Predicting Bike rental count

Abhay Sharma

15 Aug 2018

Table of Contents

1 Introduction 3

1.1 Problem Statement 3

1.2 Data Understanding 3

2 Methodology 5

2.1 Data Preprocessing 5

2.1.1 Missing Value Analysis 5

2.1.2 Outlier Analysis 5

2.2 Data Visualization 9

2.3 Feature Selection 15

3 Modelling 17

3.1 Linear Regression 17

3.2 Decision Tree Regression 20

4 Conclusion 21

4.1 Model Evaluation 21

4.2 Model Selection 21

5 Codes 22

5.1 R Code 22

5.2 Python Code 26

1 Introduction

1.1 Problem Statement

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. Here we have to predict the count of bike rental on daily basis, we have other variables to predict the bike renting on daily basis and it is a regression task there can be many factors which will be used for this i.e., environment, weather and season.

1.2 Data Understanding

Our tasks is to build regression model which will predict further target variable values which is Bike Count. Given below are the some.

Table 1.1 (Bike Renting Data)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	1/1/2011	1	0	1	0	6	0
2	1/2/2011	1	0	1	0	0	0
3	1/3/2011	1	0	1	0	1	1
4	1/4/2011	1	0	1	0	2	1

Table 2.2 (Bike Renting Data)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562

Lets see what variables tells:-
instant: Record index
dteday: Date
season: Season (1:spring, 2:summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted from Holiday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit: (extracted from Freemeteo)
1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min),
t_min=-8, t_max=+39 (only in hourly scale)
atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_
min), t_min=-16, t_max=+50 (only in hourly scale)
hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)
casual: count of casual users
registered: count of registered users
cnt: count of total rental bikes including both casual and registered
From these data we found out that out of 16 variables there is one target variable 'cnt'. Rest all are predictor variables except some variables which may be drop in further data-preprocessing.

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

2 Methodology

2.1 Data Preprocessing

Data in real world is dirty it of no use until unless we apply data preprocessing on it. In other words, Pre- processing refers to the transformations applied to your data before feeding it to the algorithm. It's a data mining technique which that involves transforming raw data into an understandable format or we can say that it prepares raw data to further processing. There are so many things that we do in data preprocessing like data cleaning, data integration, data transformation, or data reduction.

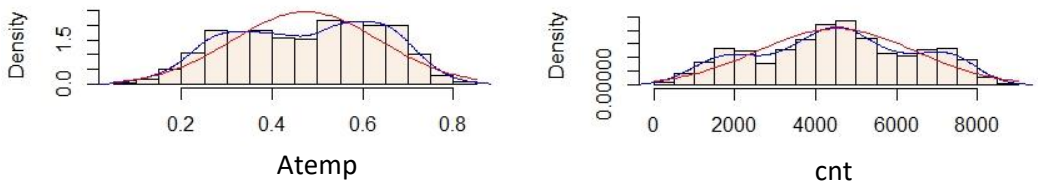
2.1.1 Missing Value Analysis

Missing Values Analysis is use to fill NULL values in data with some imputation techniques But here in our Bike Renting Data we don't have any null Value. By the way Our data doesn't contain missing value. Our Data is fit.

```
> # Missing Value Analysis
> sum(is.na(bike_data))
[1] 0
```

2.1.2 Outlier Analysis

Outlier are some values which cause deviation in central tendencies or they usually show skewness in histogram or outlier value in boxplot. We cannot see anything in figure 2.1 Before Drawing Histogram we have to break 'dteday' column to new 'date' column. Then we will also plot histogram for date also to if there any examine any extra value.



