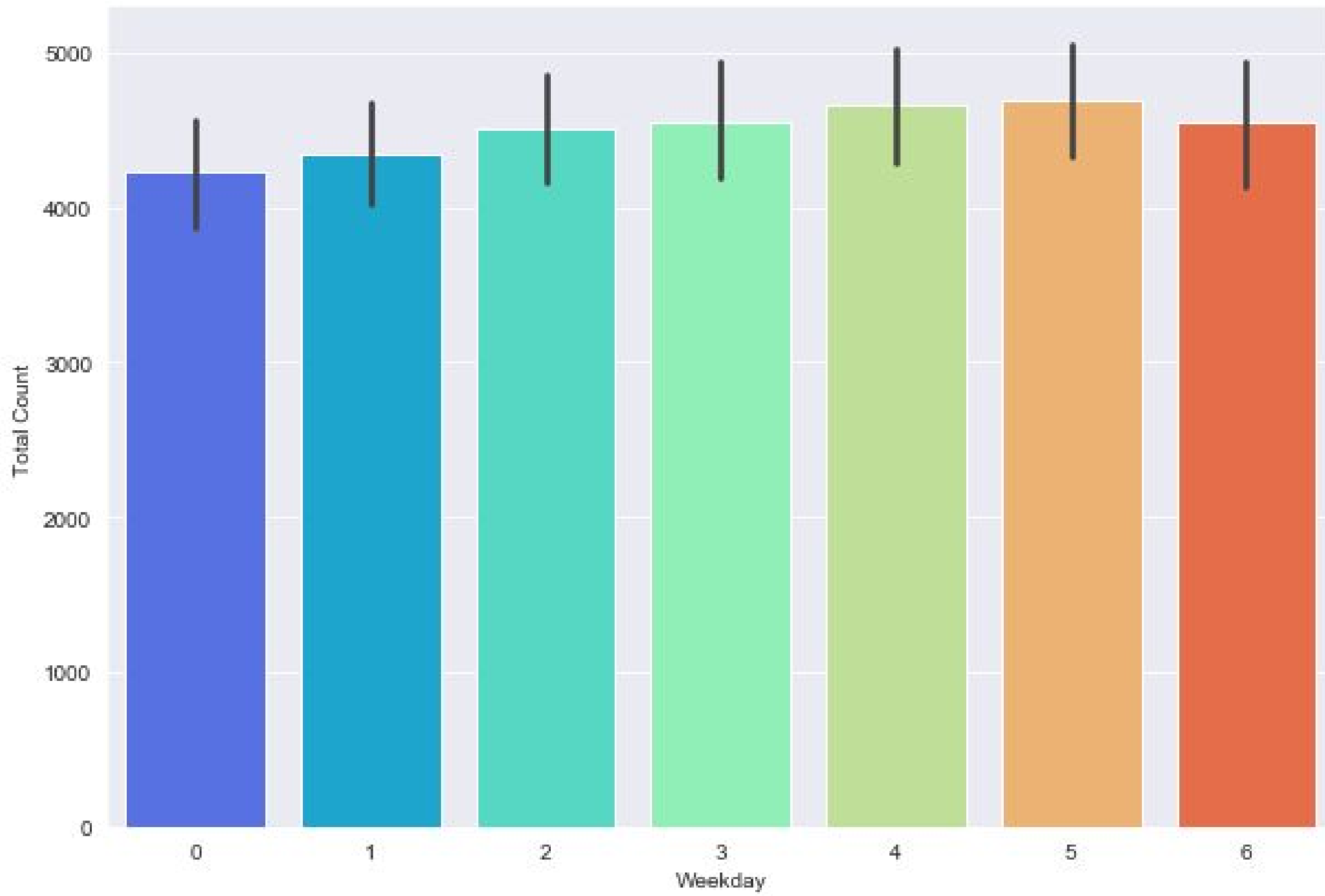
Frequency Plot Of Holiday

From the above bar plot, it can be can observed that during on non holiday the bike rental counts is higher compared to during holiday but frequency of holiday is lower than non holiday so bike rental is higher on holidays and comparatively lower on non holidays

2.1.2.5 Weekday Wise Distribution of Total Count of Bike Rent

Here, 0 -> Sunday ,1 -> Monday, 2 -> Tuesday, 3 -> Wednesday ,

4 -> Thursday, 5 -> Friday,6-> Saturday

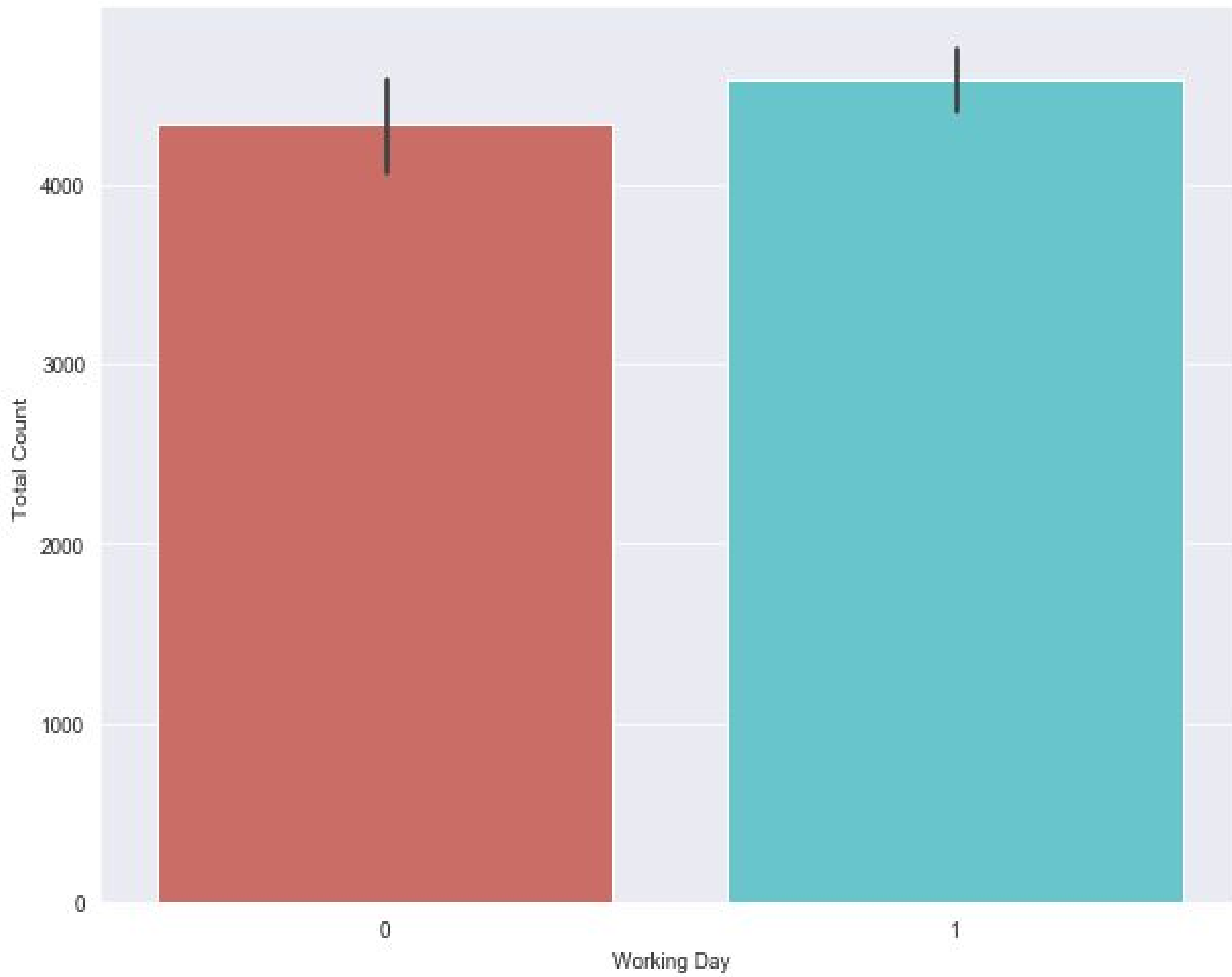


From the above plot it can be observed that on sundays and saturdays bike rental count is lower than other days. After Sunday Bike Rental Counts starts to increase and peaks from Thursday to Friday and starts declining from Saturday to Sunday.

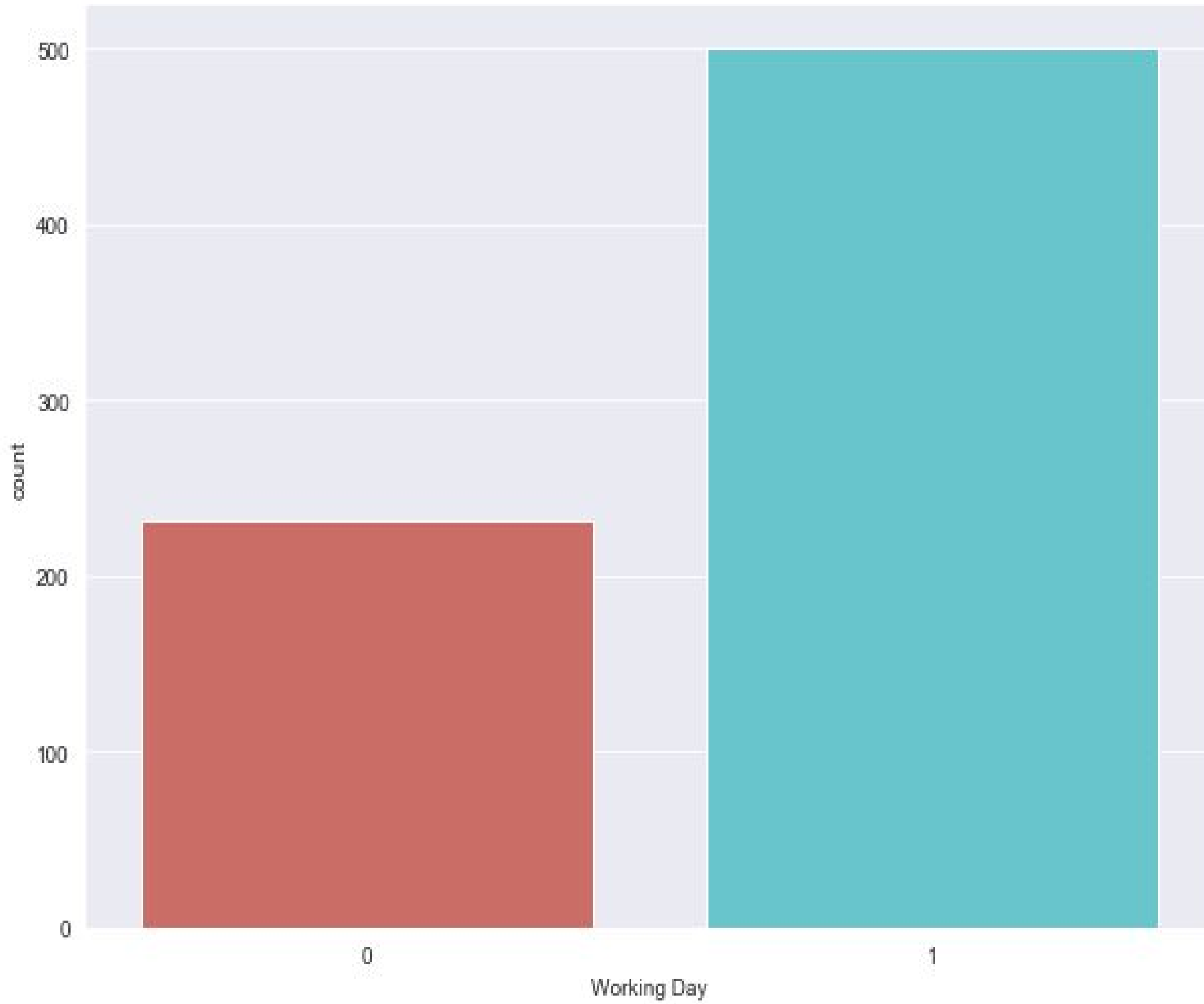
2.1.2.6 Working Day Wise Distribution of Total Count of Bike Rent

Here, Working Day 0 -> Weekend or Holiday

Working Day 1-> Working Day



Working Day vs Total Count



Frequency Plot of Working Day

From the above bar plot, it can be can observed that during on Working Day the bike rental counts is higher compared to during Non but frequency of Non Working Day is lower than Working Day so bike rental is higher on Non Working Day and comparatively lower on Working Day

