Prediction of Bike Rental count

Maltesh B Shibarad

1/11/2019

Contents

Page No

Chapter 1: Introduction

1.1 Problem Statement

The aim of this project is to predict the count of bike rentals based on the seasonal and environmental settings. By predicting the count, it would be possible to help accommodate in managing the number of bikes required on a daily basis, and being prepared for high demand of bikes during peak periods.

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, user is able to easily rent a bike from a particular position and return back to another position. Currently, there are about over 500 bike-sharing programs around the world which are composed of over 500 thousands bicycles. Today, there exists great interest in these systems due to their important role in traffic, environmental and health issues.

Apart from interesting real-world applications of bike sharing systems, the characteris- tics 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, de- parture and arrival position is explicitly recorded in these systems. This feature turns 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 important events in the city could be de- tected via monitoring these data.

What is bike rental??

A bike rental or bike hire business rents out bicycles for short periods of time, usually for a few hours. Most rentals are provided by bike shops as a sideline to their main busi- nesses of sales and service, but some shops specialize in rentals.

As with car rental, bicycle rental shops primarily serve people who do not have access to a vehicle, typically travelers and particularly tourists. Specialized bicycle rental shops therefore typically operate at beaches, parks, or other locations those tourists frequent. In this case, the fees are set to encourage renting the bikes for a few hours at a time, rarely more than a day.

1.2 Data

The goal is to build regression models which will predict the number of bikes used based on the environmental and season behavior. Given below is a sample of the data set that we are using to predict the number of bikes:

Table 1.1: Bike Rental first few Data (Columns: 1-20)

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersi	temp	atemp	hum	windspee	casual	registerec	cnt
2	1	01-01-2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
3	2	02-01-2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
4	3	03-01-2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
5	4	04-01-2011	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
6	5	05-01-2011	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
7	6	06-01-2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
8	7	07-01-2011	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
9	8	08-01-2011	1	0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891	959
10	9	09-01-2011	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768	822
11	10	10-01-2011	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321
12	11	11-01-2011	1	0	1	0	2	1	2	0.169091	0.191464	0.686364	0.122132	43	1220	1263
13	12	12-01-2011	1	0	1	0	3	1	1	0.172727	0.160473	0.599545	0.304627	25	1137	1162
14	13	13-01-2011	1	0	1	0	4	1	1	0.165	0.150883	0.470417	0.301	38	1368	1406
15	14	14-01-2011	1	0	1	0	5	1	1	0.16087	0.188413	0.537826	0.126548	54	1367	1421
16	15	15-01-2011	1	0	1	0	6	0	2	0.233333	0.248112	0.49875	0.157963	222	1026	1248
17	16	16-01-2011	1	0	1	0	0	0	1	0.231667	0.234217	0.48375	0.188433	251	953	1204
18	17	17-01-2011	1	0	1	1	1	0	2	0.175833	0.176771	0.5375	0.194017	117	883	1000
19	18	18-01-2011	1	0	1	0	2	1	2	0.216667	0.232333	0.861667	0.146775	9	674	683
20	19	19-01-2011	1	0	1	0	3	1	2	0.292174	0.298422	0.741739	0.208317	78	1572	1650
21	20	20-01-2011	1	0	1	0	4	1	2	0.261667	0.25505	0.538333	0.195904	83	1844	1927

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

S.NO	VARIABLE
1	Instant
2	Dteday
3	Season
4	Yr
5	Month
6	Holiday
7	Weekday
8	Working day
9	Weather sit
10	Temp
11	Atemp
12	Hum
13	Wind speed

Table 1.2: Predictor variables

Chapter 2: Methodology

2.1 Pre-Processing

A predictive model requires that we look at the data before we start to create a model. However, in data mining, looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is known as Exploratory Data Analysis.

To start this process we will first try and look at all the distributions of the Numeric variables.
Most analysis like regression, require the data to be normally distributed.

2.2 Missing Value Analysis

Missing Value Analysis: The Missing Value Analysis procedure performs three primary functions: Describes the pattern of missing data. Fills in (imputes) missing values with estimated values using regression or EM methods; however, multiple imputation is generally considered to provide more accurate results.

Below fig illustrate no missing value present in the dataset provided.

```
missing_values = sapply(day,function(x){sum(is.na(x))})
missing_values
  season       year      month    weekday workingday weathersit       temp   humidity  windspeed      count
       0          0          0          0          0          0          0          0          0          0
```

2.3 Distribution of continuous variables

If a random variable is a continuous variable, its probability distribution is called a continuous probability distribution. A continuous probability distribution differs from a discrete probability distribution in several ways. The probability that a continuous random variable will assume a particular value is zero.

It can be observed from the below histograms is that temperature and feel temperature are normally distributed, whereas the variables wind speed and humidity are slightly skewed. The sleekness is likely because of the presence of outliers and extreme data in those variables.

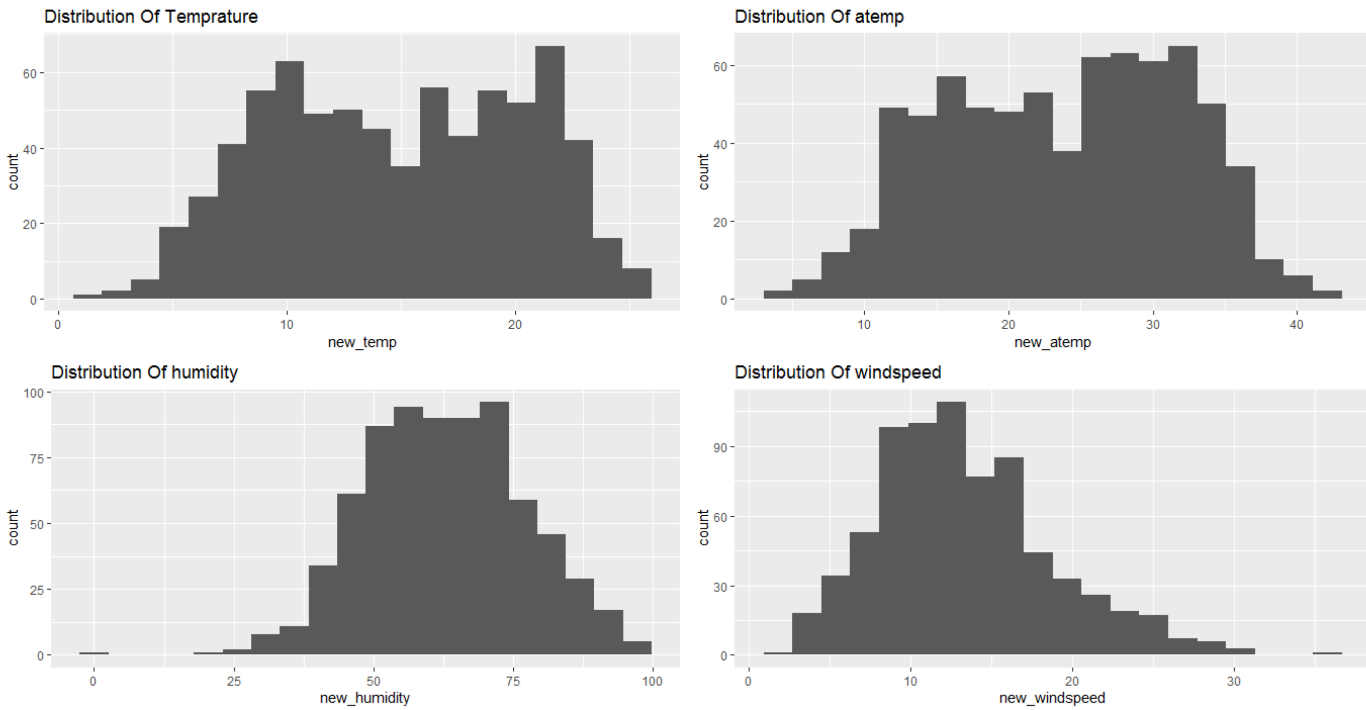


Fig: Distribution of continuous variables using Histograms

2.4 Distribution of categorical variables

Any variable that is not quantitative is categorical. Categorical variables take a value that is one of several possible categories. As naturally measured, categorical variables have no numerical meaning. Examples: Hair color, gender, field of study, college attended, political affiliation, status of disease infection.

The distribution of categorical variables is as shown in the below figure:

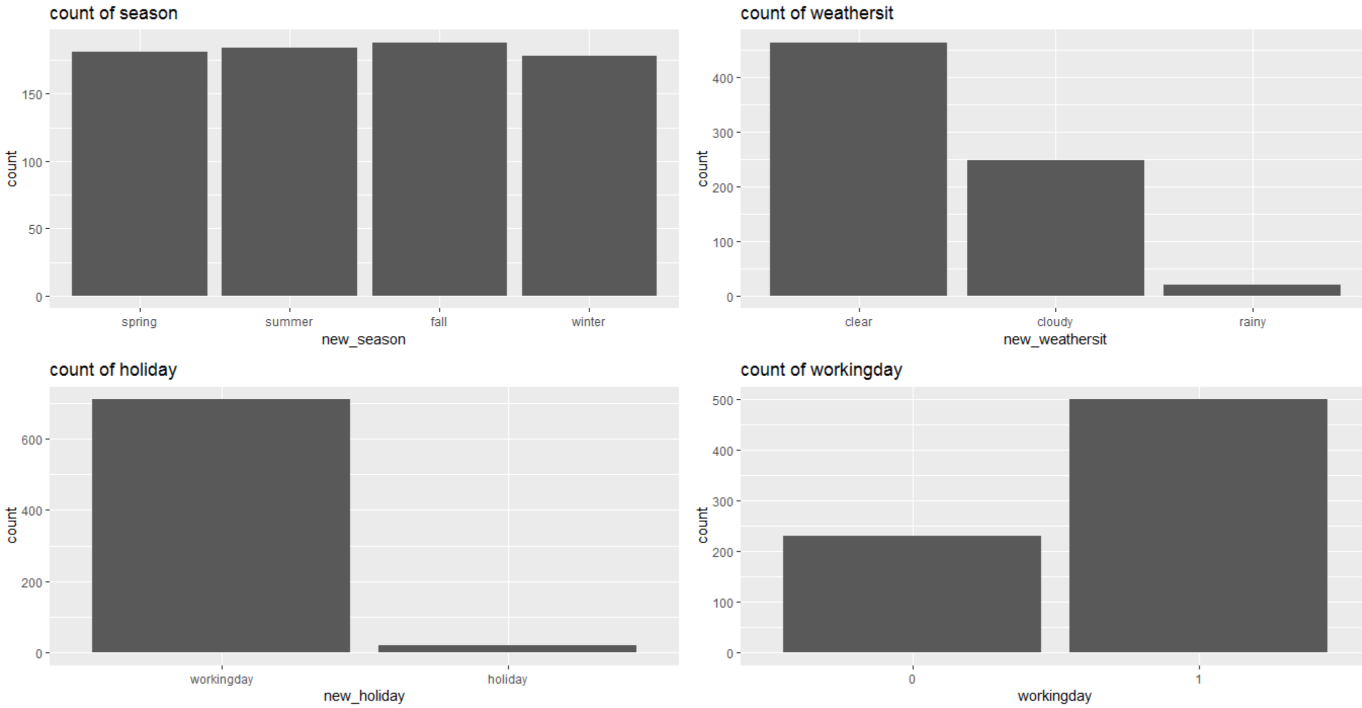


Fig: Distribution of categorical variables

2.5 Relationship of Continuous variables against bike count

The below figure shows the relationship between continuous variables and the target variable using scatter plot. It can be observed that there exists a linear positive relationship between the variables temperature and feel temperature with the bike rental count. There also exists a negative linear relationship between the variable's humidity and wind speed with the bike rental count.