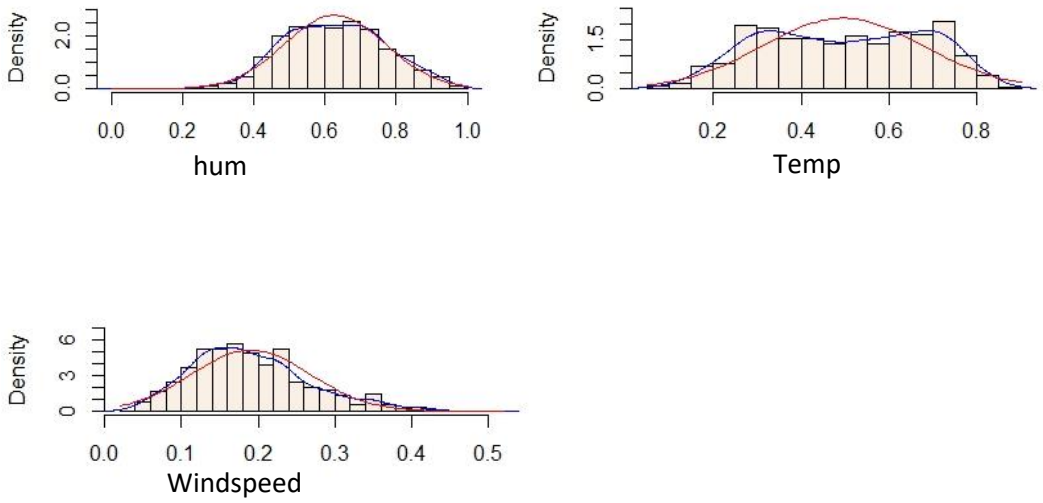


Figure 1: Probability Density function on Bike Renting Data

```
multi.hist(bike_data$temp, main = NA, dcol = c("blue", "red"),
+            dlty = c("solid", "solid"), bcol = "linen")
>
> multi.hist(bike_data$atemp, main = NA, dcol = c("blue", "red"),
+            dlty = c("solid", "solid"), bcol = "linen")
>
> multi.hist(bike_data$hum, main = NA, dcol = c("blue", "red"),  # Outlier
+            dlty = c("solid", "solid"), bcol = "linen")
>
> multi.hist(bike_data$windspeed, main = NA, dcol = c("blue", "red"),   # Out
lier
+            dlty = c("solid", "solid"), bcol = "linen")
>
> multi.hist(bike_data$cnt, main = NA, dcol = c("blue", "red"),
+            dlty = c("solid", "solid"), bcol = "linen")
```

We can see that there is some skewness in *hum* and *windspeed* histogram, Let's build their Boxplot.

We can see that there are some outliers in Data of *hum* and *windspeed* .We will remove those values as it is normalized data and for better accuracy.

Code:

```
> numeric_index = sapply(bike_data,is.numeric)
> numeric_data = bike_data[,numeric_index]
> cnames = colnames(numeric_data)
>
> for (i in 1:length(cnames))
+ {
+   assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), dat
a = subset(bike_data))+
+            stat_boxplot(geom = "errorbar", width = 0.5) +
+            geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=
18,
+                         outlier.size=1, notch=FALSE) +
+            theme(legend.position="bottom")+
+            labs(y=cnames[i],x="cnt")+
+            ggtitle(paste("Box plot of cnt for",cnames[i])))
+ }
> #gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
> #gridExtra::grid.arrange(gn13,gn14,gn16,ncol=3)
> gridExtra::grid.arrange(gn11,gn12,ncol=2)
```