2.1.2.7 Weather Situation Wise Distribution of Total Count of Bike Rent

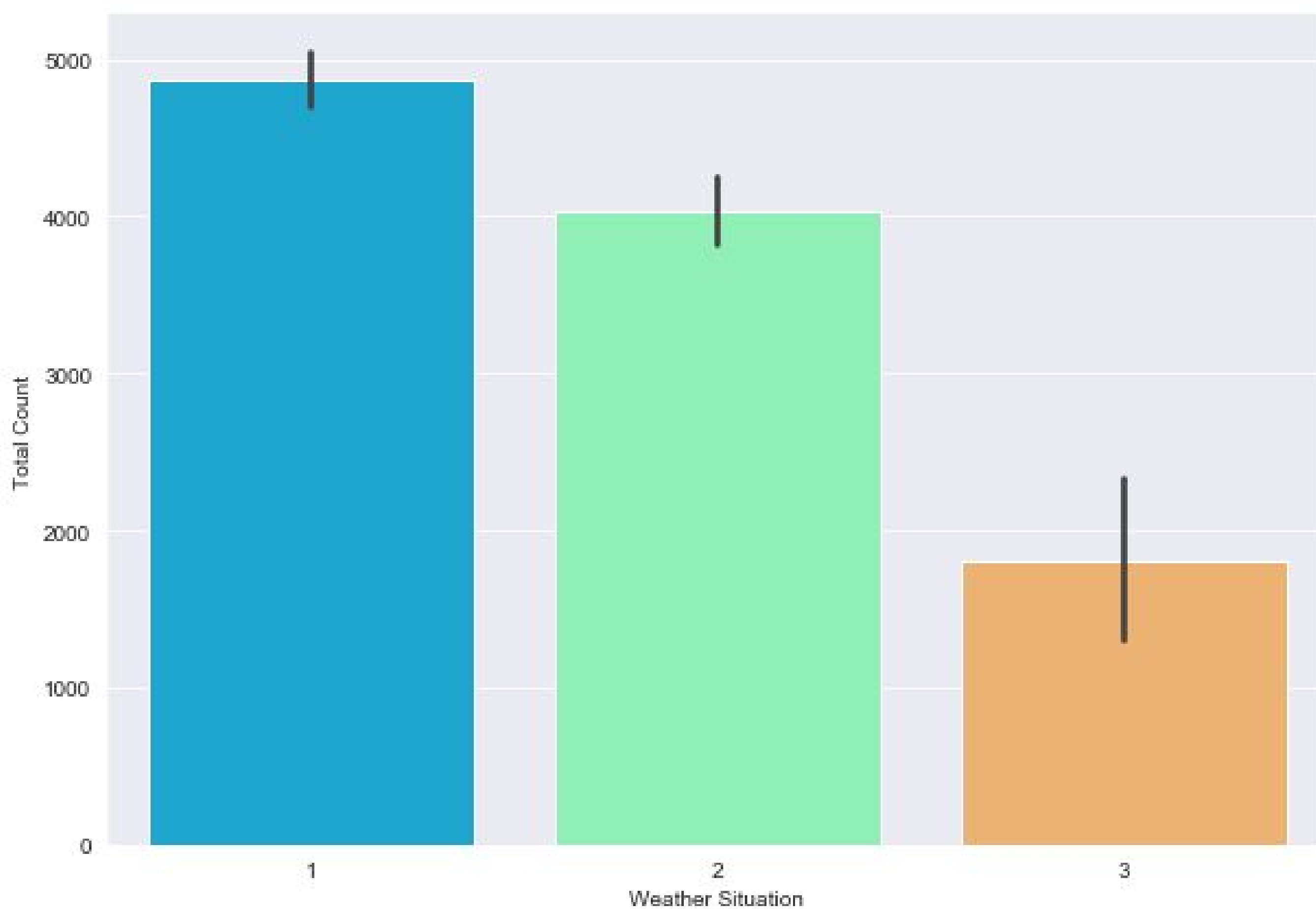
Here,

Weather Situation 0 -> Clear or Few clouds or Partly cloudy

Weather Situation 1-> Mist or Cloudy, Mist with Broken clouds

Weather Situation 2 -> Light Snow Or Light Rain With Thunderstorm and Scattered clouds

Weather Situation 4 -> Heavy Rain with Ice Pallets,Thunderstorm and Mist, Snow or Fog

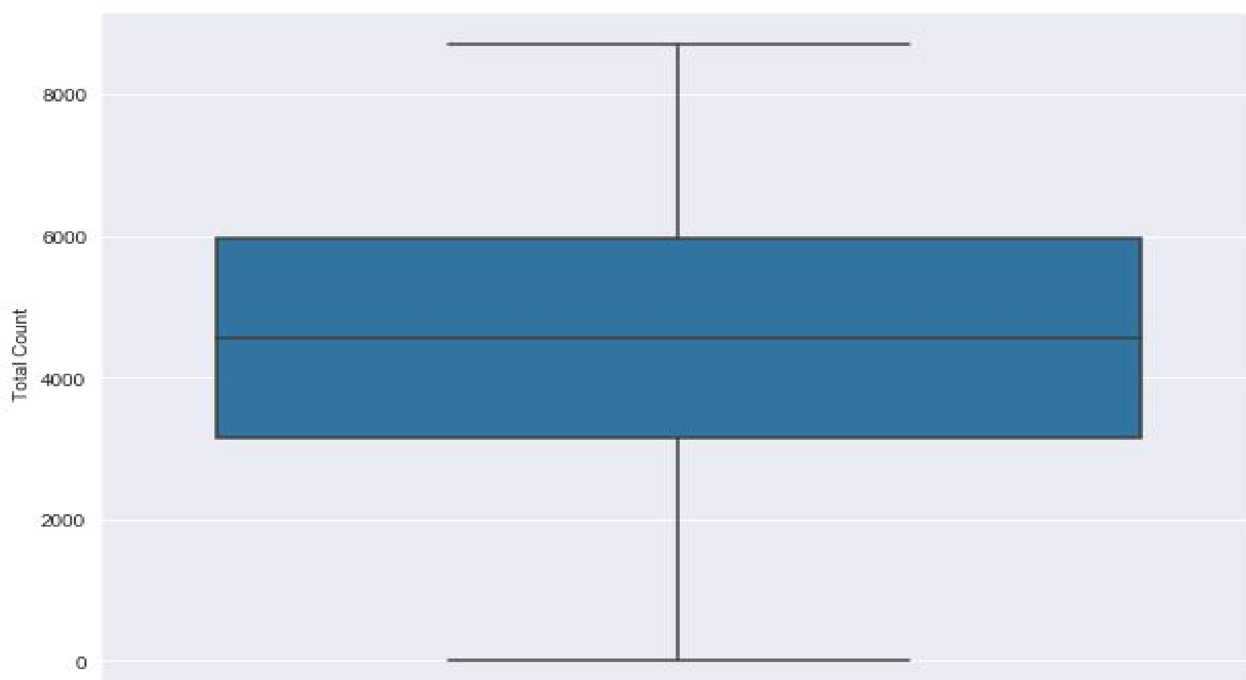


From the above bar plot, it can be observed that during Clear Weather Situations Bike Rental Count is highest and lowest on Heavy Rain or Snow. On Cloudy Weather Situations Bike Rental Count is higher than Little Rain or Little Snow

2.1.3 Outlier Analysis

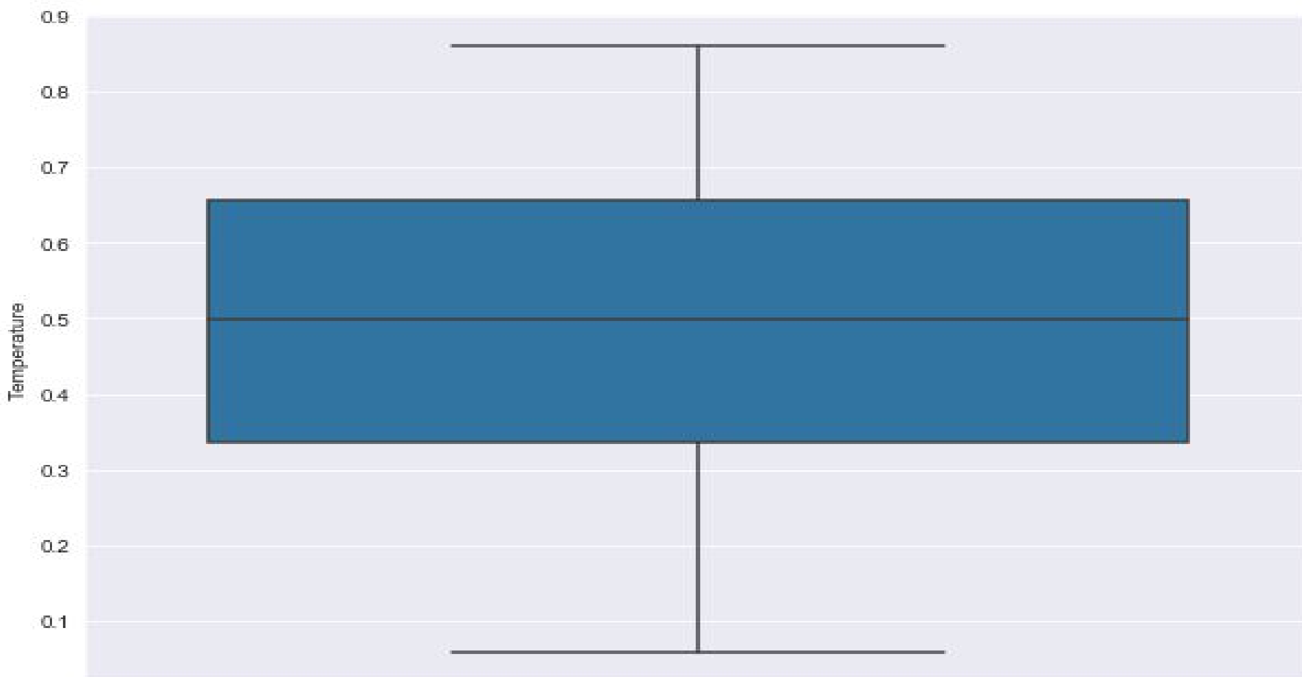
Outliers are the object that deviates significantly from the rest of the objects. They can be caused by measurement or execution error. The analysis of outlier data is referred to as outlier analysis or outlier mining.

2.1.3.1 Outlier Analysis of Total Counts



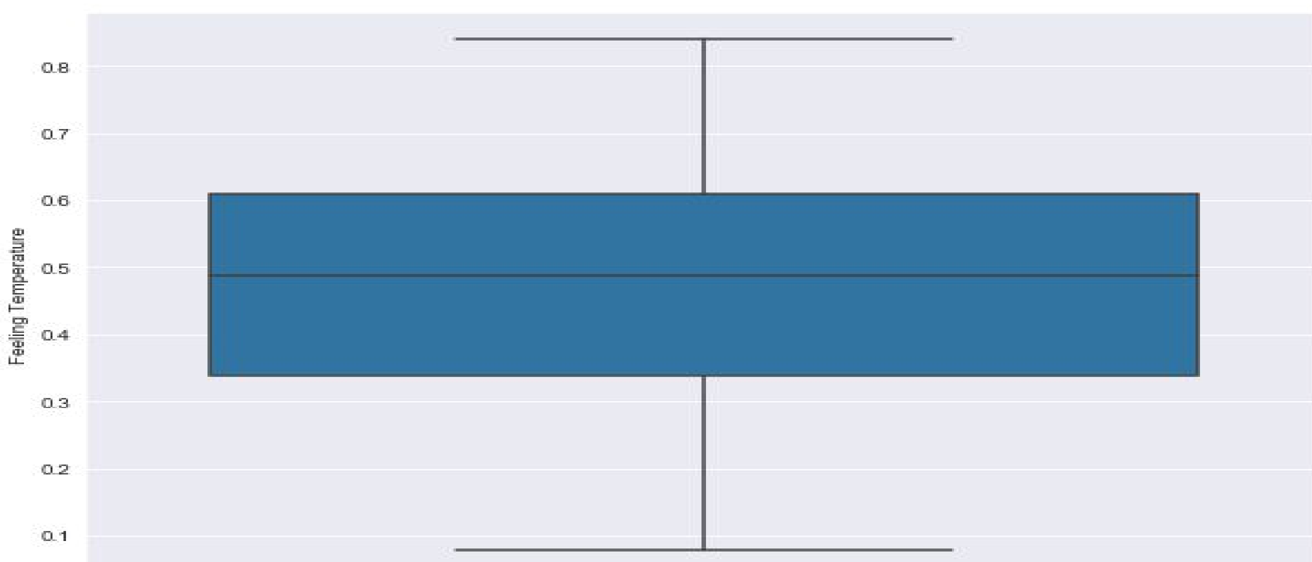
There are no outliers present in Total Counts

2.1.3.2 Outlier Analysis of Temperature



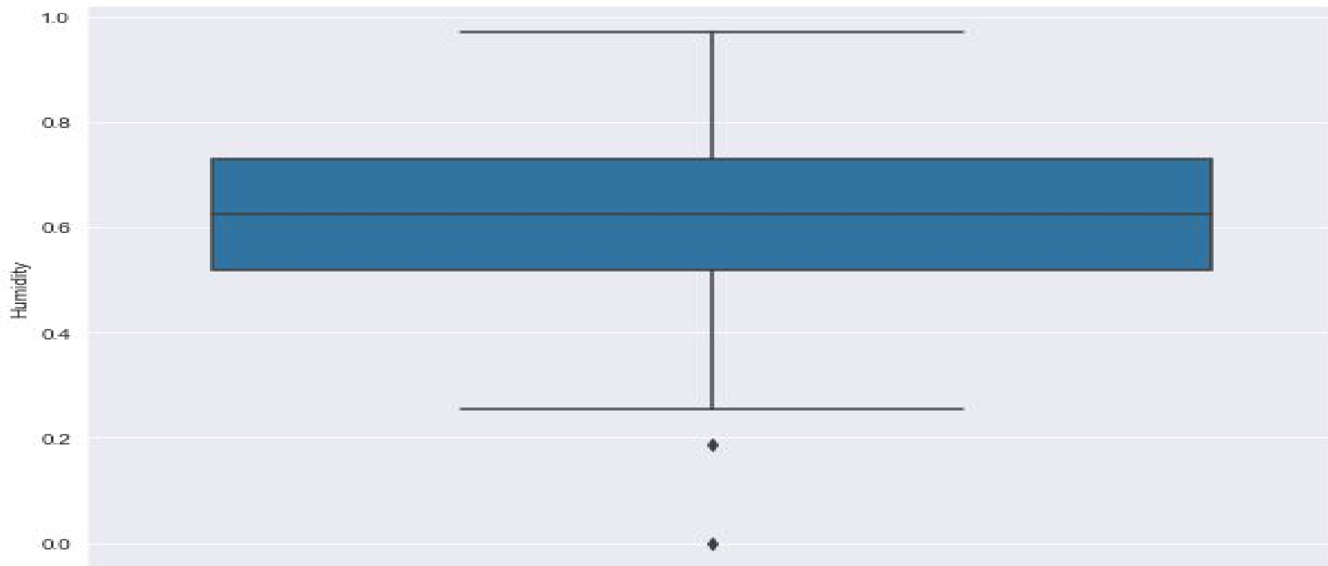
There are no outliers present in Temperature