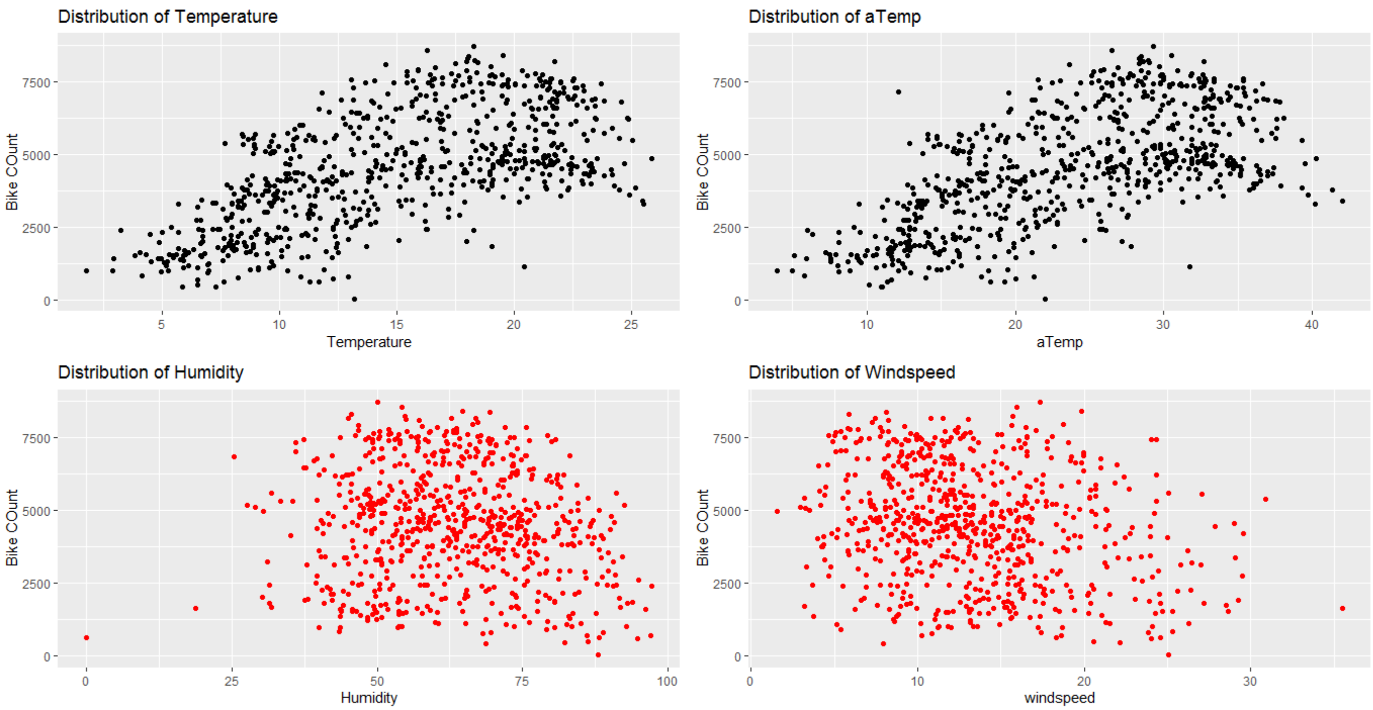


Fig: Scatter plot for continuous variables

2.5 Detection of outliers

The analysis of outlier data is referred to as outlier mining. Outliers may be detected using statistical tests that assume a distribution or probability model for the data, or using distance measures where objects that are a substantial distance from any other cluster are considered outliers.

Outliers are detected using box plots.

Below figure illustrates the box plots for all the continuous variables.

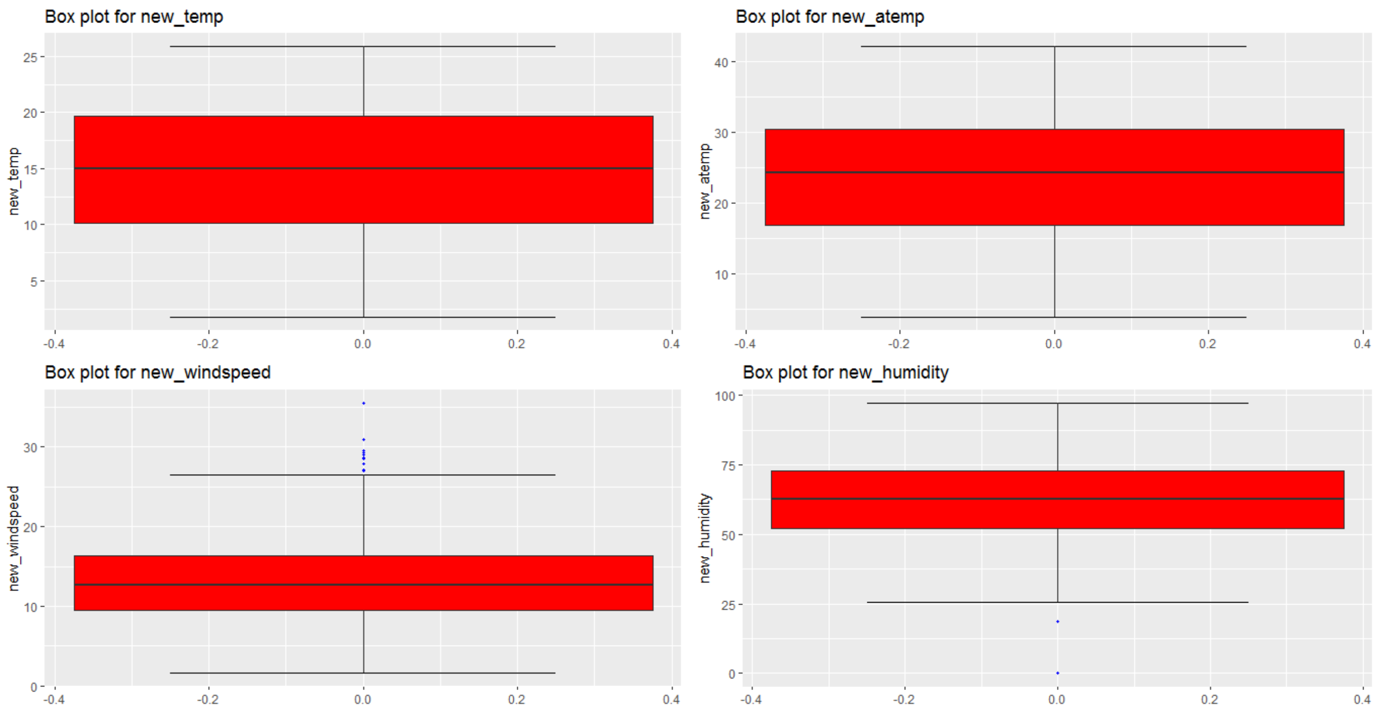


Fig: Box plot of continuous variables

In [186]:
```python
#Remove outliers in Humidity using the quatriles , find the q75 and q25
q75, q25 = np.percentile(df['real_hum'], [75 ,25])
print(q75,q25)
iqr = q75 - q25
print(iqr)
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)
```

```
73.02085 52.0
21.020849999999996
```

```python
#Remove outliers in Windspeed
q75, q25 = np.percentile (df['real_windspeed'], [75 ,25])
print(q75,q25)
iqr = q75 - q25
print(iqr)
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)

df = df.drop(df[df.iloc[:,18] < min].index)
df = df.drop(df[df.iloc[:,18] > max].index)
```

```
16.0957125 9.371110000000002
6.7246025
```

Fig: Removed outliers in Humidity and Wind speed

Outliers can be removed using the Box plot stats method, wherein the Inter Quartile Range (IQR) is calculated and the minimum and maximum values are calculated for the variables. Any value ranging outside the minimum and maximum value are discarded. The box plot of the continuous variables after removing the outliers is shown in the below figure:

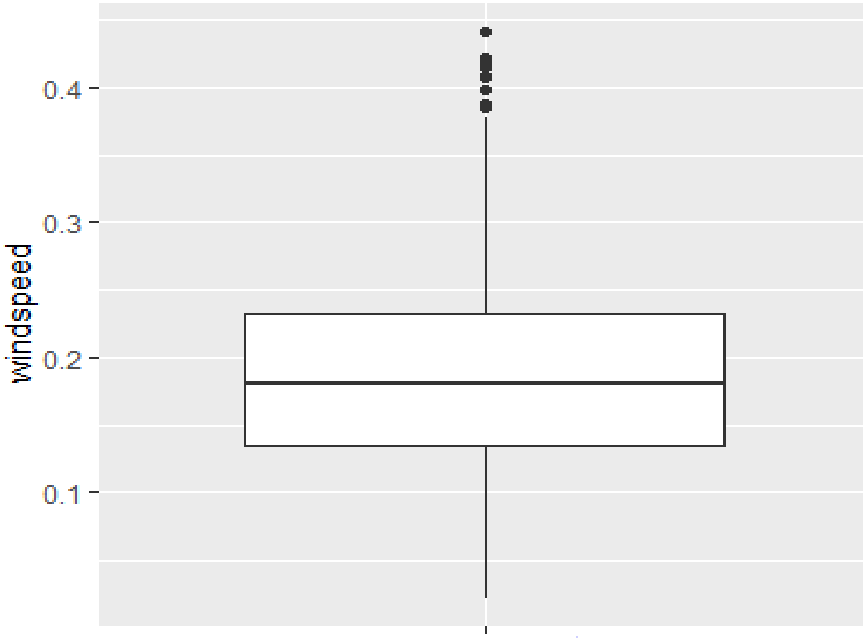


Fig: Box plot of continuous variables after removal of outlier

2.6 Feature Selections

Feature Selection is the process where you automatically or manually select those features which contribute most to your prediction variable or output in which you are interested in. Having irrelevant features in your data can decrease the accuracy of the models and make your model learn based on irrelevant features

Feature Selection reduces the complexity of a model and makes it easier to interpret. It also reduces over fitting. Features are selected based on their scores in various statistical tests for their correlation with the outcome variable. Correlation plot is used to find out if there is any multi co-linearity between variables. The highly collinear variables are dropped and then the model is executed.