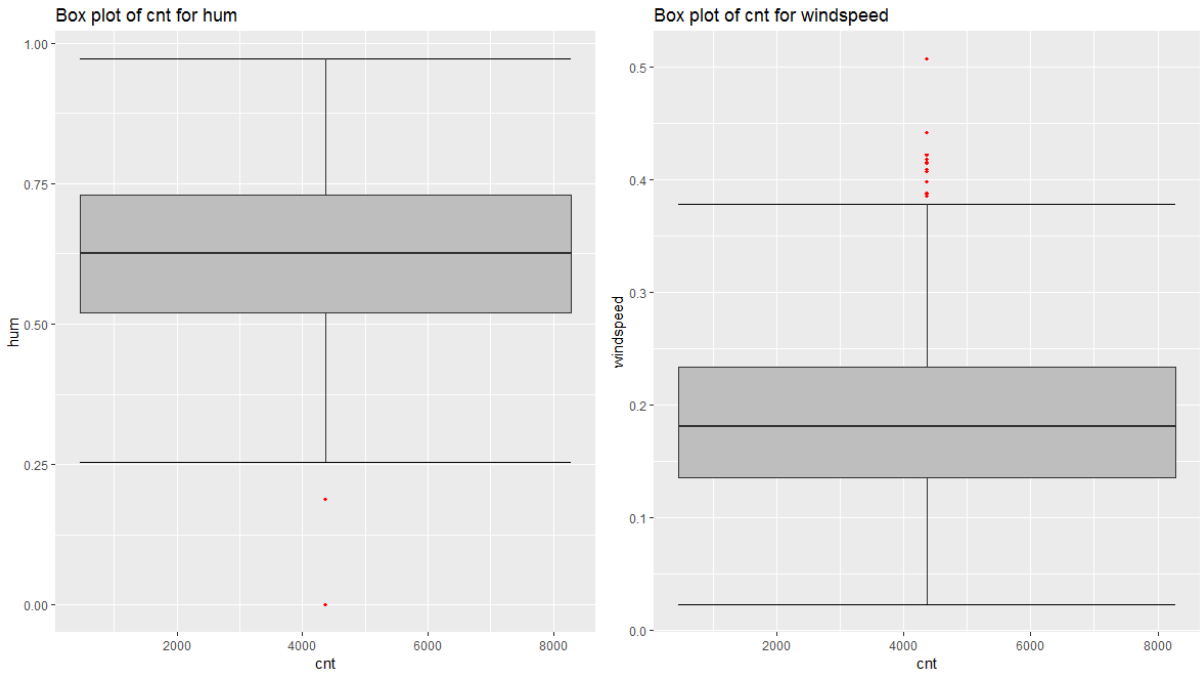


Figure 2: Box Plot of hum and windspeed (See R Code in Appendix)

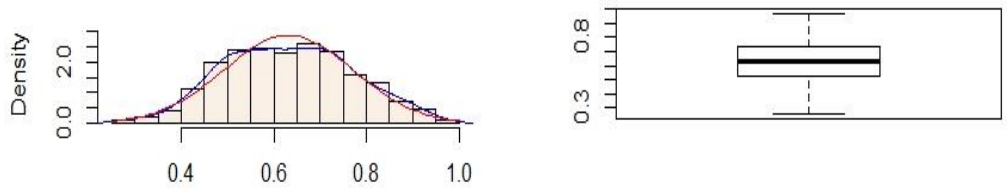


Figure 3: *hum* after Outlier Analysis

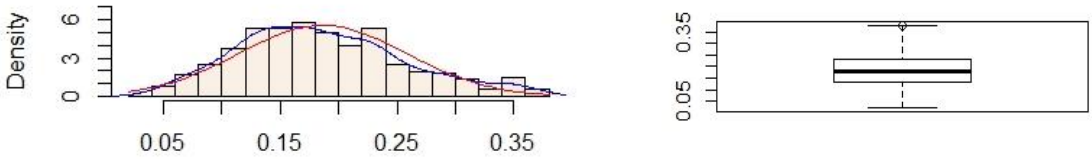


Figure 4: *windspeed* after outlier analysis

We used *box.stats* method to identify and replace outliers in *hum* and *windspeed* with mean. Also its normalized data there is not too much to do with outliers.