2.1.3.3 Outlier Analysis of Feeling Temperature



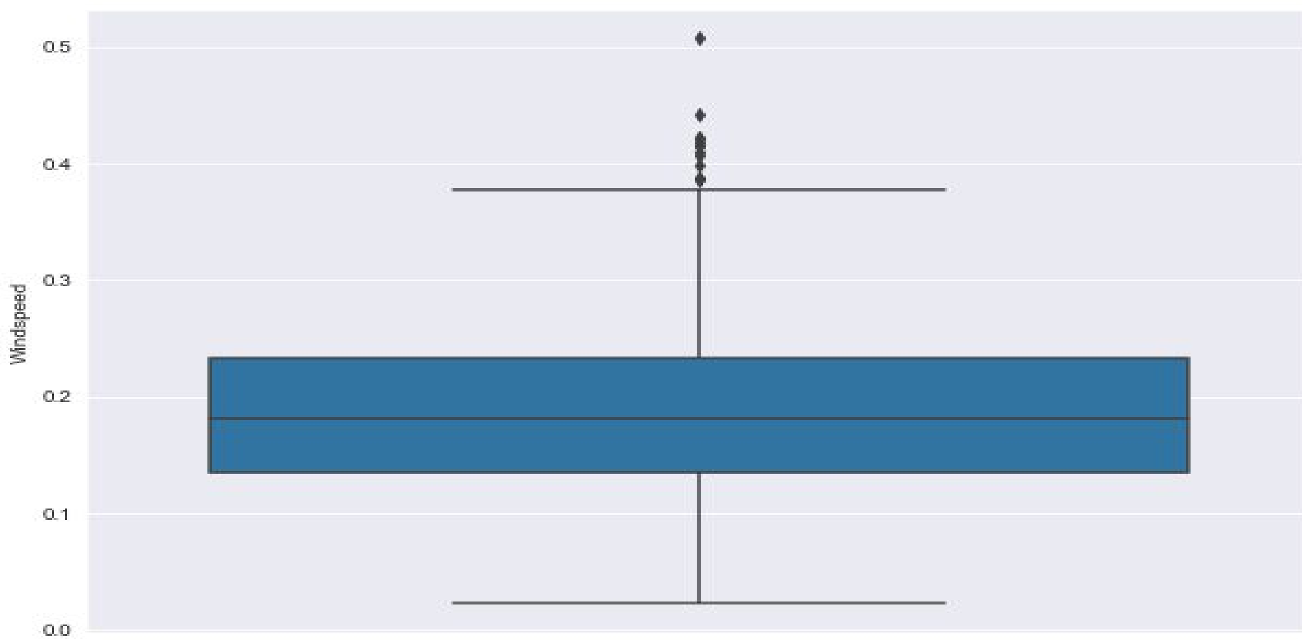
There are no outliers present in Feeling Temperature

2.1.3.4 Outlier Analysis of Humidity



There are outliers present in Humidity

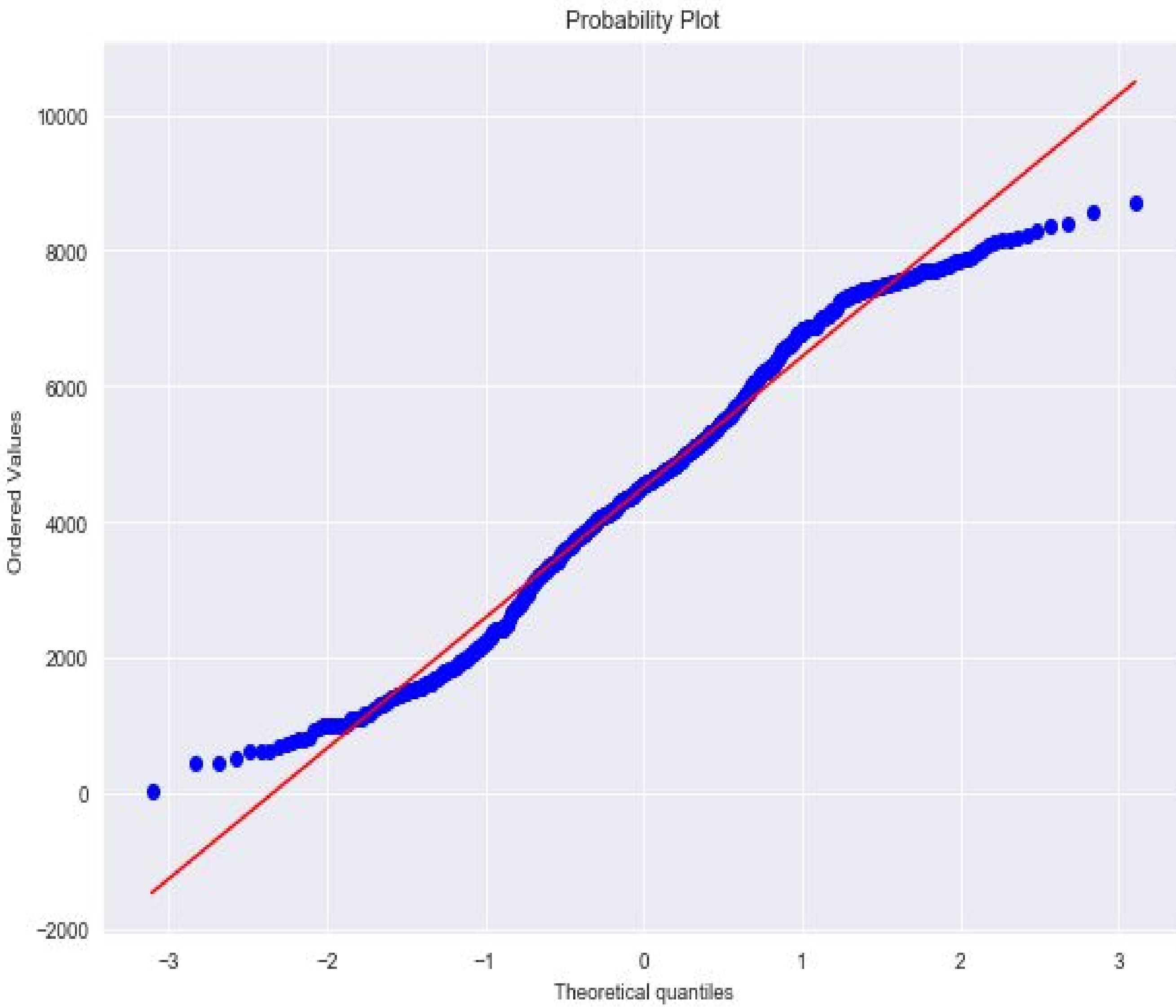
2.1.3.5 Outlier Analysis of Windspeed



There are outliers present in Windspeed

2.1.3.6 Normality Test

Normality Tests are used to determine if a data set is well-modeled by a normal distribution and to compute how likely it is for a random variable underlying the data set to be normally distributed.

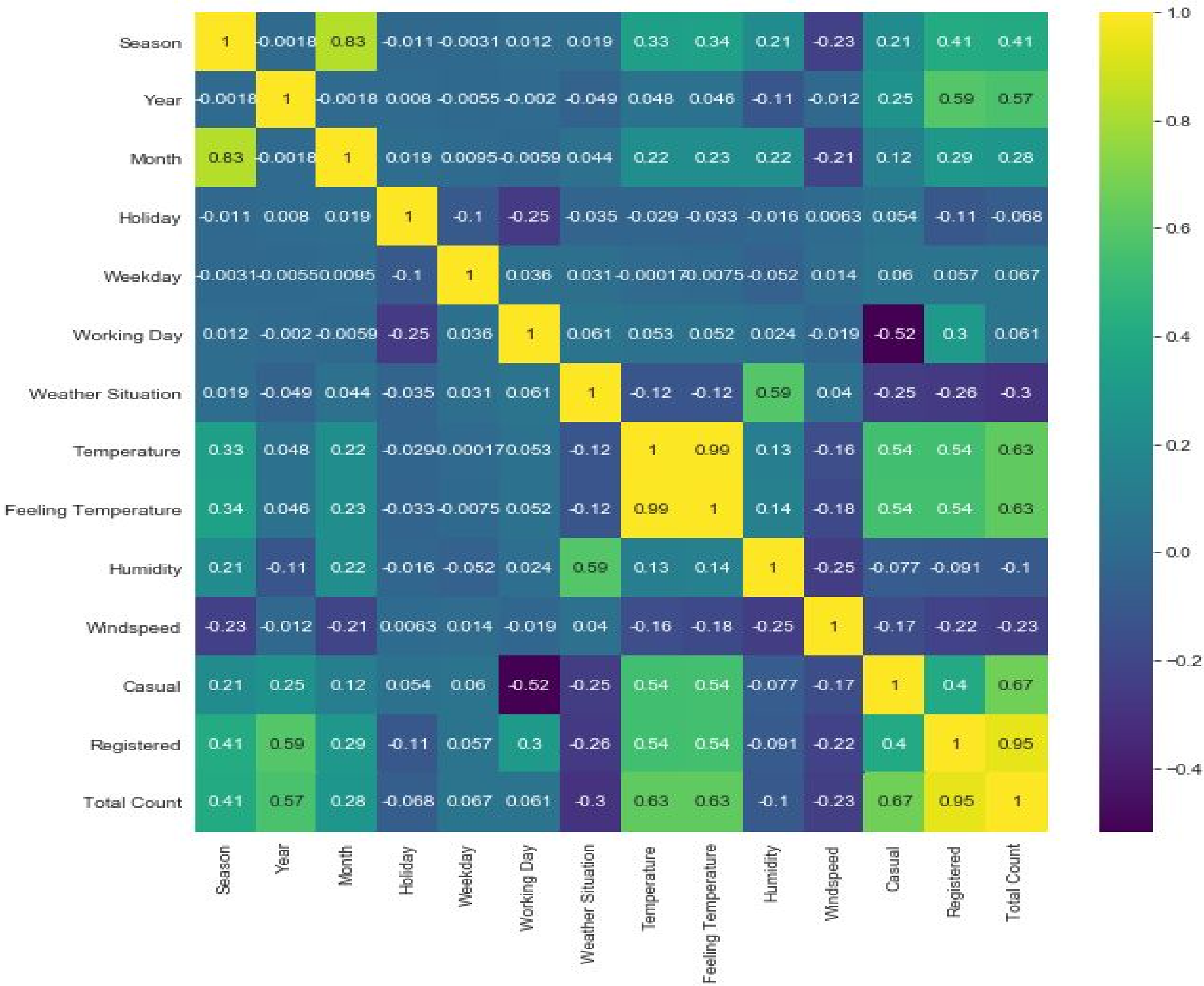


Normality Test Graph

2.1.4 Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model.It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model.

There are several methods of feature selection.We have used Correlation matrix for feature selection.



2.1.4.1 Correlation Matrix

Correlation is any statistical relationship, whether causal or not, between two random variables or bivariate data. In the broadest sense correlation is any statistical association, though it commonly refers to the degree to which a pair of variables are linearly related.

Column	Correlation Score
Season	0.406100
Year	0.566710
Month	0.279977
Holiday	-0.068348
Weekday	0.067443
Working Day	0.061156
Weather Situation	-0.297391
Temperature	0.627494
Feeling Temperature	0.631066
Humidity	-0.100659
Windspeed	-0.234545
Total Count	1.000000

2.2 Modeling

2.2.1 Model Selection

Model selection is the process of choosing one among many candidate models for a predictive modeling problem.

The dependent variable can fall in either of the four categories:

1. Nominal

2. Ordinal

3. Interval

4. Ratio

If the dependent variable is Nominal the only predictive analysis that we can perform is Classification, and if the dependent variable is Interval or Ratio like this project, the normal method is to do a Regression analysis, or classification after binning. But the dependent variable we are dealing with is not Ordinal, for which regression can be done. You always start your model building from the most simplest to more complex.