Machine learning works on a simple rule – if you put garbage in, you will only get garbage to come out. By garbage here, this means noise in the data.

This becomes even more important when the number of features is very large. You need not use every feature at your disposal for creating an algorithm. You can assist your algorithm by feeding in only those features that are really important. We could see that the feature subsets giving better results than complete set of feature for the same algorithm.

We should consider the selection of feature for model based on below criteria

i. The relationship between two independent variable should be less and
ii. The relationship between Independent and Target variables should be high.

Below fig illustrates that relationship between all numeric variables using Corrgram plot.

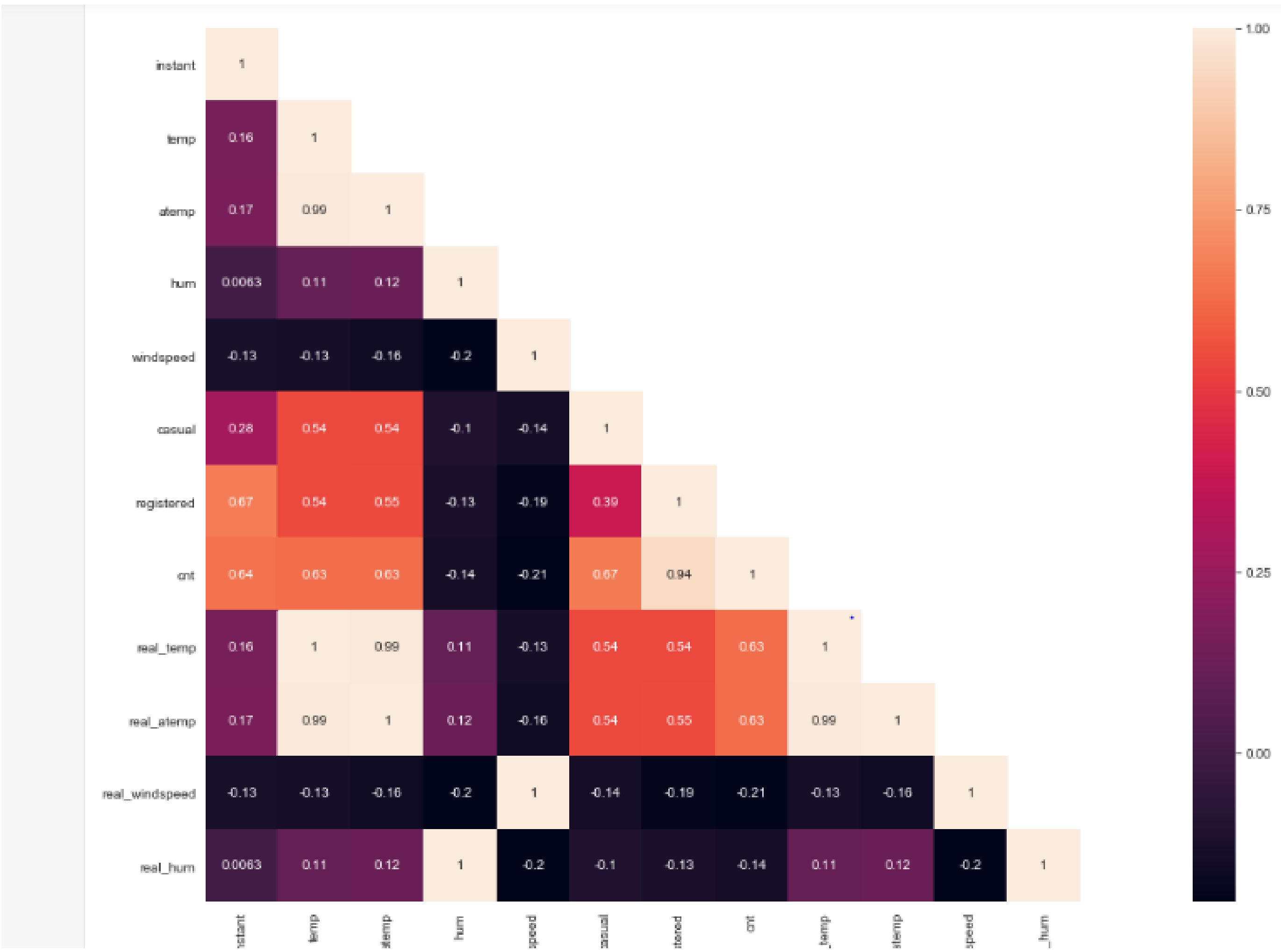


Fig: Correlation plot of all the variables

Chapter 3: Modeling

3.1 Model Selection

The dependent variable in our model is a continuous variable i.e., Count of bike rentals. Hence the models that we choose are Linear Regression, Decision Tree and Random Forest. The error metric chosen for the problem statement is Mean Absolute Error (MAE).

3.2 Decision Tree:

A decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

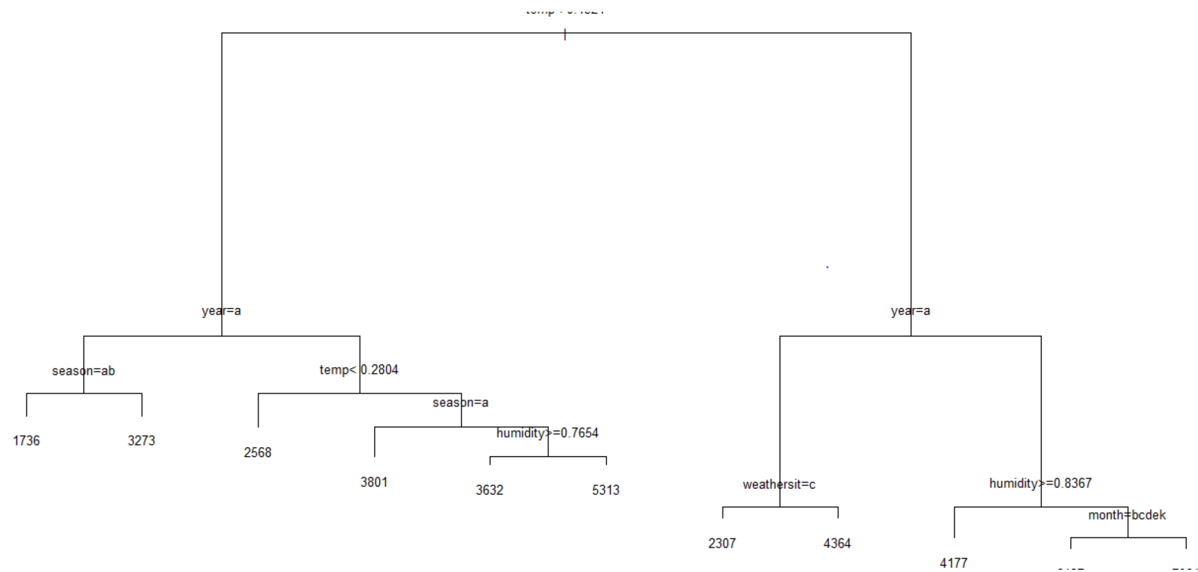
A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions.

Figure 3.3.1 Decision Tree Algorithm

```
In [214]: model_dectree

Out[214]: DecisionTreeRegressor(criterion='mse', max_depth=None, max_features=None,
                     max_leaf_nodes=None, min_impurity_decrease=0.0,
                     min_impurity_split=None, min_samples_leaf=1,
                     min_samples_split=2, min_weight_fraction_leaf=0.0,
                     presort=False, random_state=123, splitter='best')
```

Figure 3.3.2 Graphical Representation of Decision tree



3.2.1 Evaluation of Decision Tree Model

```
16    1204 1735.566
24    1416 1735.566
> #Calculation of MAPE
> regr.eval(trues = test[, 10], preds = prediction_dectree, stats = c('mae', 'mse', 'rmse', 'mape'))
         mae          mse         rmse         mape
6.232502e+02 6.619206e+05 8.135850e+02 1.891265e-01
> MAPE = function(real, pred){
+   print(mean(abs((real - pred)/real)) * 100)
+ }
> MAPE(test[,10], prediction_dectree)
[1] 18.91265
```

Figure 3.2.3 Evaluation of Decision Tree using MAPE and RMSE

Using decision tree, we can predict the value of bike count. MAE for this model is 623. The MAPE for this decision tree is 18.91265%. Hence the accuracy for this model is 81.08%.

3.3 Multiple Linear Regressions

Multiple linear regressions is the most common form of linear regression analysis. Multiple linear regressions is used to explain the relationship between one continuous dependent variable and two or more independent variables. The independent variables can be continuous or categorical.