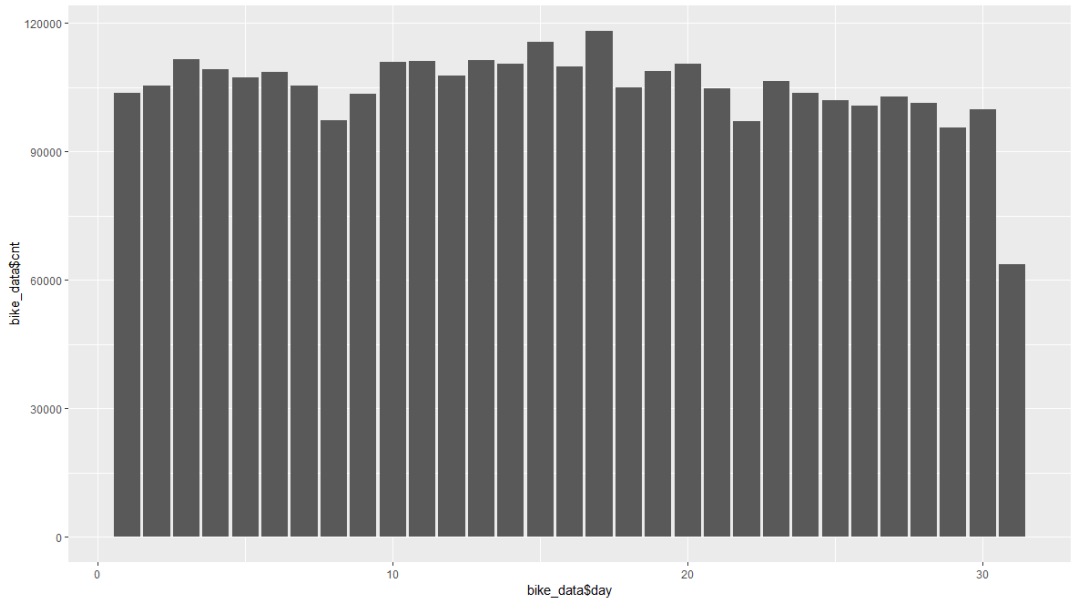
```
> outlier_val <- c('hum' , 'windspeed')
> #
> for(i in outlier_val){
+   val = bike_data[,i][bike_data[,i] %in% boxplot.stats(bike_data[,i])$out]
+   #print(length(val))
+   bike_data[,i][bike_data[,i] %in% val] = NA
+ }
> #
> bike_data$hum[is.na(bike_data$hum)] = mean(bike_data$hum, na.rm = T)
> bike_data$windspeed[is.na(bike_data$windspeed)] = mean(bike_data$windspeed,
> #
```

Outliers is the important steps of data preprocessing we have to perform them to make values accurate.

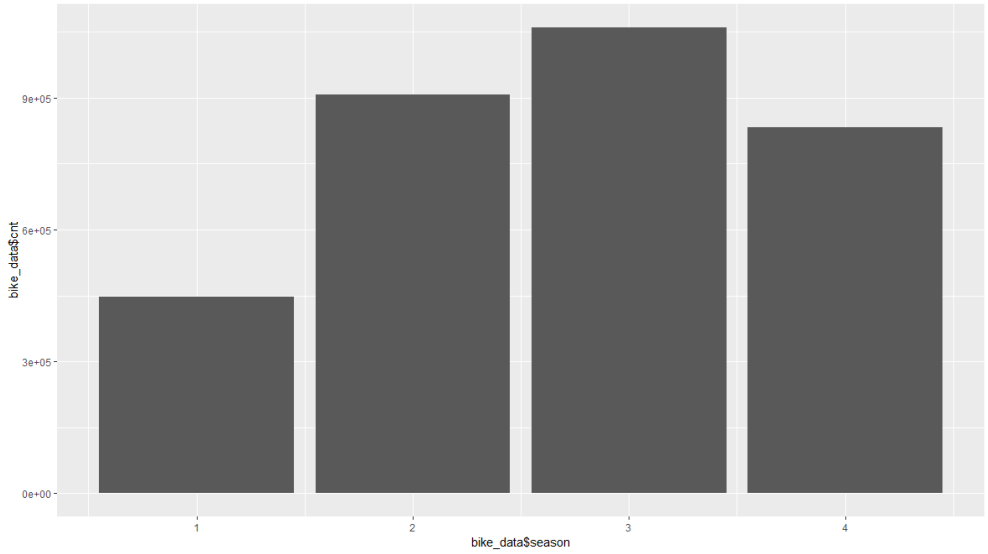
2.2 Data Visualization

Data Visualization is important concept it will help us to understand data , and will tell us answer of various questions also it will show relation between variables.