2.2.2 Multiple Linear Regression

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

```
                            OLS Regression Results
=======================================================================================
Dep. Variable:            Total Count   R-squared (uncentered):                   0.966
Model:                            OLS   Adj. R-squared (uncentered):              0.966
Method:                 Least Squares   F-statistic:                              2253.
Date:                Fri, 21 Aug 2020   Prob (F-statistic):                        0.00
Time:                        15:38:48   Log-Likelihood:                         -5898.7
No. Observations:                 717   AIC:                                  1.182e+04
Df Residuals:                     708   BIC:                                  1.186e+04
Df Model:                           9
Covariance Type:            nonrobust
=====================================================================================
                        coef    std err          t      P>|t|      [0.025      0.975]
-------------------------------------------------------------------------------------
Season              553.6691     57.601      9.612      0.000     440.579     666.759
Year               2155.3004     67.028     32.155      0.000    2023.702    2286.899
Month               -37.3258     18.170     -2.054      0.040     -73.000      -1.652
Holiday            -538.8014    202.941     -2.655      0.008    -937.240    -140.362
Weekday              99.7317     16.618      6.001      0.000      67.105     132.358
Weather Situation  -664.6676     84.973     -7.822      0.000    -831.498    -497.838
Temperature        5534.5446    197.894     27.967      0.000    5146.015    5923.074
Humidity            369.4625    276.250      1.337      0.182    -172.905     911.831
Windspeed          -355.3633    407.275     -0.873      0.383   -1154.975     444.248
==============================================================================
Omnibus:                      116.660   Durbin-Watson:                   1.045
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              250.681
Skew:                          -0.905   Prob(JB):                     3.68e-55
Kurtosis:                       5.262   Cond. No.                         103.
==============================================================================
```

As you can see the Adjusted R-squared value, we can explain only about 96% of the data using our multiple linear regression model. This is very good, but at least looking at the F-statistic and combined p-value we can reject the null hypothesis that the target variable does not depend on any of the predictor variables.

```
Call:
lm(formula = Total_Count ~ ., data = training_set)

Residuals:
    Min      1Q  Median      3Q     Max 
-4044.4  -475.3    48.2   517.4  2844.9 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)        1758.15     254.57   6.906 1.34e-11 ***
Season              507.19      61.62   8.231 1.29e-15 ***
Year               2053.29      73.77  27.835  < 2e-16 ***
Month               -41.57      19.31  -2.153 0.031769 *  
Holiday            -410.63     210.62  -1.950 0.051710 .  
Weekday              62.86      18.17   3.460 0.000582 ***
Weather_Situation  -604.16      86.79  -6.962 9.39e-12 ***
Temperature        5123.42     218.97  23.398  < 2e-16 ***
Humidity           -889.23     345.73  -2.572 0.010366 *  
Windspeed         -3015.81     534.72  -5.640 2.69e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 873.5 on 564 degrees of freedom
Multiple R-squared:  0.798,	Adjusted R-squared:  0.7948 
F-statistic: 247.6 on 9 and 564 DF,  p-value: < 2.2e-16
```

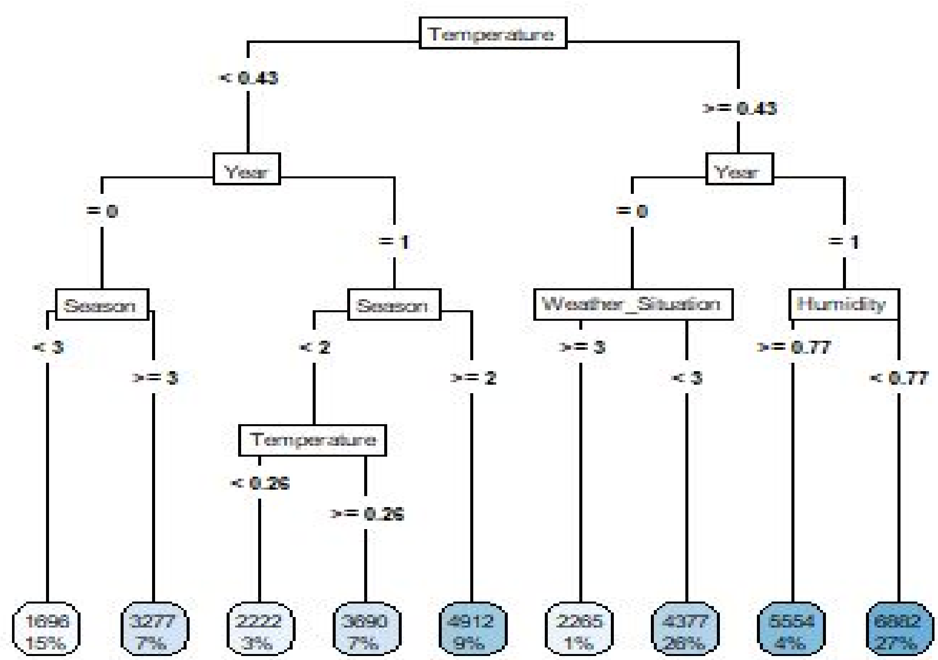
As Adjusted R-squared value, we can explain only about 80% of the data using our multiple tree regression model.

It means that the predictor is able to predict 80% of the variance in the target variable which is contributed by all the independent variables.

2.2.3 Decision Tree Regression

Decision trees are constructed via an algorithmic approach that identifies ways to split a data set based on different conditions. It is one of the most widely used and practical methods for supervised learning. Decision Trees are a non-parametric supervised learning method used for both classification and regression tasks.

Tree models where the target variable can take a discrete set of values are called classification trees. Decision trees where the target variable can take continuous values (typically real numbers) are called regression trees.

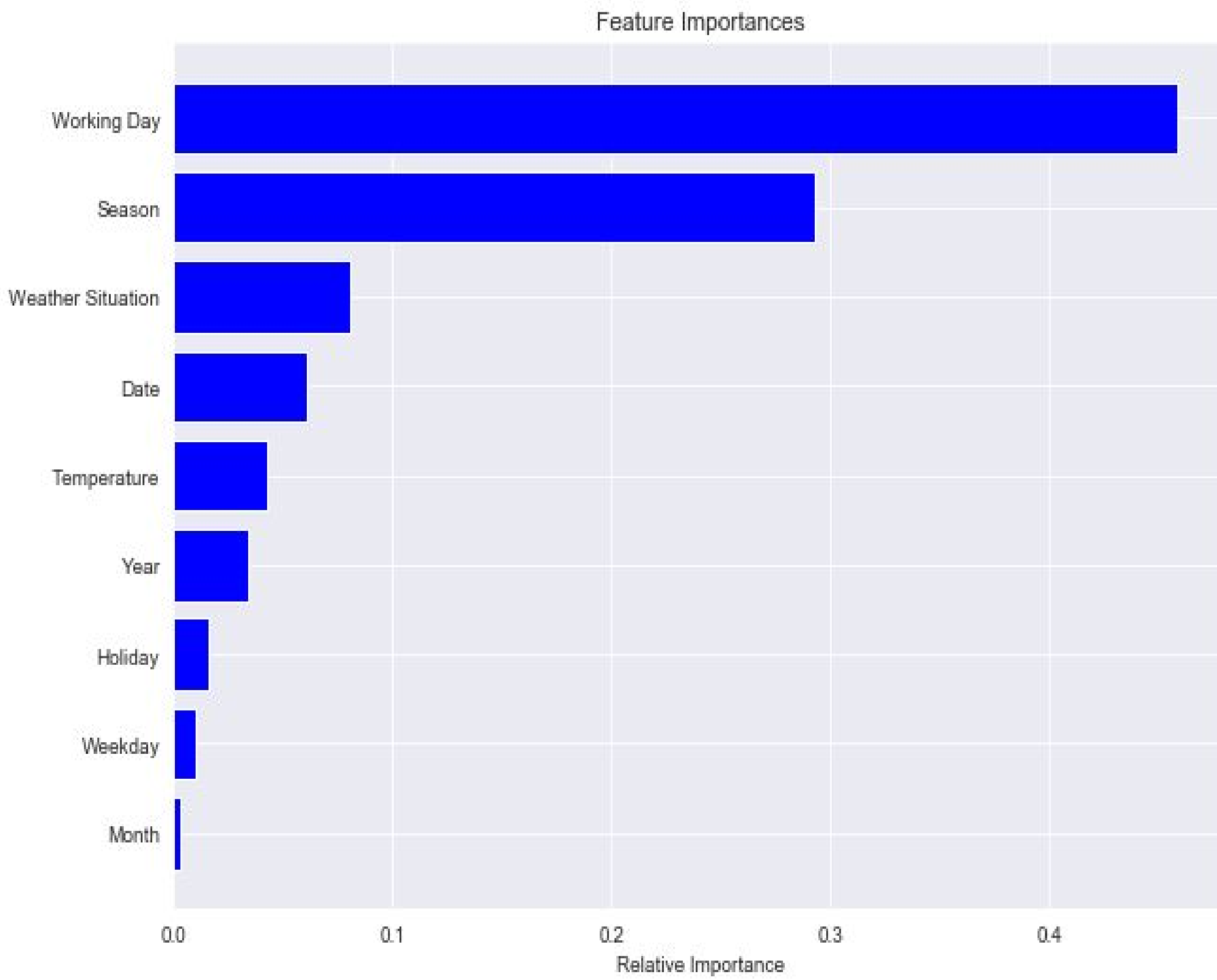


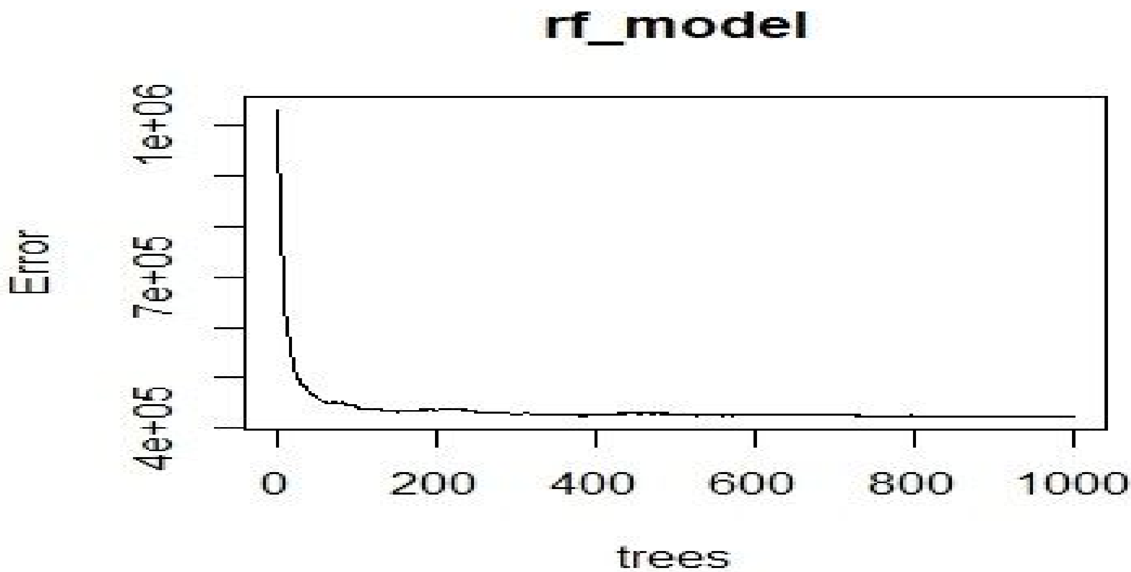
As an Adjusted R-squared value, we can explain only about 86% of the data using our decision tree regression model.

It means that the predictor is able to predict 86% of the variance in the target variable which is contributed by all the independent variables.

2.2.4 Random Forest Regression

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees





As an Adjusted R-squared value, we can explain only about 91.5% of the data using our random forest regression model.

It means that the predictor is able to predict 91.5% of the variance in the target variable which is contributed by all the independent variables.

CHAPTER 3

CONCLUSION

3.1 MODEL EVALUATION

Model Evaluation is an integral part of the model development process. It helps to find the best model that represents our data and how well the chosen model will work in the future.

Evaluating model performance with the data used for training is not acceptable in data science because it can easily generate overoptimistic and overfitted models.

There are two methods of evaluating models in data science, Hold-Out and Cross-Validation. To avoid overfitting, both methods use a test set (not seen by the model) to evaluate model performance.

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

3.1.1 Mean Absolute Error (MAE)

Mean Absolute Error is a measure of errors between paired observations expressing the same phenomenon.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - y_j|$$

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
Mean Absolute Error In Python	598	536	432

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
Mean Absolute Error In R	613	714	429

3.1.2 Root Mean Square Errors (RMSE)

Root Mean Square Error (RMSE) is the standard deviation of the residuals (prediction errors). RMSE is a measure of how spread out these residuals are.

$$RMSErrors = \sqrt{\frac{\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}{n}}$$

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
Root Mean Absolute Error In Python	822	793	614

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
Root Mean Absolute Error In R	933	960	653

3.1.3 Coefficient OF Determination

It explains the how much variance of dependent variable which is contributed by all the independent variables.

R-Squared Score

$$r^2 = 1 - \frac{\Sigma\,(y - y')^2}{\Sigma\,(y - \overline{y}')^2}$$

Adjusted R-Squared Score

Adj_r2 = 1-(1-R2)*(n-1)/(n-p-1)

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
R-Squared Score in Python	0.8495	0.8597	0.9160
Adjusted R-Squared Score in Python	0.8476	0.8580	0.9150

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
R-Squared Score in R	0.8105	0.7515	0.9114
Adjusted R-Squared Score in R	0.8075	0.7410	0.9040

3.1.4 Cross Validation

Cross-validation, sometimes called rotation estimation or out-of-sample testing, is any of various similar model validation techniques for assessing how the results of a statistical analysis will generalize to an independent data set

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

Model	Multiple Linear Regression	Decision Tree Regression	Random Forest Regression
Cross Validation Score	83.99%	73.94%	81.17%
Standard Deviation	2.96%	8.05%	4.36%

3.2 MODEL SELECTION

Model selection is the process of selecting one final machine learning model from among a collection of candidate machine learning models for a training dataset.