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

Residuals:
    Min      1Q  Median      3Q     Max 
-4021.7  -341.0    68.9   488.0  2828.2 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  1482.98     271.22   5.468 6.90e-08 ***
season2       745.91     209.80   3.555 0.000410 ***
season3       742.89     242.42   3.064 0.002286 ** 
season4      1618.25     200.02   8.090 3.77e-15 ***
year1        2020.82      67.25  30.050  < 2e-16 ***
month2        196.85     165.06   1.193 0.233533    
month3        614.96     191.89   3.205 0.001429 ** 
month4        552.33     288.15   1.917 0.055779 .  
month5        902.65     316.12   2.855 0.004459 ** 
month6        660.16     329.95   2.001 0.045904 *  
month7        146.11     361.51   0.404 0.686251    
month8        478.40     348.07   1.374 0.169853    
month9       1027.68     303.21   3.389 0.000751 ***
month10       540.79     270.97   1.996 0.046452 *  
month11      -184.39     261.19  -0.706 0.480501    
month12      -147.15     199.02  -0.739 0.459999    
weekday1     -375.50     214.24  -1.753 0.080204 .  
weekday2     -340.55     237.38  -1.435 0.151965    
weekday3     -280.01     238.76  -1.173 0.241396    
weekday4     -236.05     235.79  -1.001 0.317219    
weekday5     -227.51     236.17  -0.963 0.335801    
weekday6      471.48     120.71   3.906 0.000105 ***
workingday1   656.82     205.58   3.195 0.001478 ** 
weathersit2  -448.72      90.32  -4.968 9.01e-07 ***
weathersit3 -2000.66     233.49  -8.569  < 2e-16 ***
```

Figure 3.3.2 Multiple Linear Regression Model

```
D:/edwisor/
> head(df)
   actual  predict prediction_linreg
5    1600 1735.566         1686.3964
6    1606 1735.566         1794.3854
8     959 1735.566          675.8553
9     822 1735.566          404.8430
16   1204 1735.566         1257.3836
24   1416 1735.566         1013.9835
> #calculate MAPE
> regr.eval(trues = test[, 10], preds = prediction_linreg, stats = c ('mae',
         mae          mse         rmse         mape
5.141614e+02 4.992291e+05 7.065615e+02 1.624673e-01
> MAPE(test[, 10], prediction_linreg)
[1] 16.24673
> 
```

Figure 3.3.3 Evaluation of Regression Model

As you can see the Adjusted R-squared value, we can explain 83.75% of the data using our multiple linear regression models. By looking at the F-statistic and combined p-value, we can Reject the null hypothesis that target variable does not depend on any of the predictor variables. This model explains the data very well and is considered to be good.

Even after removing the non-significant variables, hence the accuracy of this model is chosen to be final. Mean Absolute Error (MAE) is calculated and found to be 514. MAPE of this multiple linear regression model is 16.24%. Hence the accuracy of this model is 83.75%. This model performs very well for this test data.

- **VIF (Variance Inflation factor):** variance inflation factor (VIF) is the ratio of variance in a model with multiple terms, divided by the variance of a model with one term alone

It quantifies the multi collinearity between the independent variables.

As Linear regression will work well if multi collinearity between the Independent variables is less.

Figure 3.3.4 Multi co linearity between Independent variables

```
After excluding the collinear variables, the linear correlation coefficients ranges bet
ween:
min correlation ( humidity ~ instant ):  0.001543053
max correlation ( windspeed ~ humidity ):  -0.2330023

---------- VIFs of the remained variables --------
  Variables      VIF
1   instant 1.031053
2      temp 1.053554
3  humidity 1.067626
4 windspeed 1.084664
```

In the above figure it is showing there is strong correlation between all four Variables “instant”, “temp”, “humidity” and “wind speed” so, it is enough to consider Any one variable.

3.4 Random Forest:

Random forest, like its name implies, consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction

Random forest functions in below way:

i. Draws a bootstrap sample from training data.
ii. For each sample grow a decision tree and at each node of the tree
   a. Randomly draws a subset of mtry variable and p total of features that are available
   b. Picks the best variable and best split from the subset of mtry variable
   c. Continues until the tree is fully grown.

As we saw in Decision tree, there is a over fitting and its accuracy MAPE and RMSE is also poor in order to improve the performance of the model, we are developing model using Random Forest.

Figure 3.4.1 Random Forest Implementation

```
In [278]: model_random

Out[278]: RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='auto', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, n_estimators=500, n_jobs=None,
                      oob_score=False, random_state=123, verbose=0, warm_start=False)
```

Our Random Forest model is looking quite good where it utilized maximum variables to predict the count values.

Figure 3.4.2 Evaluation of Random Forest using MAPE and RMSE

```
> head(df)
   actual  predict prediction_linreg prediction_linreg prediction_RF
5    1600 1735.566         1686.3964         1686.3964      1736.563
6    1606 1735.566         1794.3854         1794.3854      1737.446
8     959 1735.566          675.8553          675.8553      1286.100
9     822 1735.566          404.8430          404.8430      1291.323
16   1204 1735.566         1257.3836         1257.3836      1443.149
24   1416 1735.566         1013.9835         1013.9835      1423.433
> #calculation of MAPE
> regr.eval(trues = test[, 10], preds = prediction_RF, stats = c ('mae', 'mse', 'rmse',
         mae          mse         rmse         mape
4.493859e+02 3.858152e+05 6.211402e+02 1.408359e-01
> MAPE(test[, 10], prediction_RF)
[1] 14.08359
> 
```

Fig shows Random Forest model performs dramatically better than Decision tree on both training and test data and well also improve the Accuracy (MAPE = 13.93596) and decrease the RMSE (612) of the model which is quite impressive.

Chapter 4: Conclusion

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

In our case of Bike rental prediction Data, Interpretability and Computation Efficiency, do not hold much significance. Therefore, we will use Predictive performance as the criteria to compare and evaluate models.

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

4.1 Mean Absolute Error (MAE):

Mean absolute error (MAE) is a measure of difference between two continuous variables.

MAE is one of the error measures used to calculate the predictive performance of the model. We will apply this measure to our models that we have generated in the previous section.