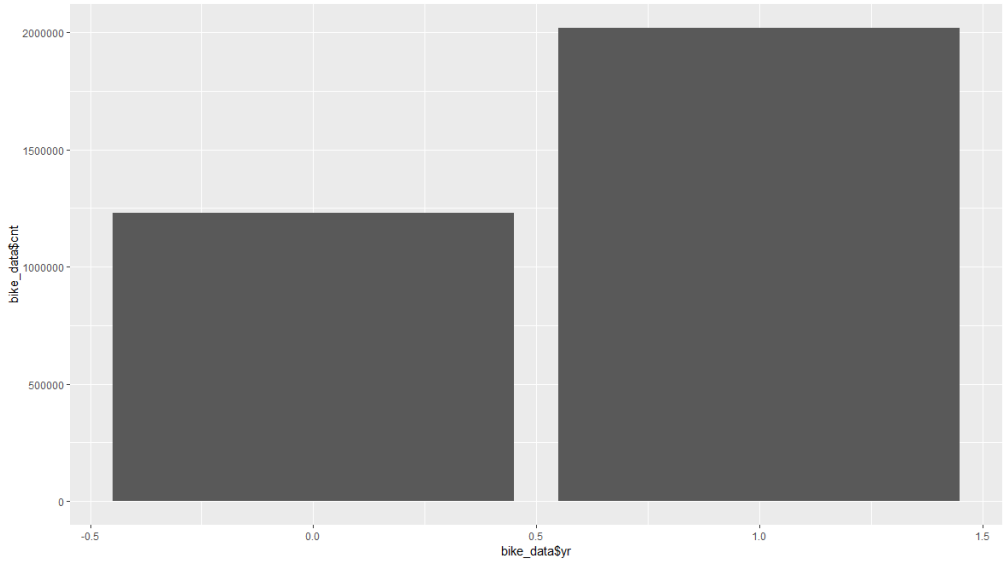
- Relation between count and days, it showing peoples are renting bike less in last days of month, or this graph cannot give us fruitful result. There is no satisfying result.



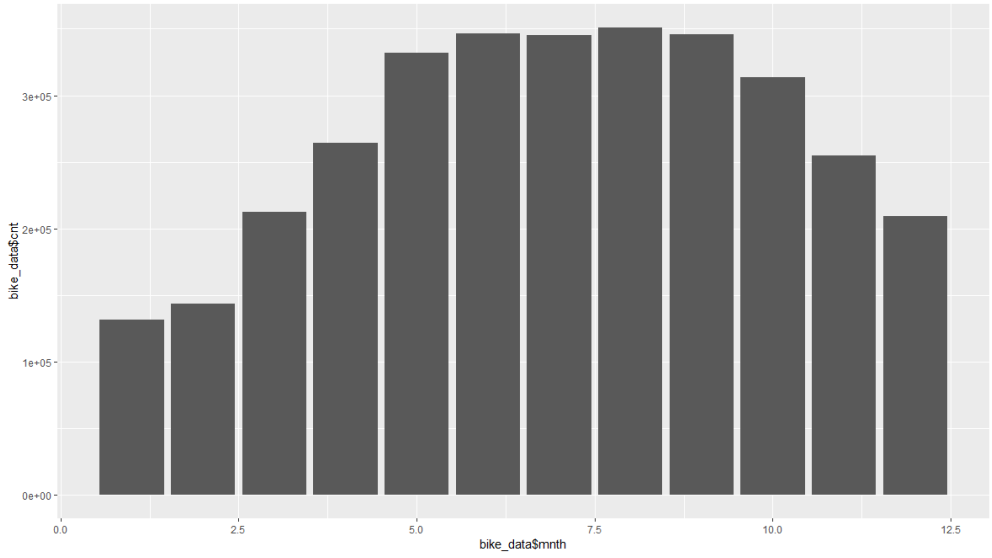
- See most of the peoples rent bike on fall(3) rather than spring(1), These are useful results and this would be important to modelling and prediction values.