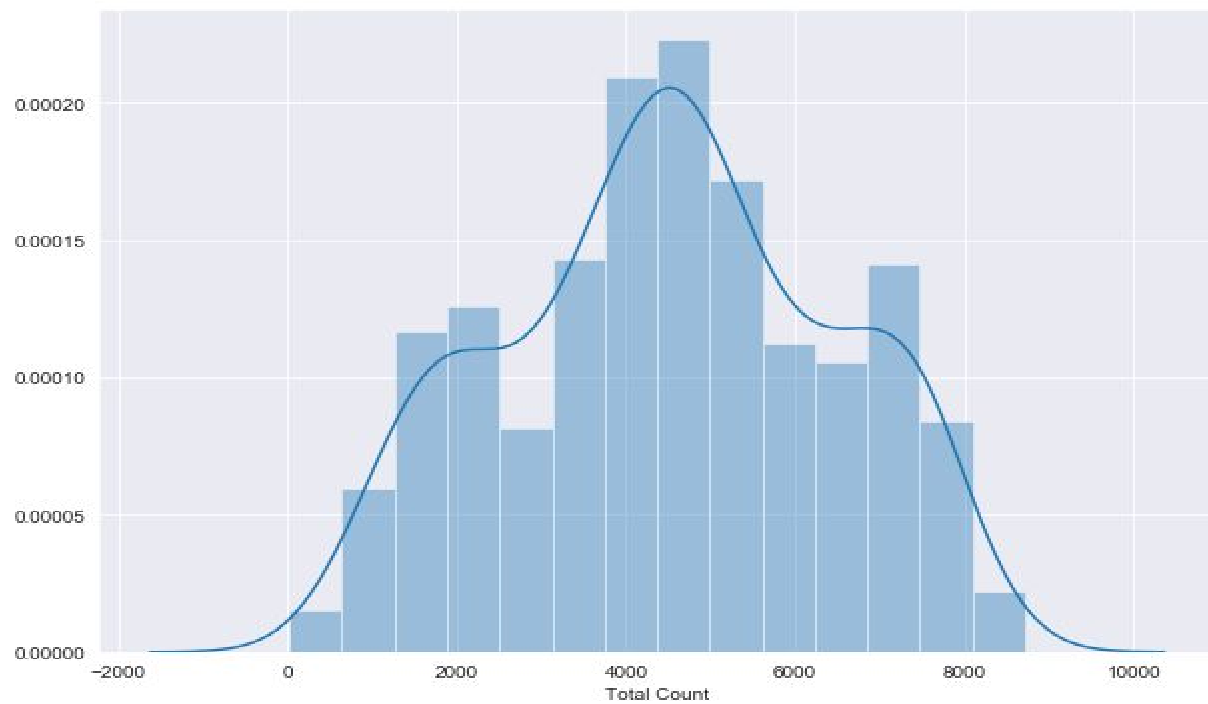
Model	Mean Absolute Error in Python	Root Mean Squared Error in Python	R Squared Score in Python	Adjusted R Squared Score in Python
Multiple Linear Regression	598	822	0.8495	0.8476
Decision Tree Regression	536	793	0.8597	0.8580
Random Forest Regression	432	614	0.9160	0.9150

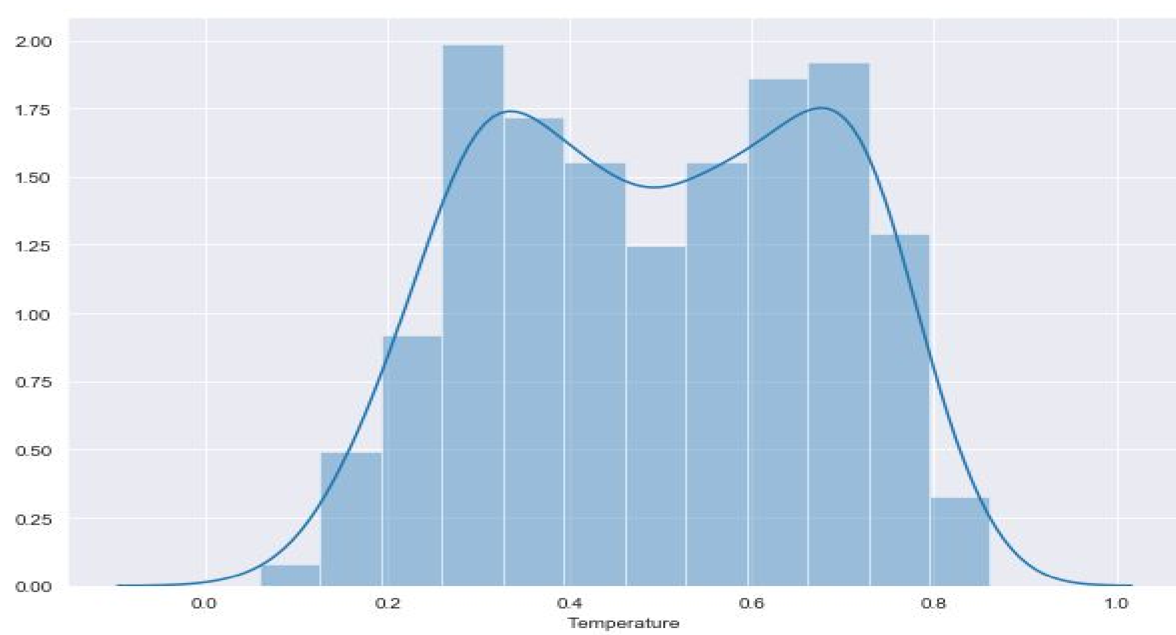
Model	Mean Absolute Error in R	Root Mean Squared Error in R	R Squared Score in R	Adjusted R Squared Score in R
Multiple Linear Regression	613	933	0.8105	0.8075
Decision Tree Regression	714	960	0.7515	0.7410
Random Forest Regression	429	653	0.9114	0.9040

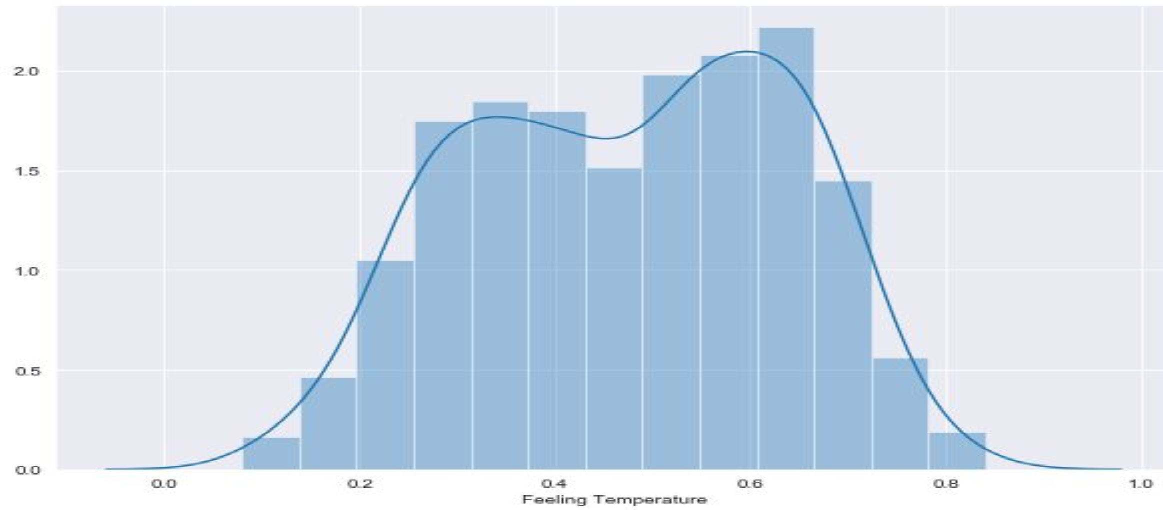
When we compare the Root Mean squared Error, Mean Absolute,R-squared Score,Adjusted R-squared Score of all three models, the Random Forest Model has lowest RMSE and MAE errors, and highest R-squared Score and Adjusted R-squared Score . So, finally a Random Forest Model is best for predicting the bike rental count on a daily basis.