```
MAE <- function (actual, pred)
{
 print(mean (abs (actual - pred)))
 }
```

Linear Regression Model:
Accuracy = 83.75%
MAE = 514.16

Decision Tree:
Accuracy = 83.75%
MAE = 623.25.

Random Forest:

Accuracy = 86.07%
MAE = 445.34

Based on the above error metrics, Random Forest is the better model for our analysis. Hence Random Forest is chosen as the model for prediction of bike rental count.

Chapter 5: Appendix

5.1 Figures

Fig: Distribution of continuous variables using Histograms

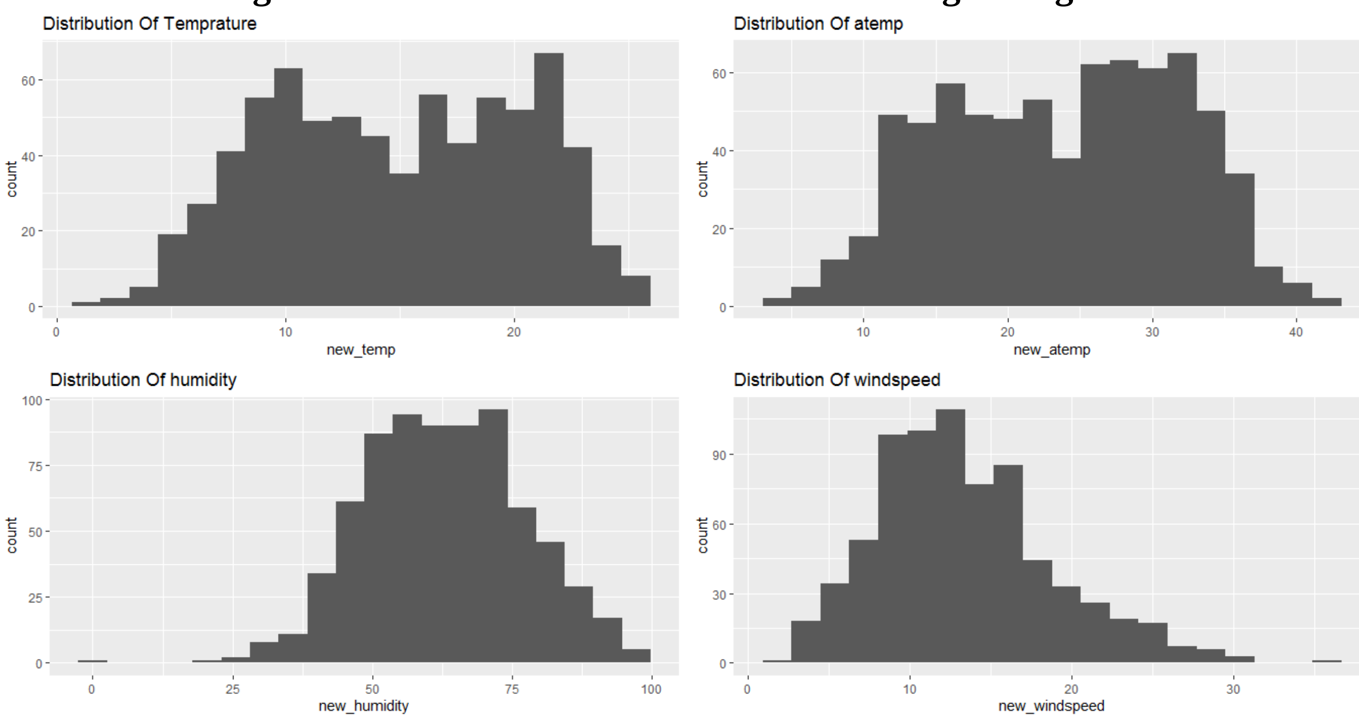


Fig: Bar graph of categorical Data using factor plot

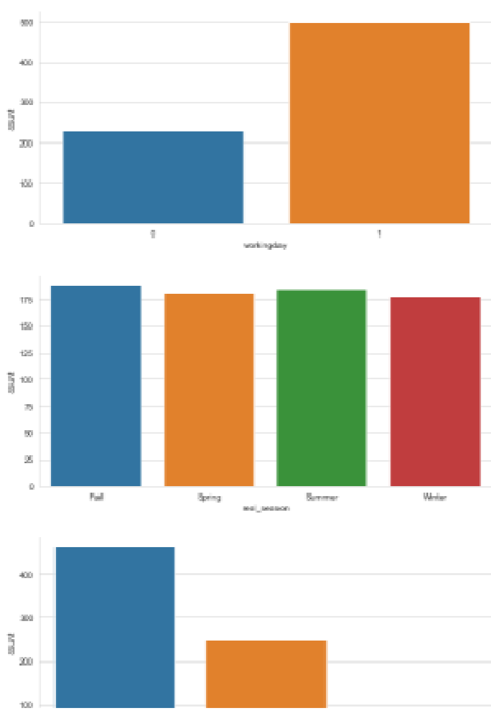


Fig: Distribution of categorical variables using bar plots

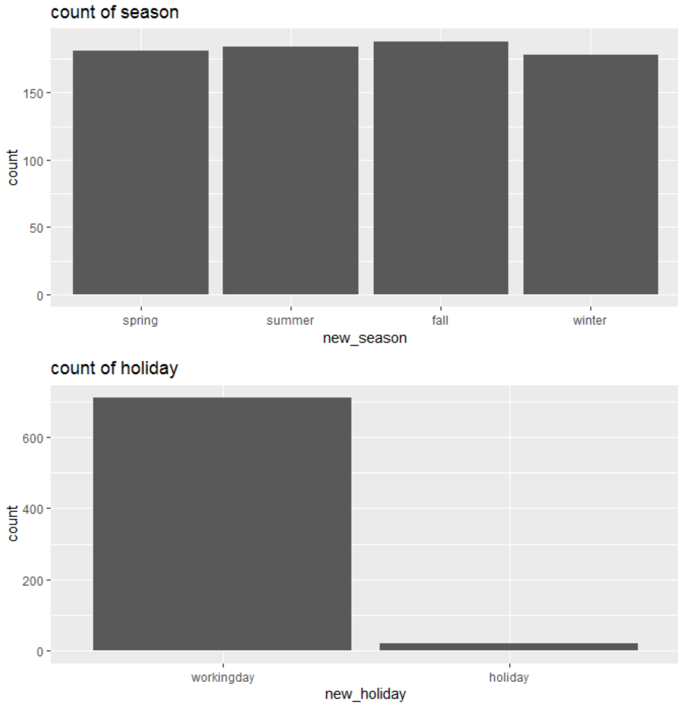


Fig: Scatter plot for continuous variables

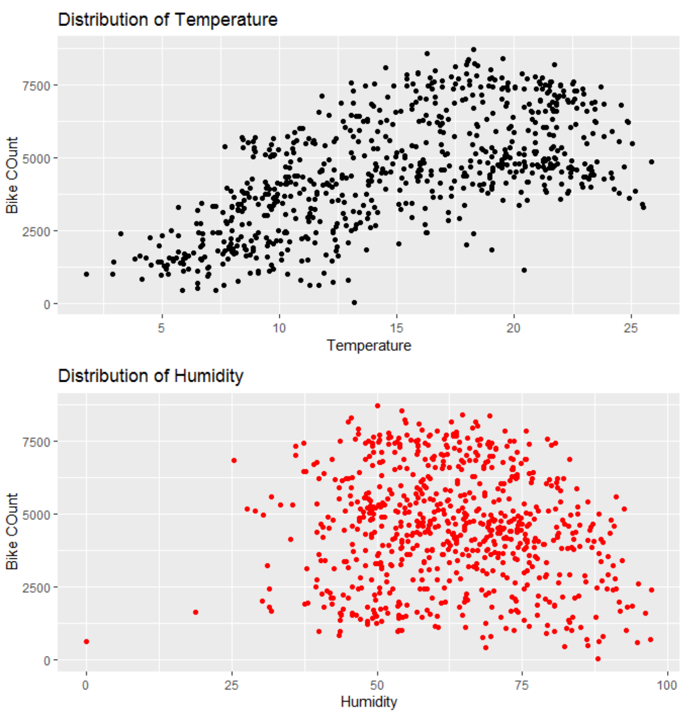


Fig: Box plot of continuous variables

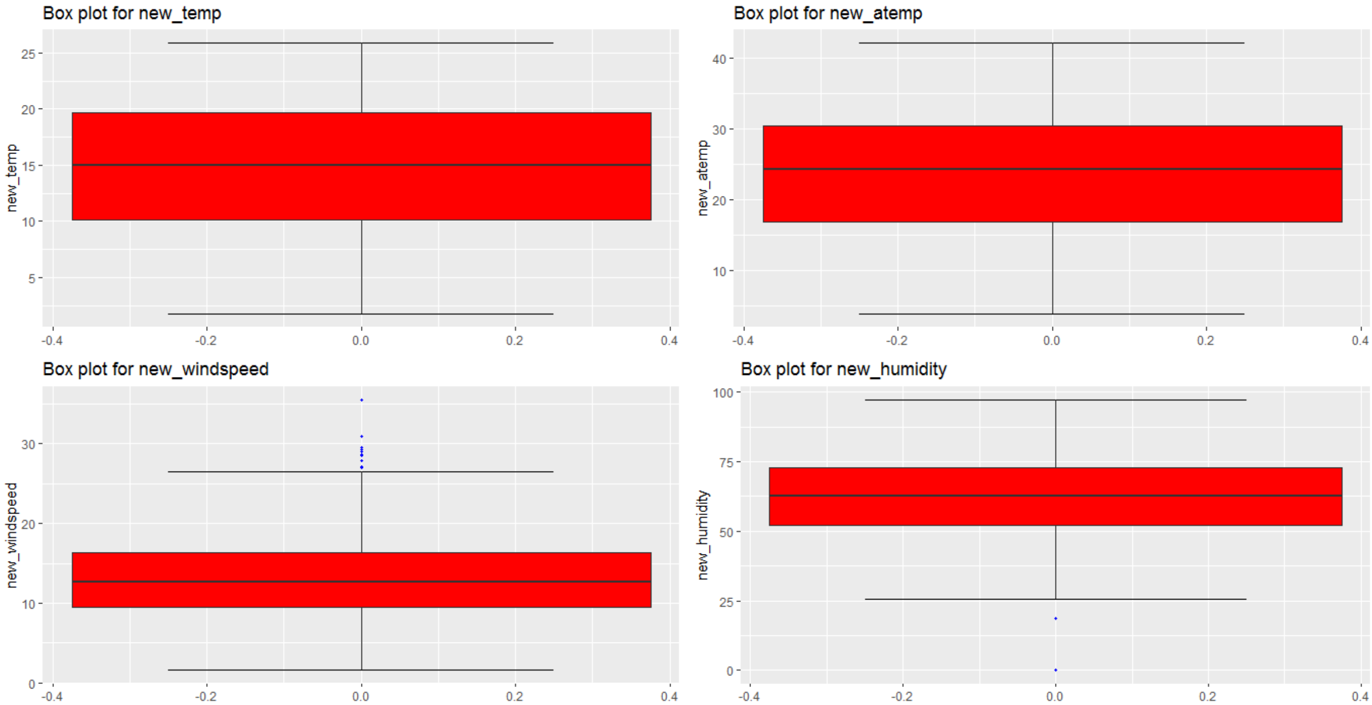
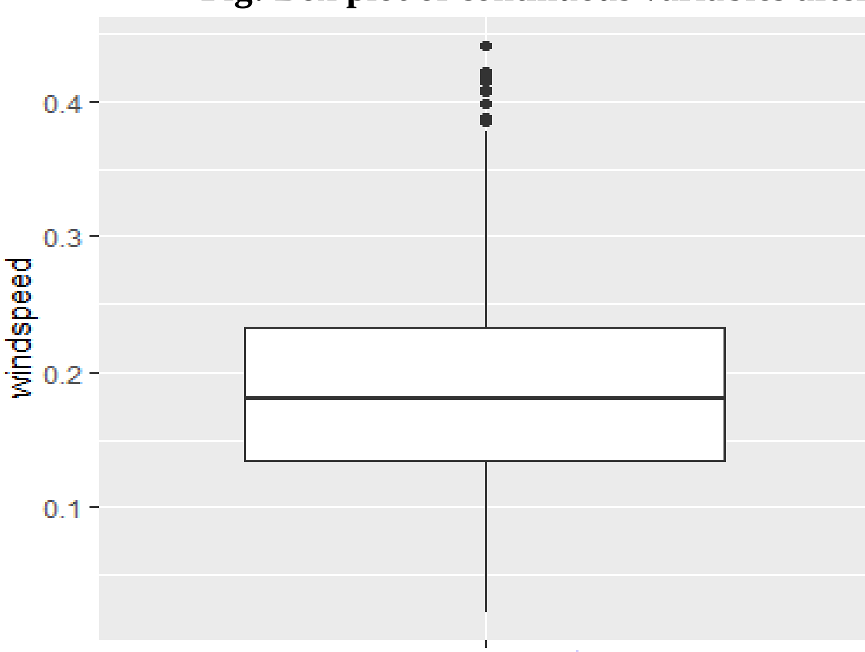


Fig: Box plot of continuous variables after removal of outliers



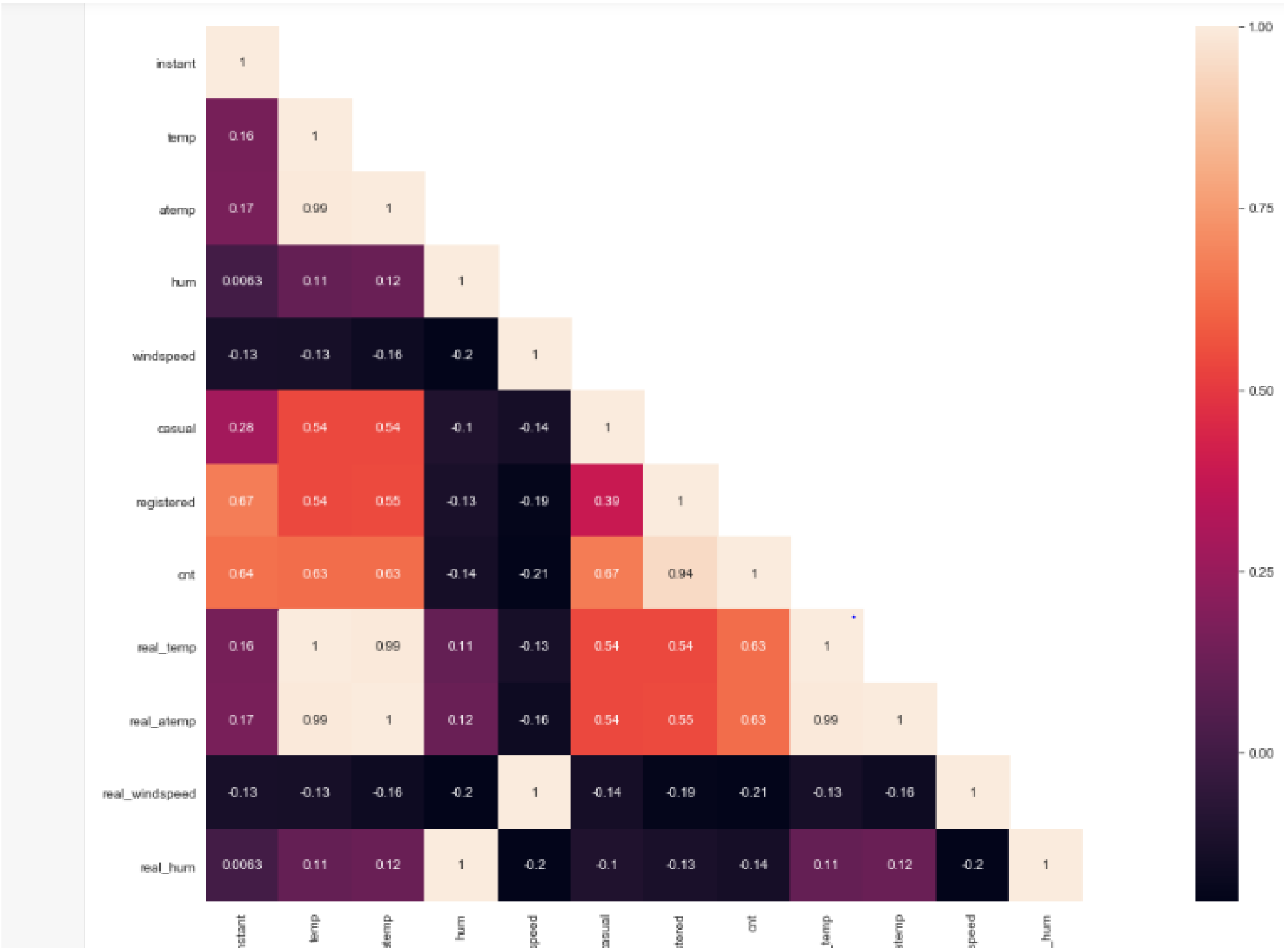


Fig: Correlation plot of all the variables

Fig: Distribution of numerical data (Humidity) using histogram

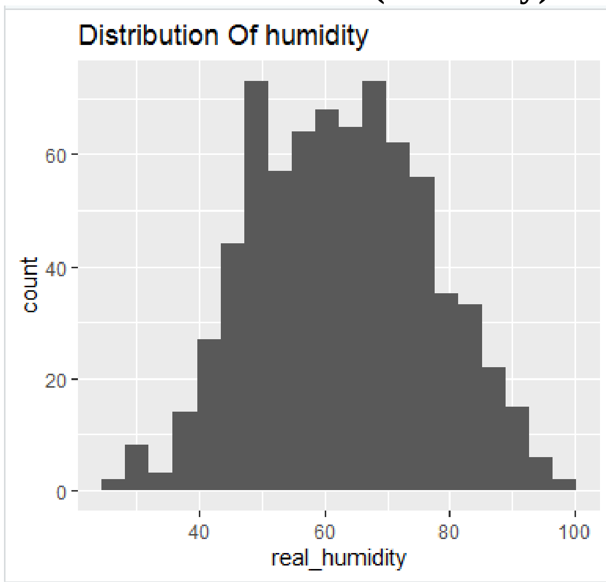


Fig: Distribution of numerical data (Temperature) using histogram

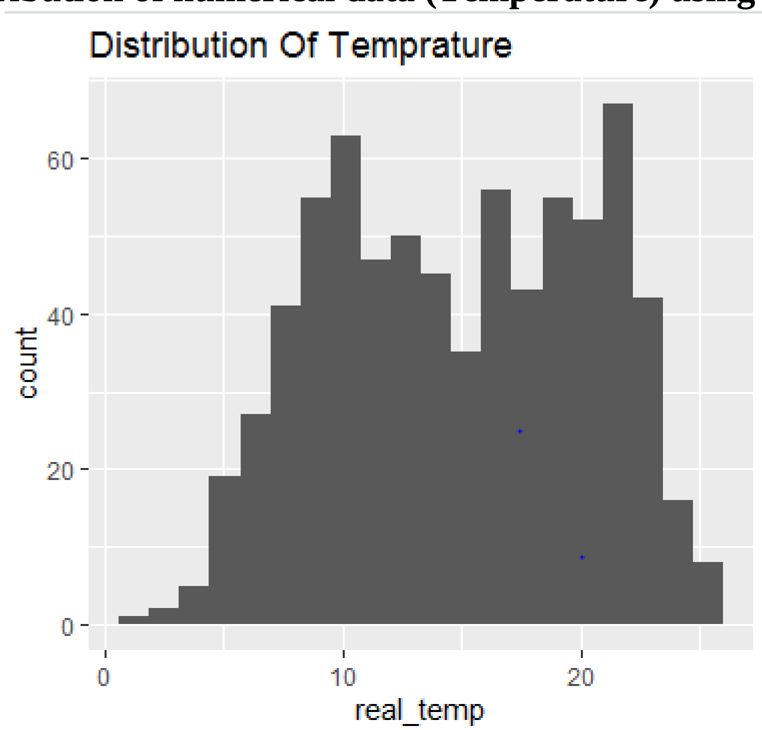


Fig: Outliers in data using box plot

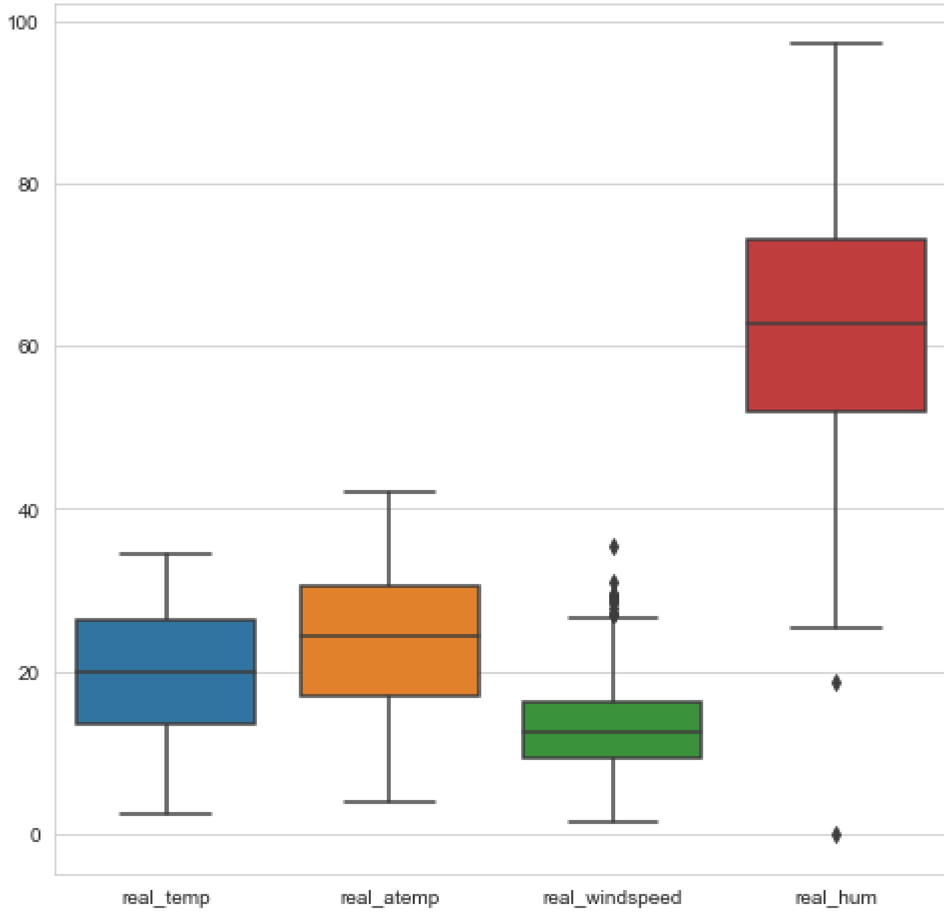


Fig: Distribution of Temperature and Humidity against Bike rental count using Scatter plot