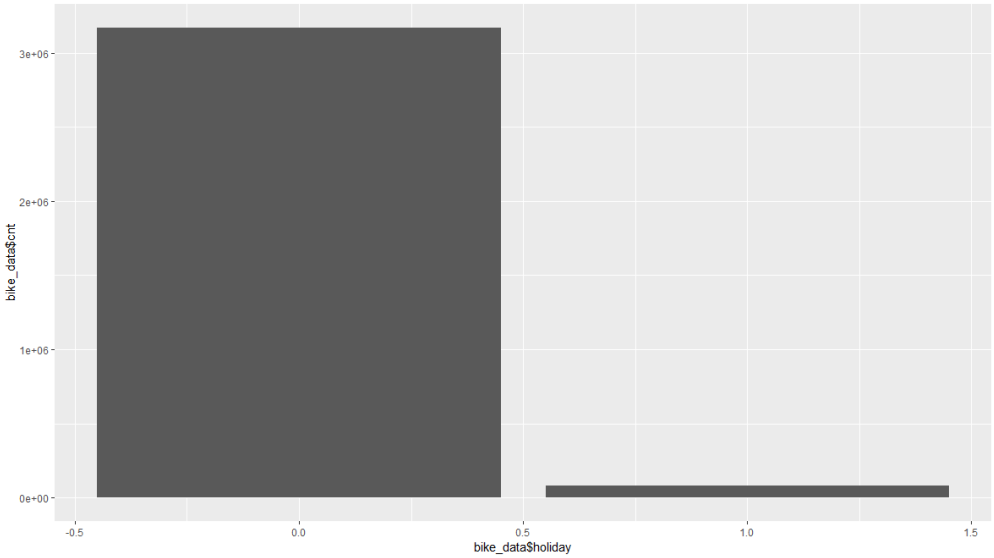
- Relation with Bike Rent and year, Most people hire bike in 2012 year rather than 2011 year, we can assume that season or environment may be more good in 2012 than 2011 or we can say that it is usual growth by increasing consumers.



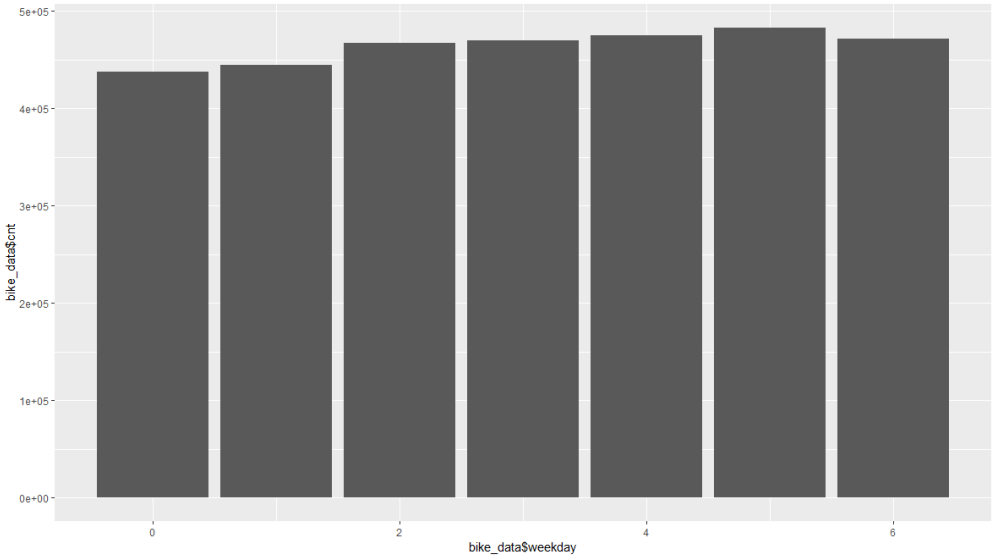
- We can see season summer(2) and fall(3) directly related to most of months with largest Bike rent 5 to 9 months. This is a valid relation between data and can be used in modelling.