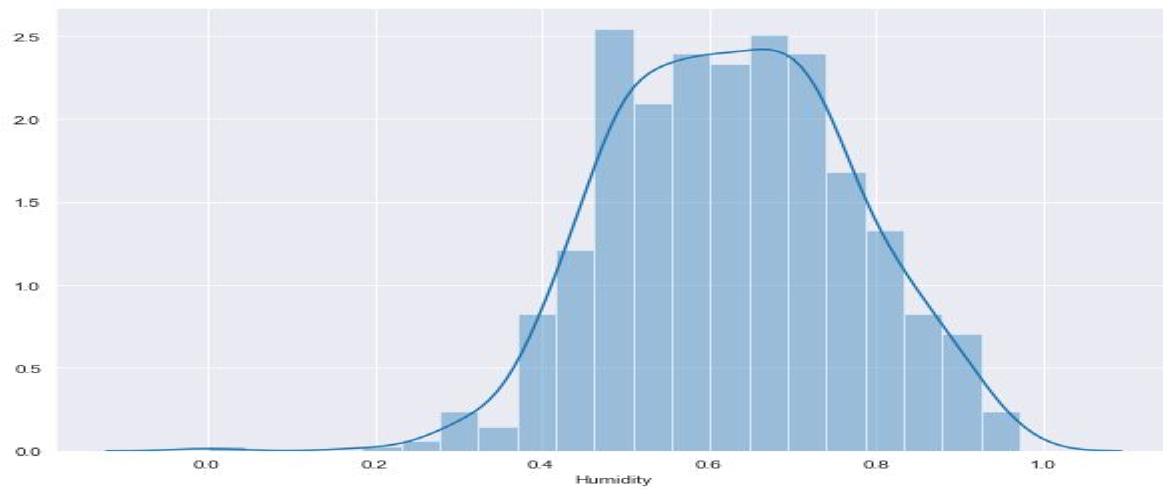
APPENDIX A EXTRA FIGURES

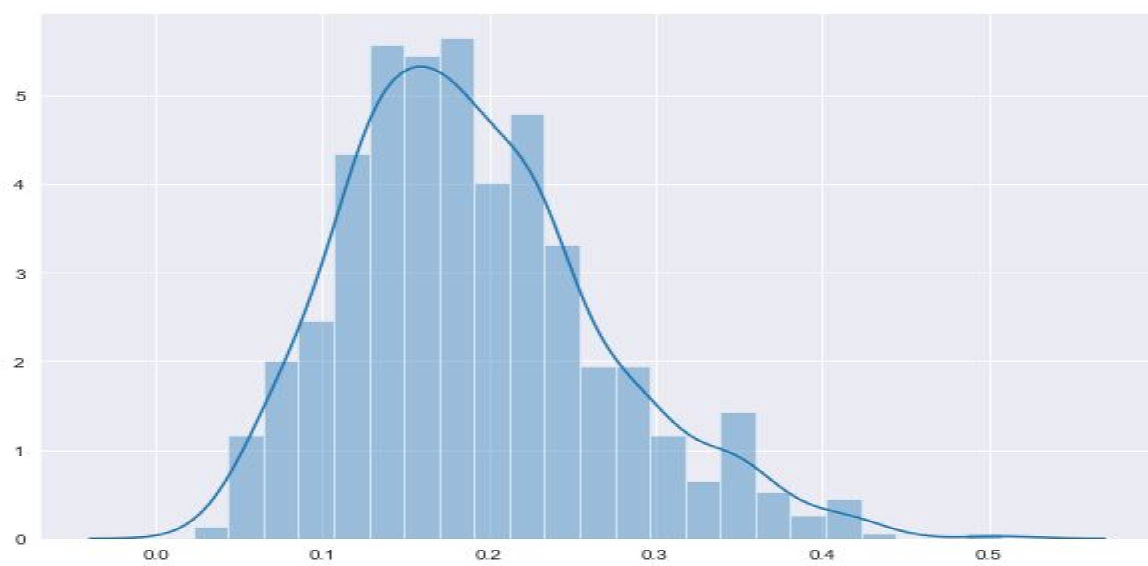
DISTRIBUTION PLOTS



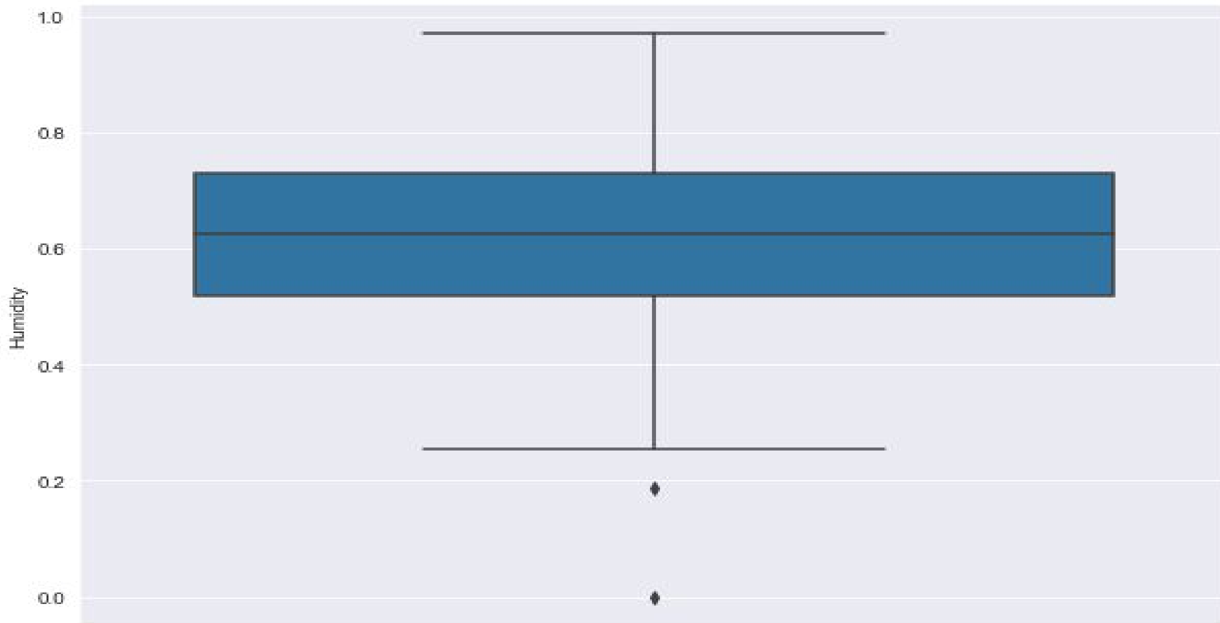



2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Feeling Temperature

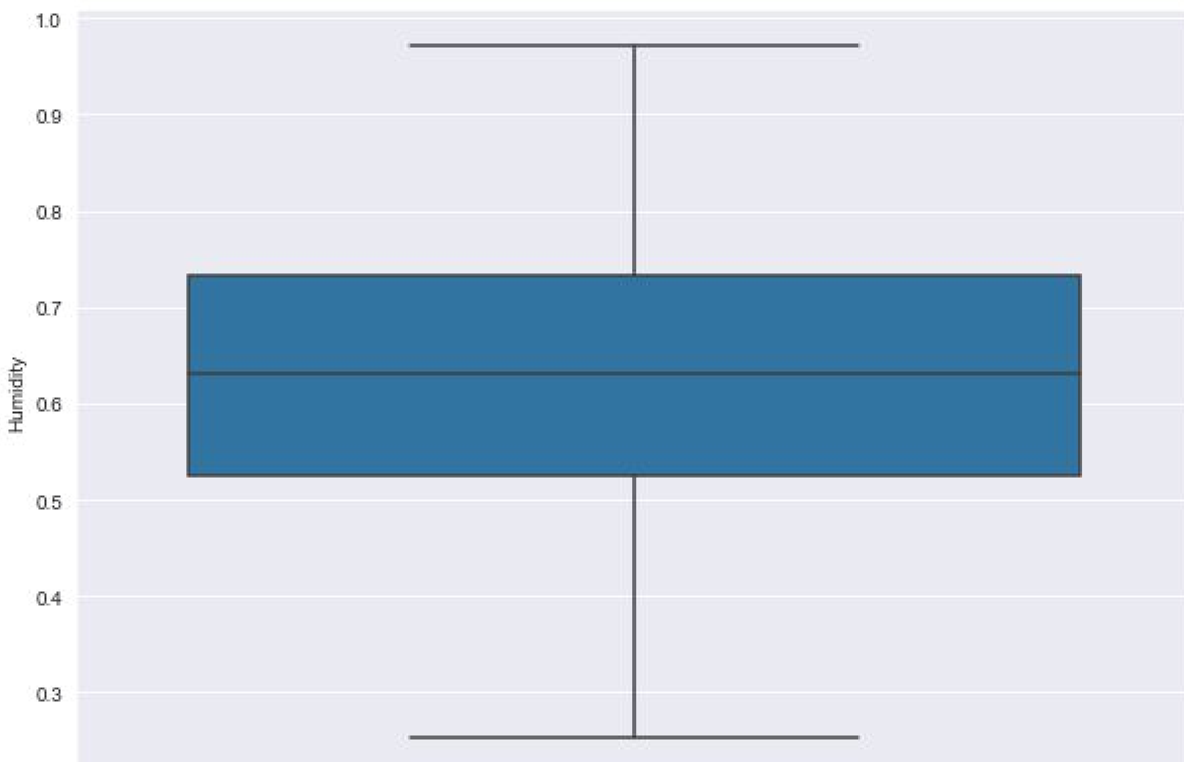

2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Humidity


5
4
3
2
1
0
0.0
0.1
0.2
0.3
0.4
0.5
Windspeed

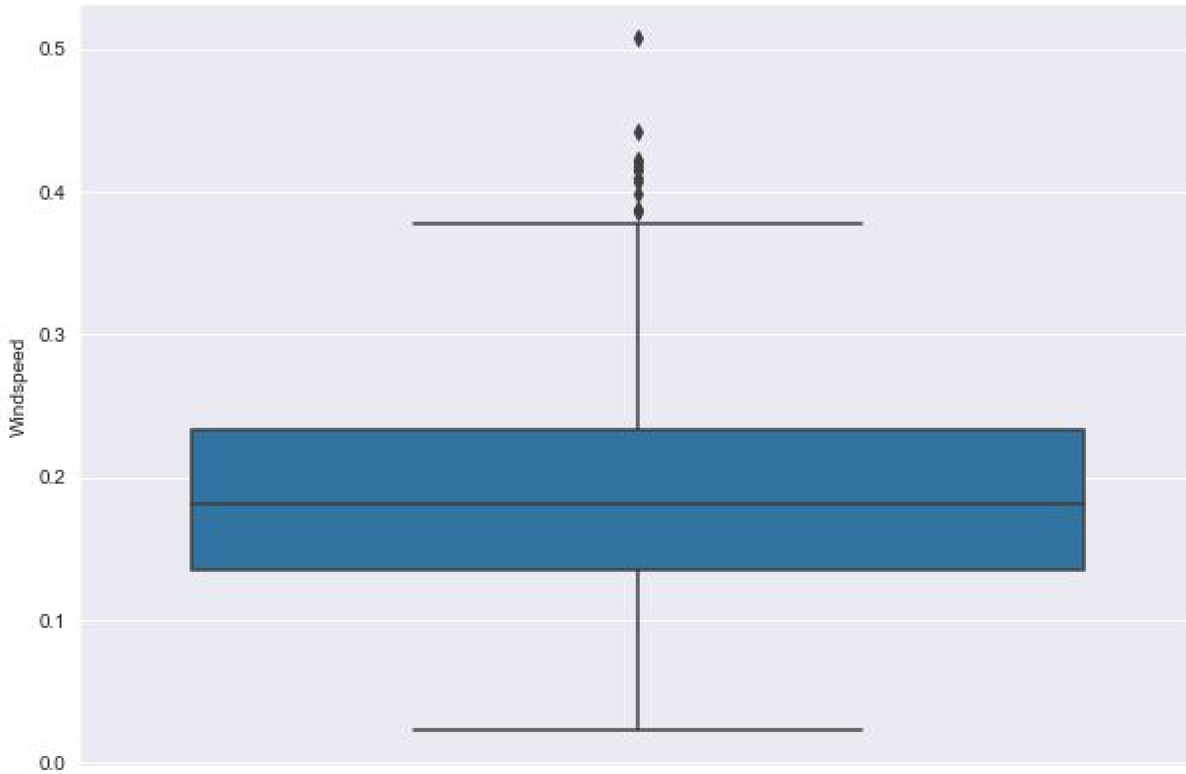
Effect of Outliers



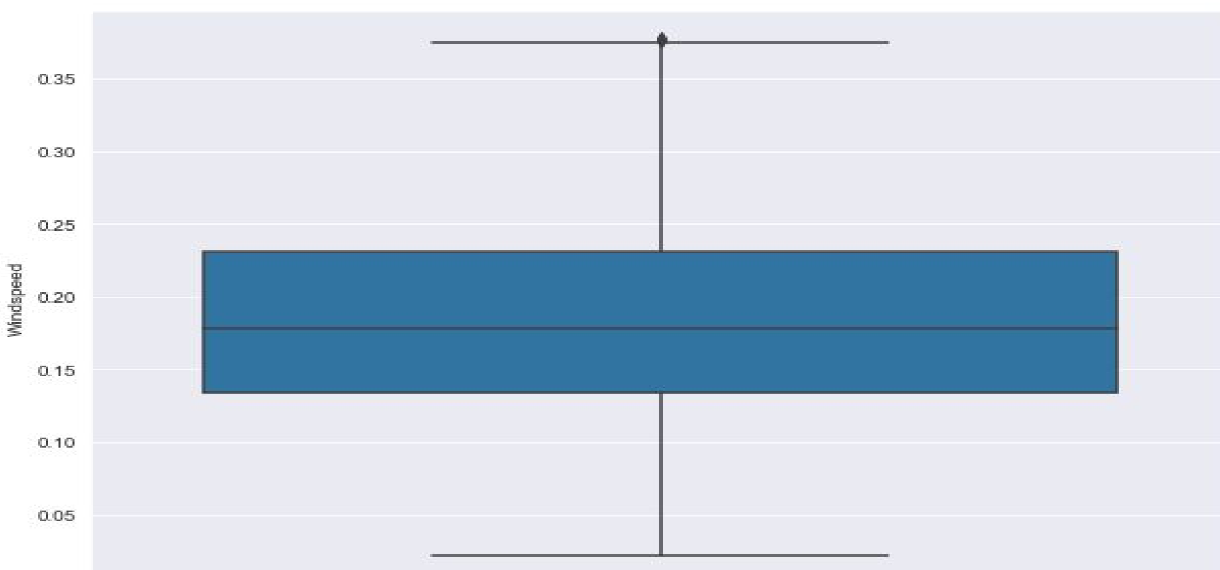
Before Removing outliers



After Removing outliers



Before Removing outliers



After Removing outliers

APPENDIX B CODES

PYTHON CODES

Importing The Libraries

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

import statsmodels.api as sm

sns.set_style('darkgrid')
```

Importing The Dataset

```
df = pd.read_csv('day.csv',index_col=0)

df.head()

df.info()

df=df.rename(columns={'dteday':'Date','season':'Season','yr':'Year','mnth':'Month','holiday':'Holiday','weekday':'Weekday','workingday':'Working Day', 'weathersit':'Weather Situation','temp':'Temperature','atemp':'Feeling Temperature','hum':'Humidity','windspeed':'Windspeed','casual':'Casual','registered':'Registered','cnt':'Total Count'})

df.describe()
```

Missing Value Analysis

```
Missing_Values = pd.DataFrame(df.isna().sum(),columns=['No Of Missing Values'])

Missing_Values
```

Data Visualisation

Season

plt.figure(figsize=(10,7))

sns.countplot(df['Season'],palette='rainbow')

plt.figure(figsize=(10,7))

sns.barplot(x='Season',y='Total Count',data=df,palette='rainbow')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Season',palette='rainbow')

Year

plt.figure(figsize=(10,7))

sns.countplot(df['Year'],palette='hls')

plt.figure(figsize=(10,7))

sns.barplot(x='Year',y='Total Count',data=df,palette='hls')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Year',palette='hls')

Month

plt.figure(figsize=(10,7))

sns.countplot(df['Month'],palette='rainbow')

plt.figure(figsize=(10,7))

sns.barplot(x='Month',y='Total Count',data=df,palette='rainbow',hue='Year')

plt.figure(figsize=(10,7))

sns.barplot(x='Month',y='Total Count',data=df,palette='rainbow',hue='Season')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Month',palette='rainbow')

Holiday

plt.figure(figsize=(10,7)

sns.countplot(df['Holiday'],palette='hls')

plt.figure(figsize=(10,7))

sns.barplot(x='Holiday',y='Total Count',data=df,palette='hls')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Holiday',palette='hls')

Weekday

plt.figure(figsize=(10,7))

sns.countplot(df['Weekday'],palette='rainbow')

plt.figure(figsize=(10,7))

sns.barplot(x='Weekday',y='Total Count',data=df,palette='rainbow')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Weekday',palette='rainbow')

Working Day

plt.figure(figsize=(10,7))

sns.countplot(df['Working Day'],palette='hls')

plt.figure(figsize=(10,7))

sns.barplot(x='Working Day',y='Total Count',data=df,palette='hls')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Working Day',palette='hls')

Weather Situation

plt.figure(figsize=(10,7))

sns.countplot(df['Weather Situation'],palette='rainbow')

plt.figure(figsize=(10,7))

sns.barplot(x='Weather Situation',y='Total Count',data=df,palette='rainbow')

for i in df[['Temperature', 'Feeling Temperature', 'Humidity','Windspeed']]:

plt.figure(figsize=(10,7))

sns.scatterplot(data=df,x=i,y='Total Count',hue='Weather Situation',palette='rainbow')

for i in df[['Year', 'Month', 'Holiday', 'Weekday', 'Working Day','Weather Situation']]:

plt.figure(figsize=(10,7))

sns.barplot(data=df,x=i,y='Total Count',hue='Season')

for i in df[['Season', 'Month', 'Holiday', 'Weekday', 'Working Day','Weather Situation']]:

plt.figure(figsize=(10,7))

sns.barplot(data=df,x=i,y='Total Count',hue='Year')

for i in df[['Season', 'Year', 'Holiday', 'Weekday', 'Working Day','Weather Situation']]:

plt.figure(figsize=(10,7))

sns.barplot(data=df,x=i,y='Total Count',hue='Month')

for i in df[['Season', 'Year','Month' , 'Weekday', 'Working Day','Weather Situation']]:

plt.figure(figsize=(10,7))

sns.barplot(data=df,x=i,y='Total Count',hue='Holiday')

for i in df[['Season', 'Year','Month' ,'Holiday', 'Working Day','Weather Situation']]:

plt.figure(figsize=(10,7))

sns.barplot(data=df,x=i,y='Total Count',hue='Weekday')