```
Out[203]: Text(0, 0.5, 'Count of bikes')
```

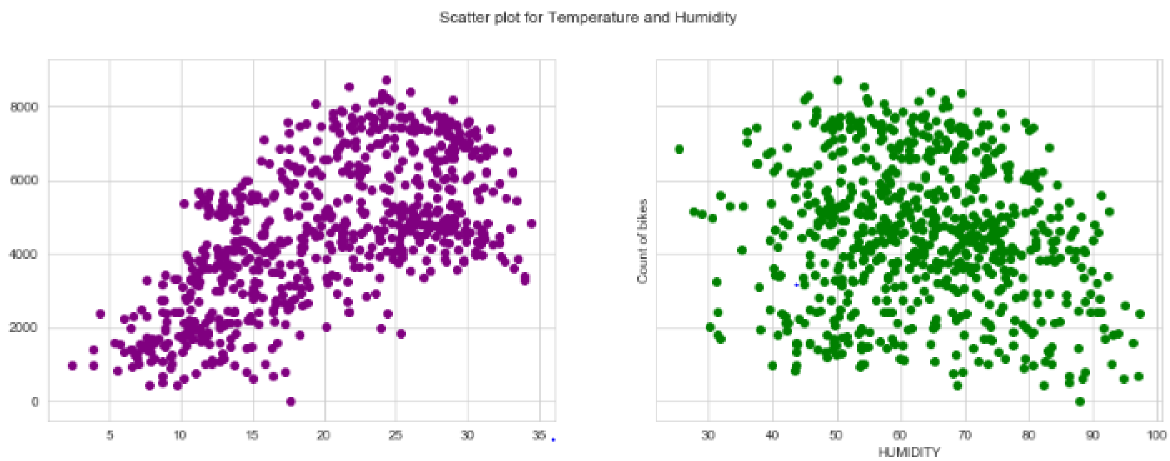
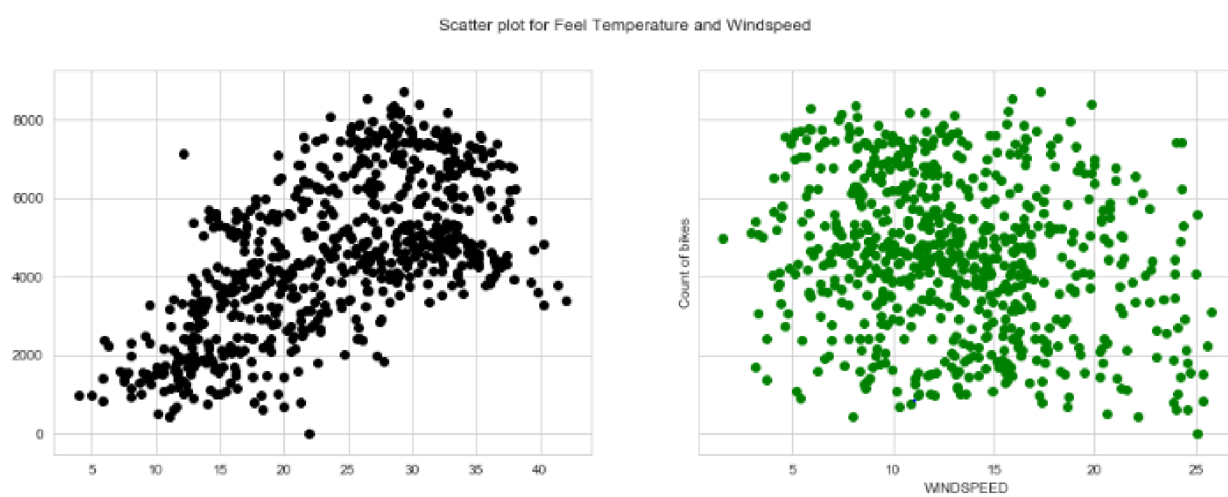


Fig: Distribution of Feel Temperature and Wind speed against Bike rental count using scatter plot