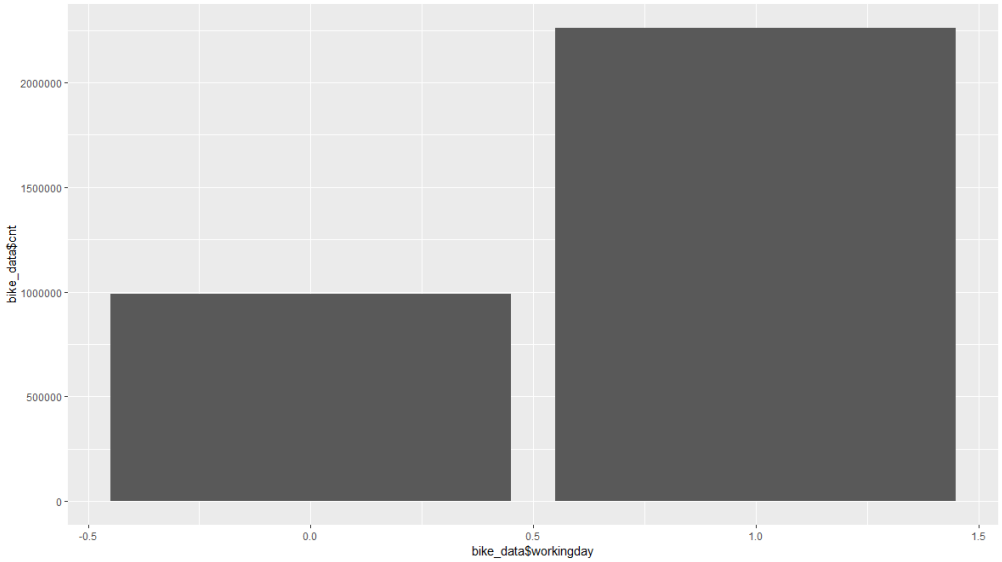
- No one rent bike on holiday, only 1 % of cases are there when on holiday people hired bike. These values are important for modelling and we can also say that some of these values are outliers but its useful.



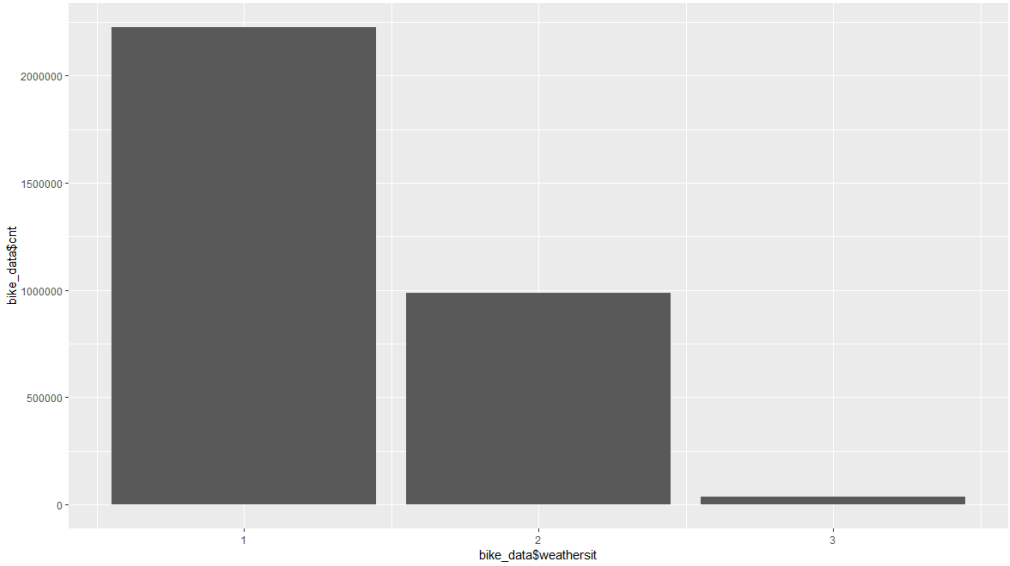
- Relation between weekdays and count, not so few rents on Sunday. This is common result and be used further not so impactfully.



- Working Day and count, bike are more hired on working day. Peoples less hired bike in holiday , most of them would be on long ride after renting.



- No one hired bike on heavy rain or snow fall weather and less for light rain and snow fall, Peoples hired bike on clear weather too much. This variable I useful too much for predictive model.