```
for i in df[['Season', 'Year','Month' ,'Holiday','Weekday' ,'Weather Situation']]:
plt.figure(figsize=(10,7))
sns.barplot(data=df,x=i,y='Total Count',hue='Working Day')
for i in df[['Season', 'Year','Month' ,'Holiday','Weekday' ,'Working Day']]:
plt.figure(figsize=(10,7))
sns.barplot(data=df,x=i,y='Total Count',hue='Weather Situation')
plt.figure(figsize=(10,7))
sns.distplot(df['Total Count'])
plt.figure(figsize=(10,7))
sns.distplot(df['Temperature'])
plt.figure(figsize=(10,7))
sns.distplot(df['Feeling Temperature'])
plt.figure(figsize=(10,7))
sns.distplot(df['Humidity'])
plt.figure(figsize=(10,7))
sns.distplot(df['Windspeed'])
```

Outlier Analysis

```
plt.figure(figsize=(10,7))
sns.boxplot(data=df,y='Total Count')
plt.figure(figsize=(10,7))
sns.boxplot(data=df,y='Temperature')
plt.figure(figsize=(10,7))
sns.boxplot(data=df,y='Feeling Temperature')
plt.figure(figsize=(10,7))
sns.boxplot(data=df,y='Humidity')
```

```
plt.figure(figsize=(10,7))

sns.boxplot(data=df,y='Windspeed')
```

Removing Outliers

```
columns = ["Temperature",'Feeling Temperature',"Humidity","Windspeed"]

for i in columns:

    print (i)

    q75,q25 = np.percentile(df.loc[:,i],[75,25])

    iqr = q75-q25


    min = q25 - (iqr*1.5)

    max = q75 + (iqr*1.5)

    print (min)

    print (max)


    df = df.drop(df[df.loc[:,i] < min].index)

    df = df.drop(df[df.loc[:,i] > max].index)
```

Analysis After Removing Outliers

```
plt.figure(figsize=(10,7))

sns.boxplot(data=df,y='Temperature')

plt.figure(figsize=(10,7))

sns.boxplot(data=df,y='Feeling Temperature')

plt.figure(figsize=(10,7))

sns.boxplot(data=df,y='Humidity')
```

```
plt.figure(figsize=(10,7))

sns.boxplot(data=df,y='Windspeed')
```

Chi Sqaure Test

```
from scipy.stats import chi2_contingency

from scipy.stats import chi2

stat, p, dof, expected = chi2_contingency(df.drop('Date',axis=1))

print('dof=%d' % dof)

print(expected)

# interpret test-statistic

prob = 0.95

critical = chi2.ppf(prob, dof)

print('probability=%.3f, critical=%.3f, stat=%.3f' % (prob, critical, stat))

if abs(stat) >= critical:

	print('Variables are Dependent (reject H0)')

else:

	print('Variables are Dependent Independent (fail to reject H0)')

# interpret p-value

alpha = 1.0 - prob

print('significance=%.3f, p=%.3f' % (alpha, p))

if p <= alpha:

	print('Variables are Dependent Dependent (reject H0)')

else:

	print('Variables are Dependent Independent (fail to reject H0)')
```

Normality Test

```
import scipy

from scipy import stats

plt.figure(figsize=(10,7))

stats.probplot(df['Total Count'].tolist(),dist='norm',plot=plt)

plt.show()
```

Feature Selection

```
plt.figure(figsize=(10,10))

sns.heatmap(df.corr(),annot=True,cmap='viridis')

df.corr()['Total Count']

df.columns

#Dropping Casual and Registered due to their multicollinearity

#Working Day due low correlation with Total Count

#Dropping Feeling Temperature due to multicollinearity with Temperature

X = df[['Season', 'Year', 'Month', 'Holiday', 'Weekday',

        'Weather Situation', 'Temperature', 'Humidity',

        'Windspeed']]

y = df['Total Count']
```

Splitting The Dataset

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
```

Multiple Linear Regression

```
#Summary of Linear Regression by Ordinary Least Square Method
```

```
mod = sm.OLS(y, X)    # Describe model
res = mod.fit()       # Fit model
print(res.summary())
#Fitting The Linear Regression on Training Set
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train, y_train)
#Predictiion by Linear Regression Model on Test Set
pred_lr = lr.predict(X_test)
#Cross Validation Score and Standard Deviation
from sklearn.model_selection import cross_val_score
CV_Score = cross_val_score(lr,  X = X_test, y = y_test, cv = 5)
print("CV_Score: {:.2f} %".format(CV_Score.mean()*100))
print("Standard Deviation: {:.2f} %".format(CV_Score.std()*100))
#  The intercept
print(lr.intercept_)
#Coefficient Of Independent Variables in Regression
col = ['Season', 'Year', 'Month', 'Holiday', 'Weekday',
        'Weather Situation', 'Temperature', 'Humidity',
        'Windspeed']
coeff_df = pd.DataFrame(lr.coef_,col,columns=['Coefficient'])
coeff_df
#Visualisation Of Linear Regression Prediction Model
plt.scatter(y_test, pred_lr)
plt.plot(y_test,y_test,color='red')
```

```
plt.title("Scatter Plot with Linear fit");

#Visualisation Of Residual Vs Actual

fig,ax=plt.subplots(figsize=(15,8))

ax.scatter(y_test,y_test-pred_lr)

ax.axhline(lw=2,color='black')

ax.set_title('Cross validation prediction plot')

ax.set_xlabel('Observed')

ax.set_ylabel('Residual')

plt.show()

from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

#Model Evaluatiom

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_lr))

print('\n')

print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,pred_lr)))

print('\n')

print('r2_score:',r2_score(y_test,pred_lr))

print('\n')

print('Accuracy Of The Linear Regression Model:',lr.score(X_test,y_test)*100,'%')

#Adjusted r2 Score

n=X.shape[0]

p=X.shape[1]

R2 = r2_score(y_test,pred_lr)

Adj_r2 = 1-(1-R2)*(n-1)/(n-p-1)

Adj_r2
```

Decision Tree Regression

```
#Fitting The Decision Tree Regression on Training Set
from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor(random_state = 0)
dtr.fit(X_train, y_train)
#Prediction by Decision Tree Regression on Test Set
pred_dtr = dtr.predict(X_test)
#Cross Validation Score and Standard Deviation
CV_Score = cross_val_score(dtr, X = X_test, y = y_test, cv = 5)
print("CV_Score: {:.2f} %".format(CV_Score.mean()*100))
print("Standard Deviation: {:.2f} %".format(CV_Score.std()*100))
#Visualisation Of Decision Tree Regression Prediction Model
plt.scatter(y_test, pred_dtr)
plt.plot(y_test,y_test,color='red')
plt.title("Scatter Plot with Decision Tree Regression");
#Visualisation Of Residual Vs Actual
fig,ax=plt.subplots(figsize=(15,8))
ax.scatter(y_test,y_test-pred_dtr)
ax.axhline(lw=2,color='black')
ax.set_title('Residual vs Actual plot')
ax.set_xlabel('Observed')
ax.set_ylabel('Residual')
plt.show()
#Importing the Decision Tree Plot Libraries
from IPython.display import Image
from sklearn.externals.six import StringIO
```

```
from sklearn.tree import export_graphviz

import pydot

features = list(df[['Season', 'Year', 'Month', 'Holiday', 'Weekday',

            'Weather Situation', 'Temperature', 'Humidity',

            'Windspeed']])

features

#Decision Tree Plot

dot_data = StringIO()

export_graphviz(dtr, out_file=dot_data,feature_names=features,filled=True,rounded=True)

graph = pydot.graph_from_dot_data(dot_data.getvalue())

Image(graph[0].create_png())

#Model Evaluation

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_dtr))

print('\n')

print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,pred_dtr)))

print('\n')

print('r2_score:',r2_score(y_test,pred_dtr))

print('\n')

print('Accuracy Of The Decision Tree Regression Model:',dtr.score(X_test,y_test)*100,'%')

#Adjusted r2 Score

n=X.shape[0]

p=X.shape[1]

R2 = r2_score(y_test,pred_dtr)

Adj_r2 = 1-(1-R2)*(n-1)/(n-p-1)

Adj_r2
```

Random Forest Regression Model

```
#Fitting The Random Forest Regression on Training Set

from sklearn.ensemble import RandomForestRegressor

rfr = RandomForestRegressor(n_estimators = 10, random_state = 0)

rfr.fit(X_train, y_train)

#Prediction by Random Forest Regression on Test Set

pred_rfr = rfr.predict(X_test)

#Cross Validation Score and Standard Deviation

CV_Score = cross_val_score(rfr, X = X_test, y = y_test, cv = 5)

print("CV_Score: {:.2f} %".format(CV_Score.mean()*100))

print("Standard Deviation: {:.2f} %".format(CV_Score.std()*100))

#Visualisation Of Random Forest Regression Prediction Model

plt.scatter(y_test, pred_rfr)

plt.plot(y_test,y_test,color='red')

plt.title("Scatter Plot with Random Forest Regression");

#Visualisation Of Residual Vs Actual

fig,ax=plt.subplots(figsize=(15,8))

ax.scatter(y_test,y_test-pred_rfr)

ax.axhline(lw=2,color='black')

ax.set_title('Residual Vs Actual plot')

ax.set_xlabel('Observed')

ax.set_ylabel('Residual')

plt.show()

#Visualisation Of Relative Importance of Features

features=df.columns
```

```
importances = rfr.feature_importances_

indices = np.argsort(importances)

plt.figure(figsize=(10,7))

plt.figure(1)

plt.title('Feature Importances')

plt.barh(range(len(indices)), importances[indices], color='b', align='center')

plt.yticks(range(len(indices)), features[indices])

plt.xlabel('Relative Importance')

#Model Evaluation

print('Mean Absolute Error:',mean_absolute_error(y_test,pred_rfr))

print('\n')

print('Root Mean Squared Error:',np.sqrt(mean_squared_error(y_test,pred_rfr)))

print('\n')

print('r2_score:',r2_score(y_test,pred_rfr))

print('\n')

print('Accuracy Of The Random Forest Regression Model:',rfr.score(X_test,y_test)*100,'%')

#Adjusted r2 Score

n=X.shape[0]

p=X.shape[1]

R2 = r2_score(y_test,pred_rfr)

Adj_r2 = 1-(1-R2)*(n-1)/(n-p-1)

Adj_r2

#Random Forest Regression has highest R2 score and accuracy so Final Model is Random Forest Regression
```

Predicting a sample input

```
#Predicting a sample input

#('Season'=1,'Year'=1,'Month'=4,'Holiday'=1,'Weekday'=1,'Weather_Situation'=2,'Temperature'=27
,'Humidity'=0.8,'Windspeed'=0.16)

Prediction = rfr.predict([[1,1,4,1,1,2,27,0.8,0.16]])

print('Prediction of Total Bike Rental Count a sample input is',Prediction[0])
```