```
Out[205]: Text(0, 0.5, 'Count of bikes')
```



Chapter 6: R CODE

```
#Remove All The Stored Objects
rm(list = ls())

#Get Working Directory
setwd("D:/edwisor")

getwd()

#Set Current Working Directory
day <- read.csv("D:/edwisor/day.csv", header = TRUE)

##Install Required Packages and libraries
#Load Libraries
x = c ("plyr","dplyr", "ggplot2","rpart","dplyr","DMwR","randomForest","usdm",
    "corrgram","DataCombine")


lapply(x, require, character.only = TRUE)
rm(x)

##explore the data

#Structure Of Variables
str(day)

#Verify First Six Rows of Data
head(day)

#Column Names
names(day)

## count of number of rows and columns
nrow(day)
ncol(day)

#Target Variable Is 'cnt' And Other Variable Are Indepentent Variable
#verify Summary of Data
summary(day)
```

#rename the columns to give the proper meaning

```
names(day)[names(day)=="hum"] = "humidity"
names(day)[names(day)=="cnt"] = "count"
names(day)[names(day)=="yr"] = "year"
names(day)[names(day)=="mnth"] = "month"
```

#Check The Column Names

```
names(day)
```

#check the relationship between 'temp' and 'atemp' variable

```
ggplot(day, aes(x= temp,y=atemp)) +
  geom_point()+
  geom_smooth()
```

#This graph explians that there is strong relationship between 'temp' and 'atemp'

#lets Check the relationship between 'temp' and 'hum' variable

```
ggplot(day, aes(x= temp,y=humidity)) +
  geom_point()+
  geom_smooth()
```

 # Humidity is increasing till temparature is at point 0.7 and then decreasing gradually

#Check the relationship between 'temp' and 'windspeed' variable

```
ggplot(day, aes(x= temp,y=windspeed)) +
  geom_point()+
  geom_smooth()
```

 # it is showing that very less negative correlation between temp and windspeed

#FEATURE SELECTION

#create new columns with constant multiplied as the actual values have less range and its hard to analyse

```
day$real_temp = day$temp * 30
day$real_atemp = day$atemp * 50
day$real_windspeed = day$windspeed *70
day$real_humidity = day$humidity * 100
```

day$real_season = factor(x = day$season, levels = c (1,2,3,4), labels = c('spring', 'summer', 'fall', 'winter'))
day$real_year = factor(x = day$year, levels = c(0,1), labels = c ('2011', '2012'))
day$real_holiday = factor(x = day$holiday, levels = c (0,1), labels = c ('workingday','holiday'))
day$real_weathersit = factor(x = day$weathersit, levels = c (1,2,3,4), labels = c ('clear','cloudy', 'rainy', 'heavy rainy'))

#changing the data type of the variables to the required data types
day$weathersit = as.factor(day$weathersit)
day$month = as.factor(day$month)
day$season = as.factor(day$season)
day$dteday = as.factor(day$dteday)
day$workingday = as.factor(as.character(day$workingday))
day$weekday = as.factor(as.character(day$weekday))
day$holiday =as.factor(day$holiday)
day$year = as.factor(day$year)

#MISSING VALUE ANALYSIS :

missing_values = sapply(day,function(x){sum(is.na(x))})
missing_values
 # We could see that there are no missing fields in any of the variables.

#let us plot the graphs and explore the data and analyse the relationship btw variables

#Check the distribution of categorical Data using bar graph
bargraph_season = ggplot(data = day, aes(x = real_season)) + geom_bar() + ggtitle('count of season')
bargraph_weather = ggplot(data = day, aes(x = real_weathersit)) + geom_bar() + ggtitle('count of weathersit')
bargraph_holiday = ggplot(data = day, aes(x = real_holiday)) + geom_bar() + ggtitle('count of holiday')
bargraph_workday = ggplot(data = day, aes(x = workingday)) + geom_bar() + ggtitle('count of workingday')

#Ploting the all data together
gridExtra::grid.arrange(bargraph_season, bargraph_weather, bargraph_holiday, bargraph_workday, ncol=2)

#Check the distribution of numerical data using histogram
histogram_temp = ggplot(data = day, aes(x = real_temp)) + ggtitle('Distribution Of Temprature') + geom_histogram(bins = 20)
histogram_atemp = ggplot(data = day, aes(x = real_atemp)) + ggtitle('Distribution Of atemp') + geom_histogram(bins = 20)
histogram_hum = ggplot(data = day, aes(x = real_humidity)) + ggtitle('Distribution Of humidity') + geom_histogram(bins = 20)
histogram_wind = ggplot(data = day, aes(x = real_windspeed)) + ggtitle('Distribution Of windspeed') + geom_histogram(bins = 20)

#Ploting the all data together using histogram
gridExtra::grid.arrange(histogram
temp, histogram_atemp, histogram_hum, histogram_wind, ncol=2)

#Check the distribution of numerical data using scatterplot
scatter_temp = ggplot(data = day, aes(x = real_temp, y = count)) + ggtitle("Distribution of Temperature") + geom_point() + xlab("Temperature") + ylab("Bike COunt")

scatter_atemp = ggplot(data = day, aes(x = real_atemp, y = count)) + ggtitle("Distribution of aTemp") + geom_point() + xlab("aTemp") + ylab("Bike COunt")

scatter_hum = ggplot(data = day, aes(x = real_humidity, y = count)) + ggtitle("Distribution of Humidity") + geom_point(color ="red") + xlab("Humidity") + ylab("Bike COunt")

scatter_windspeed = ggplot(data = day, aes(x = real_windspeed, y = count)) + ggtitle("Distribution of Windspeed") + geom_point(color = "red") + xlab("windspeed") + ylab("Bike Count")
#Plotting the all data together using Scatterplot
gridExtra::grid.arrange(scatter_temp, scatter_atemp, scatter_hum, scatter_windspeed, ncol=2)
#Checking for OUTLIERS in data using boxplot
cnames = colnames(day [, c('real_temp', 'real_atemp', 'real_windspeed', 'real_humidity')])
for (i in 1:length(cnames))
 {
 assign(paste0('gn', i), ggplot(aes_string(y = cnames[i]),data = day) +
 stat_boxplot(geom = 'errorbar', width = 0.5) +
 geom_boxplot(outlier.colour = 'blue',fill = 'red', outlier.shape = 20, outlier.size = 1, notch =

```
FALSE) +
        theme(legend.position = 'bottom') +
        labs(y =cnames[i]) +
        ggtitle(paste("Box plot for",cnames[i])))
}
gridExtra::grid.arrange(gn1, gn2, gn3, gn4, ncol = 2)
```

#There is an outlier in windspeed

##Remove outliers in Windspeed

```
outlier_analysis = day [, 20][day[, 20] %in% boxplot.stats(day[, 20])$out]
day = day [ which(!day[, 20] %in% outlier_analysis),]
```

#Boxplot after removing outliers
#Boxplot for casual variable

```
ggplot(data = day, aes(x="", y= windspeed)) + geom_boxplot()
```

#Check for multicollinearity using VIF

```
df = day [, c( "instant","temp","atemp","humidity","windspeed" )]
vifcor(df)
```

#Check for collinearity using corelation graph

```
corrgram(day, order = F, upper.panel = panel.pie, text.panel = panel.txt, main =
'correlation plot')
```

#Removing the unwanted variables

```
day = subset(day, select = -
c(holiday,instant,dteday,atemp,casual,registered,real_temp,real_atemp,real_windspeed,
                    real_humidity,real_season,real_year,real_holiday,real_weathersit))
```

#remove all the objects except the data set (day) to build the model on top of it

```
rmExcept(keepers = 'day')
```

#Model development on the cleaned data set

#DECISION TREE
#Divide the data into train and test

```
set.seed(1234)
```

```
# train_index stores the index of the 80% of the data
train_index = sample(1:nrow(day), 0.8* nrow(day))
#train stores the 80% of the data
train = day[train_index,]
# test stores the remaining 20 % of the data
test = day[-train_index,]

#rpart for regression
model_dectree = rpart(count ~ . , data = train, method = 'anova')

#Predict for new test cases
prediction_dectree = predict(model_dectree, test[,-15])
print(model_dectree)

#Graphical Representation of Decision tree
par(cex= 0.8)
plot(model_dectree)
text(model_dectree)

#Prediction of the test cases
prediction_dectree = predict(model_dectree, test[, -10])

#Create dataframe for actual and predicted values
df = data.frame('actual' = test [,10], 'predict' = prediction_dectree)
head(df)

#Calculation of MAPE
regr.eval(trues = test[, 10], preds = prediction_dectree, stats = c('mae', 'mse', 'rmse',
'mape'))


MAPE = function(real, pred){
print(mean(abs((real - pred)/real)) * 100)
}
MAPE(test[,10], prediction_dectree)

#MAPE = 18.91265 %
#MAE = 623.25
#RMSE = 813.58
#ACCURACY = 81.08%
```

##LINEAR REGRESSION CLASSIFICATION

#Train the data using linear regression
model_linreg = lm (formula = count ~ . , data = train)

#Summary of the model
summary(model_linreg)

#Predict the test cases
prediction_linreg =predict(model_linreg, test[, -10])

#Create dataframe for actual and predicted values
df = cbind(df, prediction_linreg)
head(df)

#calculate MAPE
regr.eval(trues = test[, 10], preds = prediction_linreg, stats = c ('mae', 'mse', 'rmse', 'mape'))
MAPE(test[, 10], prediction_linreg)

#MAPE: 16.24673%
#RMSE: 706.56
#MAE: 514.16
#Accuracy: 83.75%
#Adjusted R squared: 0.8362
#F-statistic: 111.1
#Plot the graph real vs predicted values

plot(test$count, lty = 2, col = 'blue')
lines(prediction_linreg, col = 'black')

##RANDOM FOREST CLASSIFICATION
#Train the data using RANDOM FOREST

model_RF = randomForest(count ~ . , data = train, ntree = 500)

#predict the test cases
prediction_RF = predict(model_RF, test [, -10])

#create dataframe for real and predicted values
df = cbind(df, prediction_RF)
head(df)

#calculation of MAPE
regr.eval(trues = test[, 10], preds = prediction_RF, stats = c ('mae', 'mse', 'rmse', 'mape'))
MAPE(test[, 10], prediction_RF)

#MAPE = 13.93596 %
#MAE = 445.34
#RMSE = 612.54
#ACCURACY = 86.07%

#Random Forest is 86.07% accurate andd hence chosen as the model for prediction of bike rental count.
#and RMSE Is 6.125429e+02
--

#Random Forest accuracy is 86.07% hence chosen as the model for prediction of bike rental count and RMSE is 6.125429e+02

Chapter 7: Python Code

```
pip install seaborn
#set working directory
os.chdir("D:/edwisor")
print(os.getcwd())
#read the csv file from the working directory
day = pd.read_csv("day.csv", sep=",")
day
#we have 731 records and 16 variables
day.shape
#print first few records
print(day.head())
#check the data type of the variables
print(day.dtypes)
#Create a new dataframe containing required columns and creating new columns
df = day.copy()
df.head()

#UNIVARIATE ANALYSIS
# here the target variable is Cnt
#descriptive statistics summary
day['cnt'].describe()
#Check whether target variable is normal or not
sns.distplot(day['cnt'])

#Create new columns with some constants multiplying as the values are very small and distinct to analyse
df['real_temp'] = day['temp'] * 40
df['real_atemp'] = day['atemp'] * 50
df['real_windspeed'] = day['windspeed'] * 70
df['real_hum'] = day['hum'] * 100

# replacing the values with actual values and recreating the variables
df['real_season'] = day['season'].replace([1,2,3,4],["Summer","Spring","Fall","Winter"])
df['real_yr'] = day['yr'].replace([0,1],["2011","2012"])
df['real_holiday'] = day['holiday'].replace([0,1],["Working day","Holiday"])
df['real_weathersit'] = day['weathersit'].replace([1,2,3,4],["Clear","Cloudy","Rain","Heavy Rain"])
```

#Check the data types of the variables
```
df.dtypes
```

#Change the data types t the required data type for further analysis
```
df['weathersit'] = df['weathersit'].astype('category')
df['holiday'] = df['holiday'].astype('category')
df['weekday'] = df['weekday'].astype('category')
df['mnth'] = df['mnth'].astype('category')
df['yr'] = df['yr'].astype('category')
df['season'] = df['season'].astype('category')
df['workingday'] = df['workingday'].astype('category')
df['real_season'] = df['real_season'].astype('category')
df['real_yr'] = df['real_yr'].astype('category')
df['real_holiday'] = df['real_holiday'].astype('category')
df['real_weathersit'] = df['real_weathersit'].astype('category')
```

 # recheck the data types of variables again after changing
```
df.dtypes
```

#Check the count of values of categorical variables in the data set
```
print(df.real_yr.value_counts())
print(df.real_holiday.value_counts())
print(df.real_weathersit.value_counts())
print(df.mnth.value_counts())
print(df.workingday.value_counts())
print(df.weekday.value_counts())
```

#Check for the missing values in the data set
```
df.isnull().sum()
```

 # we could see that there are no missing values in any of the variables
#Check the bar graph of categorical Data using factorplot
```
sns.set_style("whitegrid")
sns.factorplot(data=df, x='real_weathersit', kind= 'count',size=4,aspect=2)
sns.factorplot(data=df, x='real_season', kind= 'count',size=4,aspect=2)
sns.factorplot(data=df, x='workingday', kind= 'count',height=4,aspect=2)
```

#Check the distribution of numerical data using histogram
```
plt.hist(data=df, x='real_temp', bins='auto', label='Temperature')
plt.xlabel('Temperature in Celcius')
plt.title("Temperature Distribution")
```

```
#Checking  the distribution of numerical data using histogram plots
plt.hist(data=df, x='real_hum', bins='auto', label='Humidity')
plt.xlabel('Humidity')
plt.title("Humidity Distribution")

#Check for outliers in data using boxplot analysis
sns.boxplot(data=df[['real_temp','real_atemp','real_windspeed','real_hum']])
fig=plt.gcf()
fig.set_size_inches(8,8)

#Remove outliers in Humidity using the quatriles , find the q75 and q25 and remove values above max and below min respectively
q75, q25 = np.percentile(df['real_hum'], [75 ,25])
print(q75,q25)
iqr = q75 - q25
print(iqr)
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)

# remove the values from  humidity which are above max value and below min value
df = df.drop(df[df.iloc[:,19] < min].index)
df = df.drop(df[df.iloc[:,19] > max].index)

print(min)
print(max)
df.head()

#Remove outliers in Windspeed
q75, q25 = np.percentile (df['real_windspeed'], [75 ,25])
print(q75,q25)
iqr = q75 - q25
print(iqr)
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)

df = df.drop(df[df.iloc[:,18] < min].index)
df = df.drop(df[df.iloc[:,18] > max].index)

print(min)
print(max)

df.head()
```

#Check for collinearity using corelation matrix.

```
cor_mat= df[:].corr()
mask = np.array(cor_mat)
mask[np.tril_indices_from(mask)] = False
fig=plt.gcf()
fig.set_size_inches(30,12)
sns.heatmap(data=cor_mat,mask=mask,square=True,annot=True,cbar=True)

cor_mat= df[:].corr()
cor_mat
```

#Check the distribution of Temperature and Humdity against count using scatter plot

```
fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)
axs[0].scatter(data=df, x='real_temp', y='cnt', color = 'purple')
axs[1].scatter(data=df, x='real_hum', y='cnt', color = 'green')
fig.suptitle('Scatter plot for Temperature and Humidity')
plt.xlabel("HUMIDITY")
plt.ylabel("Count of bikes")
```

#Check the distribution of Feel Temperature and Windspeed against Bike rental count using scatter plot

```
fig, axs = plt.subplots(1,2, figsize=(15, 5), sharey=True)
axs[0].scatter(data=df, x='real_atemp', y='cnt', color = 'black')
axs[1].scatter(data=df, x='real_windspeed', y='cnt', color = 'green')
fig.suptitle('Scatter plot for Feel Temperature and Windspeed')
plt.xlabel("WINDSPEED")
plt.ylabel("Count of bikes")

df=df.drop(columns=['holiday','instant','dteday','atemp','casual','registered','real_temp','real_
atemp', 'real_windspeed','real_hum','real_season','real_yr','real_holiday','real_weathersit'])

Print(Df)
```

 # Above is the data set to build a model on top of it
#DECISION TREE CLASSIFICATION

#Import Libraries for decision tree

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
```

#Divide data into train and test , train as 80% and 20% will be the test data

```
train,test = train_test_split(df, test_size = 0.2, random_state = 123)
```

```
#Train the model
model_dectree = DecisionTreeRegressor(random_state=123).fit(train.iloc[:,0:9],
train.iloc[:,9])
model_dectree

#Predict the results of test data
prediction_dectree = model_dectree.predict(test.iloc[:,0:9])

# predicted values of the test data
prediction_dectree

# creating a new data frame with actual and predicted values of the test data
df_dt = pd.DataFrame({'actual': test.iloc[:,9], 'pred': prediction_dectree})

df_dt.head()

#Function for Mean Absolute Percentage Error
def MAPE(y_real,y_pred):
    mape = np.mean(np.abs((y_real - y_pred)/y_real))
    return mape

#Calculate MAPE for decision tree model
MAPE(test.iloc[:,9],prediction_dectree)
#MAPE: 16.98%
#Accuracy: 83.02%

# as the accuracy is less comparatively , let us build model using another algorithm
#LINEAR REGRESSION CLASSIFICATION
#import libraries for Linear regression
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error

#Train the model
model_linreg = sm.OLS(train.iloc[:,9].astype(float), train.iloc[:,0:9].astype(float)).fit()

#Check the summary of model
model_linreg
model_lr.summary()


#Predict the results of test data
predictions_linreg = model_linreg.predict(test.iloc[:,0:9])
```

```
predictions_linreg

#Create a dataframe for actual values and predicted values to compare both
df_lr = pd.DataFrame({'real': test.iloc[:,9], 'pred': predictions_linreg})
df_lr.head()



#Calclulate MAPE Accuracy_LR
MAPE(test.iloc[:,9],predictions_linreg)
#MAPE:17.58%

Accuracy_LR = (1-0.17583)*100
print(Accuracy_LR)
#ACCURACY: 82.42%

#Create continuous data. Save target variable first
train_linreg = train[['cnt','temp','hum','windspeed']]
test_linreg = test[['cnt','temp','hum','windspeed']]

#Create dummies for categorical variables
cat_names = ["mnth", "yr", "season", "weekday", "workingday", "weathersit"]

for i in cat_names:
    temp1 = pd.get_dummies(train[i], prefix = i)
    temp2 = pd.get_dummies(test[i], prefix = i)

#joing the train_linreg data with the temp1 and test_linreg with temp2
    train_linreg = train_linreg.join(temp1)
    test_linreg = test_linreg.join(temp2)

    cat_names
train_linreg.head()

test_linreg.head()
#Train the model
model_linreg = sm.OLS(train_linreg.iloc[:,0].astype(float),
train_linreg.iloc[:,1:34].astype(float)).fit()


#summary of model
model_linreg.summary()
```

#Predict the results of test data
predictions_linreg = model_linreg.predict(test_linreg.iloc[:,1:34])

predictions_linreg

#Create a dataframe for actual values and predicted values to compare and evaluate
df_lr = pd.DataFrame({'actual': test_linreg.iloc[:,0], 'pred': predictions_linreg})
df_lr.head()

#CALCULATE MAPE
MAPE(test_lr.iloc[:,0],predictions_linreg)

#MAPE:18.5%

Accuracy_Linreg = (1-0.185)*100
print(Accuracy_Linreg)

#ACCURACY: 81.50%

#RANDOM FOREST CLASSIFICATION

#Import library for RandomForestRegressor
from sklearn.ensemble import RandomForestRegressor

#Train the model
model_random =
RandomForestRegressor(n_estimators=500,random_state=123).fit(train.iloc[:,0:9],
train.iloc[:,9])
model_random

#Predict the results of test data
predictions_random = model_random.predict(test.iloc[:,0:9])
predictions_random

#Create a dataframe for actual values and predicted values to compare
df_rf = pd.DataFrame({'actual': test.iloc[:,9], 'pred': predictions_random})
df_rf.head()

#CALCULATE MAPE
MAPE(test.iloc[:,9],predictions_random)

#MAPE: 13.10%
#ACCURACY:86.90%

 # hence , we select random forest as the best algorithm which fits in this case as the accuracy is 86.90%

--

Chapter 8: References

Towardsdatascience.com, quora , edwisor.com ,stack overflow and wikipedia.org.