R CODES

```
#Importing Important Libraries
library("ggplot2")
library("corrgram")
library( "DMwR")
library( "usdm")
library( "caret")
library( "randomForest")
library( "e1071")
library( "DataCombine")
library("doSNOW")
library("inTrees")
library( "rpart.plot")
library("rpart")
library(caTools)
library(rpart)
#Setting Up The Working Directory
setwd('C:/Users/Raaz/Documents/Data Science/Project')
#Importing The Dataset
df <- read.csv('day.csv')
head(df)
nrow(df)
ncol(df)
```

```
str(df)

ls(df)

summary(df)

#Removing instant,date

df$instant <- NULL

df$dteday <- NULL

#Changing Label Name

names(df) <- c('Season','Year','Month','Holiday',

          'Weekday','Workingday','Weather_Situation',

          'Temperature','Feeling_Temperature','Humidity','Windspeed',

          'Casual','Registered','Total_Count')

head(df)

#Missing Value Analysis

missing_val<-data.frame(apply(df,2,function(x){sum(is.na(x))}))

names(missing_val)[1]='missing_val'

Missing_val

#Data Visualization

#Season

ggplot(df, aes(x=Season)) + geom_bar(fill="firebrick") + labs(title="Total Count by
Season")

ggplot(df,aes(x=Season,y=Total_Count,fill=Season))+theme_bw()+geom_col()+labs(x='
Season',

                    y='Total_Count',title='Season wise monthly distribution of counts')

qplot(data=df,x=Temperature,y=Total_Count,colour=Season,main="Temperature VS
Total Count")

qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Season,main="Feeling
Temperature VS Total Count")

qplot(data=df,x=Humidity,y=Total_Count,colour=Season,main="Humidity VS Total
Count")

qplot(data=df,x=Windspeed,y=Total_Count,colour=Season,main="Windspeed VS
Total Count")
```

```
#Year
ggplot(df, aes(x=Year)) + geom_bar(fill="firebrick") + labs(title="Total Count by Year")
ggplot(df,aes(x=Year,y=Total_Count,fill=Year))+theme_bw()+geom_col()+labs(x='Year',
                    y='Total_Count',title='Year wise monthly distribution of counts')
qplot(data=df,x=Temperature,y=Total_Count,colour=Year,main="Temperature VS
Total Count")
qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Year,main="Feeling
Temperature VS Total Count")
qplot(data=df,x=Humidity,y=Total_Count,colour=Year,main="Humidity VS Total
Count")
qplot(data=df,x=Windspeed,y=Total_Count,colour=Year,main="Windspeed VS Total
Count")
#Month
ggplot(df, aes(x=Month)) + geom_bar(fill="firebrick") + labs(title="Total Count by
Month")
ggplot(df,aes(x=Month,y=Total_Count,fill=Month))+theme_bw()+geom_col()+labs(x='
Month',
                    y='Total_Count',title='Month wise monthly distribution of counts')
qplot(data=df,x=Temperature,y=Total_Count,colour=Month,main="Temperature VS
Total Count")
qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Month,main="Feeling
Temperature VS Total Count")
qplot(data=df,x=Humidity,y=Total_Count,colour=Month,main="Humidity VS Total
Count")
qplot(data=df,x=Windspeed,y=Total_Count,colour=Month,main="Windspeed VS Total
Count")
#Holiday
ggplot(df, aes(x=Holiday)) +  geom_bar(fill="firebrick") + labs(title="Total Count by
Holiday")
ggplot(df,aes(x=Holiday,y=Total_Count,fill=Holiday))+theme_bw()+geom_col()+labs(x=
'Holiday',
          y='Total_Count',title='Holiday wise monthly distribution of counts')
```

```
qplot(data=df,x=Temperature,y=Total_Count,colour=Holiday,main="Temperature VS Total Count")
qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Holiday,main="Feeling Temperature VS Total Count")
qplot(data=df,x=Humidity,y=Total_Count,colour=Holiday,main="Humidity VS Total Count")
qplot(data=df,x=Windspeed,y=Total_Count,colour=Holiday,main="Windspeed VS Total Count")
#Weekday
ggplot(df, aes(x=Weekday)) +  geom_bar(fill="firebrick") + labs(title="Total Count by Weekday")
ggplot(df,aes(x=Weekday,y=Total_Count,fill=Weekday))+theme_bw()+geom_col()+labs(x='Weekday',
        y='Total_Count',title='Weekday wise monthly distribution of counts')
qplot(data=df,x=Temperature,y=Total_Count,colour=Weekday,main="Temperature VS Total Count")
qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Weekday,main="Feeling Temperature VS Total Count")
qplot(data=df,x=Humidity,y=Total_Count,colour=Weekday,main="Humidity VS Total Count")
qplot(data=df,x=Windspeed,y=Total_Count,colour=Weekday,main="Windspeed VS Total Count")
#Workingday
ggplot(df, aes(x=Workingday)) + geom_bar(fill="firebrick")+ labs(title="Total Count by Workingday")
ggplot(df,aes(x=Workingday,y=Total_Count,fill=Workingday))+theme_bw()+geom_col()+labs(x='Workingday',
        y='Total_Count',title='Workingday wise monthly distribution of counts')
qplot(data=df,x=Temperature,y=Total_Count,colour=Workingday,main="Temperature VS Total Count")
qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Workingday,main="Feeling Temperature VS Total Count")
qplot(data=df,x=Humidity,y=Total_Count,colour=Workingday,main="Humidity VS Total Count")
```

```
qplot(data=df,x=Windspeed,y=Total_Count,colour=Workingday,main="Windspeed VS Total Count")

#Weather_Situation

ggplot(df, aes(x=Weather_Situation)) +  geom_bar(fill="firebrick") + labs(title="Total Count by Weather_Situation")

ggplot(df,aes(x=Weather_Situation,y=Total_Count,fill=Weather_Situation))+theme_bw()+geom_col()+labs(x='Weather_Situation',
        y='Total_Count',title='Weather_Situation wise monthly distribution of counts')

qplot(data=df,x=Temperature,y=Total_Count,colour=Weather_Situation,main="Temperature VS Total Count")

qplot(data=df,x=Feeling_Temperature,y=Total_Count,colour=Weather_Situation,main="Feeling Temperature VS Total Count")

qplot(data=df,x=Humidity,y=Total_Count,colour=Weather_Situation,main="Humidity VS Total Count")

qplot(data=df,x=Windspeed,y=Total_Count,colour=Weather_Situation,main="Windspeed VS Total Count")

#Outliers Analysis

#Box plot for Total Count

boxplot(df$Total_Count,main="Total_Count",sub=paste(boxplot.stats(df$Total_Count)$out))

#Box plot for Temperature outliers

boxplot(df$Temperature,main="Temperature",sub=paste(boxplot.stats(df$Temperature)$out))

#Box plot for Feeling Temperature outliers

boxplot(df$Feeling_Temperature,main="Feeling Temperature",sub=paste(boxplot.stats(df$Feeling_Temperature)$out))

#Box plot for Humidity outliers

boxplot(df$Humidity,main="Humidity",sub=paste(boxplot.stats(df$Humidity)$out))

#Box plot for Windspeed outliers

boxplot(df$Windspeed,main="Windspeed",sub=paste(boxplot.stats(df$Windspeed)$out))

#Removing Outliers
```

```
# Removing Humidity outliers
df <- df[df$Humidity > quantile(df$Humidity, .25) - 1.5*IQR(df$Humidity) &
         df$Humidity < quantile(df$Humidity, .75) + 1.5*IQR(df$Humidity), ]
# Removing Windspeed outliers
df <- df[df$Windspeed > quantile(df$Windspeed, .25) - 1.5*IQR(df$Windspeed) &
         df$Windspeed < quantile(df$Windspeed, .75) + 1.5*IQR(df$Windspeed), ]
nrow(df)
#Outliers Analysis After Removing Outliers
#Box plot for Humidity outliers
boxplot(df$Humidity,main="Humidity",sub=paste(boxplot.stats(df$Humidity)$out))
#Box plot for Windspeed outliers
boxplot(df$Windspeed,main="Windspeed",sub=paste(boxplot.stats(df$Windspeed)$
out))
#Normality Test
#Quintle-Quintle normal plot
qqnorm(df$Total_Count)
#Quintle-Quintleline
qqline(df$Total_Count)
#Feature Selection
library(corrgram)
corrgram(df[,1:11], order=TRUE, lower.panel=panel.shade,
         upper.panel=panel.pie, text.panel=panel.txt,
         main = 'CORRELATION PLOT')
#Dropping Casual and Registered due to their multicollinearity
#Working Day due low correlation with Total Count
#Dropping Feeling Temperature due to multicollinearity with Temperature
df_new <-subset(df,select=c('Season', 'Year', 'Month', 'Holiday', 'Weekday',
                  'Weather_Situation', 'Temperature', 'Humidity',
                  'Windspeed','Total_Count'))
head(df_new)
```

```
#Chi Square Test
H0<- c("Variables are Independent")
H1 <- c("Variables are Dependent")
res<-chisq.test(df)
p_value<-res$p.value
alpha<-0.95
if (alpha>p_value) {
  print("Rejecting The Null Hypothesis")
  print(H1)
} else {
  print("Rejecting The Null Hypothesis")
  print(H0)
}#Splitting The Dataset
split = sample.split(df_new$Total_Count, SplitRatio = 0.8)
training_set = subset(df_new, split == TRUE)
test_set = subset(df_new, split == FALSE)
test_set
```

#Linear Regression model

```
#Fitting The Linear Regression model on Training Set
lr_model = lm(Total_Count ~. , data = training_set)
summary(lr_model)#Prediction by Linear Regression model on Training Set
pred_lr = predict(lr_model, test_set[,-10])# Visualizing the Predicted Test set results
ggplot() +
  geom_point(aes(x = test_set$Total_Count, y = pred_lr), colour = 'red') +
  ggtitle('Linear Regression Model model') +
  xlab('Actual values') +
  ylab('Predicted values')
#Residual plot
residuals<- test_set$Total_Count-pred_lr
```

```
plot(test_set$Total_Count,residuals,xlab='Observed',ylab='Residuals',
          main='Residual plot in Linear Regression Model',col='blue')
abline(0,0)# Cross Validation of Linear Regression Model
set.seed(123)
train.control <- trainControl(method = "cv", number = 5)
# Train the model
model <- train(Total_Count ~., data = training_set, method = "lm",
          trControl = train.control)
# Summarize the results
print(model)#Evaluation of Linear Regression Model
postResample(test_set$Total_Count, pred_lr)
rmse<-RMSE(test_set$Total_Count, pred_lr)
print(rmse)
mae<-MAE(test_set$Total_Count, pred_lr)
print(mae)
# R²
summary(lr_model)$r.squared
# adjusted R²
summary(lr_model)$adj.r.squared
```

#Decision Tree Regression

```
set.seed(121)
#Fitting the Decision Tree Regression Model on Training Dataset
dt_model = rpart(Total_Count~. , data = training_set, method = "anova")
summary(dt_model)
#Decision Tree Plot
plt = rpart.plot(dt_model, type = 5, digits = 2, fallen.leaves = TRUE)
#Predictions by Decision Tree Regression Model on Test Dataset
pred_dtr = predict(dt_model, test_set[,-10])
# # Visualizing the Predicted Test set results
```

```
ggplot() +
  geom_point(aes(x = test_set$Total_Count, y = pred_dtr), colour = 'red') +
  ggtitle('Decision Tree model') +
  xlab('Actual values') +
  ylab('Predicted values')
#Residual plot
residuals<- test_set$Total_Count-pred_dtr
plot(test_set$Total_Count,residuals,xlab='Observed',ylab='Residuals',
        main='Residual plot in Linear Regression Model',col='blue')
abline(0,0)
# Cross Validation of Decision Tree Regression Model
set.seed(123)
train.control <- trainControl(method = "cv", number = 10)
# Train the model
model <- train(Total_Count ~., data = training_set, method = 'rpart',
        trControl = train.control)
# Summarize the results
print(model)
#Evaluation of Model
postResample(pred_dtr,test_set$Total_Count)
rmse<-RMSE(test_set$Total_Count, pred_dtr)
print(rmse)
mae<-MAE(test_set$Total_Count, pred_dtr)
print(mae)
rss <- sum((pred_dtr - test_set$Total_Count ) ^ 2)  ## residual sum of squares
tss <- sum((test_set$Total_Count - mean(test_set$Total_Count)) ^ 2)  ## total sum of squares
rsq <- 1 - rss/tss
print(rsq)#R²
# adjusted R²
```

```
n <- nrow(test_set)

p <- ncol(test_set)

Adj_r2 <- 1-(1-rsq)*(n-1)/(n-p-1)

print(Adj_r2)
```

#Random Forest

```
set.seed(101)

#Fitting the Random Forest Regression Model on Training Dataset

rf_model = randomForest(Total_Count~. , data = training_set, importance = TRUE, ntree = 1000)

rf_model

#Error plotting

plot(rf_model)

#Variable Importance plot

varImpPlot(rf_model)

#Predictions by Random Forest on the Test Dataset

pred_rfr = predict(rf_model, test_set[,-10])

# Visualizing the  Predicted Test set results

ggplot() +

  geom_point(aes(x = test_set$Total_Count, y = pred_rfr), colour = 'red') +

  ggtitle('Random Forest model') +

  xlab('Actual values') +

  ylab('Predicted values')

#Residual plot

residuals<test_set$Total_Count-pred_rfr

plot(test_set$Total_Count,residuals,xlab='Observed',ylab='Residuals',main='Random Forest Residual plot',col='blue')

abline(0,0)

 Cross Validating the Random Forest Model

set.seed(123)

train.control <- trainControl(method = "cv", number = 10)
```

```
# Train the model

model <- train(Total_Count ~., data = training_set, method = 'ranger',  trControl = train.control)

# Summarize the results

print(model)#Importance of Independent Variables

importance(rf_model)#Evaluation Of Model

postResample(pred_rfr,test_set$Total_Count)

rmse<-RMSE(test_set$Total_Count, pred_rfr)

print(rmse)

mae<-MAE(test_set$Total_Count, pred_rfr)

print(mae)

# R²

rss <- sum((pred_rfr - test_set$Total_Count ) ^ 2)  ## residual sum of squares

tss <- sum((test_set$Total_Count - mean(test_set$Total_Count)) ^ 2)  ## total sum of squares

rsq <- 1 - rss/tss

print(rsq)#R²

# adjusted R²

n <- nrow(test_set)

p <- ncol(test_set)

Adj_r2 <- 1-(1-rsq)*(n-1)/(n-p-1)

print(Adj_r2)#Random Forest Regression has highest R2 score and accuracy so Final Model is Random Forest Regression#Predicting a sample input

#Input('Season'=1, 'Year'=1, 'Month'=4, 'Holiday'=1, 'Weekday'=1,'W

eather_Situation'=2, 'Temperature'=27,'Humidity'=0.8,'Windspeed'=0.16)Prediction = predict(rf_model, data.frame('Season'=1, 'Year'=1, 'Month'=4, 'Holiday'=1, 'Weekday'=1,'Weather_Situation'=2, 'Temperature'=27, 'Humidity'=0.8,'Windspeed'=0.16))print('Prediction of Total Bike Renatl Count a sample input is')

print(Prediction)
```

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. An Introduction to Statistical Learning. Vol. 6. Springer. Wickham, Hadley. 2009. Ggplot2: Elegant Graphics for Data Analysis. Springer Science & Business Media.