Code:

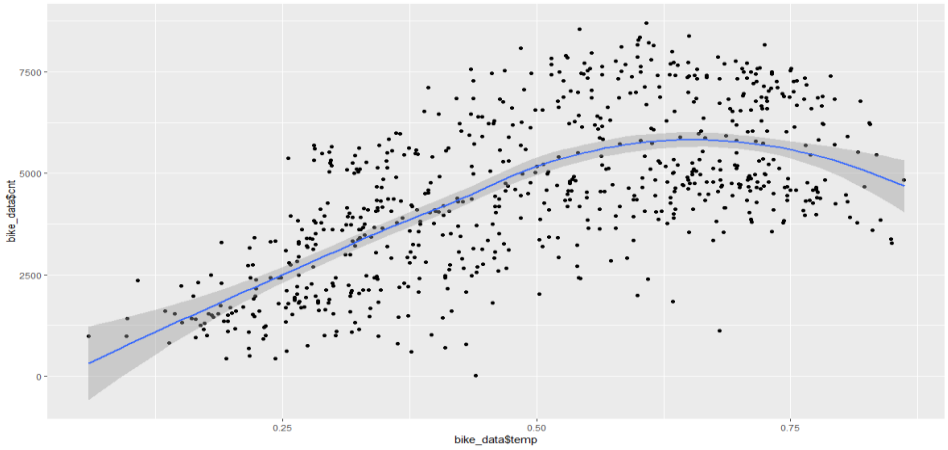
```
> ggplot(bike_data, aes(x = bike_data$day, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE)
> ggplot(bike_data, aes(x = bike_data$season, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE)
> ggplot(bike_data, aes(x = bike_data$yr, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
> ggplot(bike_data, aes(x = bike_data$mnth, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
> ggplot(bike_data, aes(x = bike_data$holiday, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
> ggplot(bike_data, aes(x = bike_data$weekday, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
> ggplot(bike_data, aes(x = bike_data$workingday, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
> ggplot(bike_data, aes(x = bike_data$weathersit, y = bike_data$cnt )) +
  geom_col(show.legend = TRUE )
```

Some More visualization with continuous variable and cnt

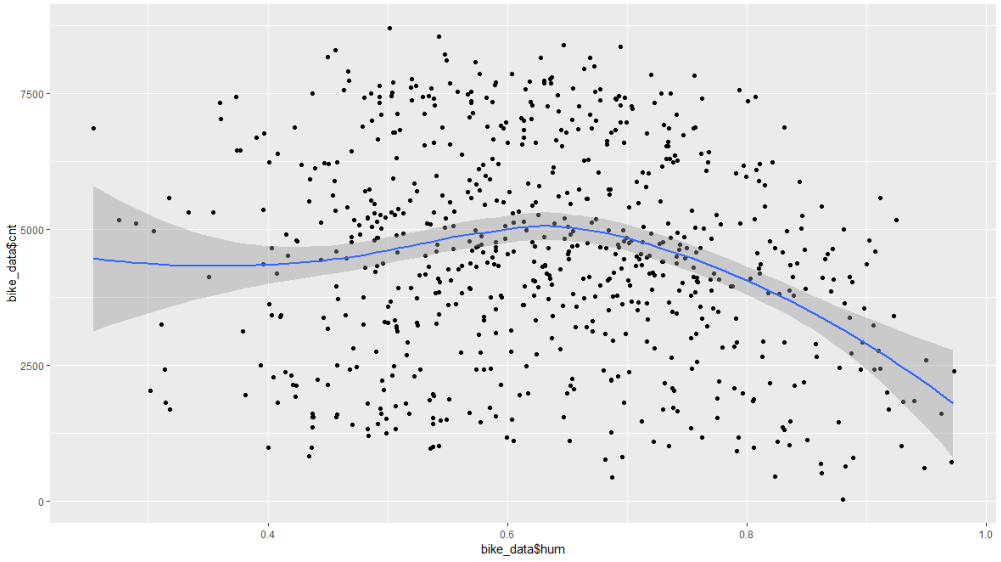
Code:

```
> ggplot(bike_data, aes(x = bike_data$temp, y = bike_data$cnt )) + geom_jitte
r() + geom_smooth(method = loess, formula = y ~ x)
>
> ggplot(bike_data, aes(x = bike_data$atemp, y = bike_data$cnt )) + geom_jitt
er() + geom_smooth(method = loess, formula = y ~ x)
>
> ggplot(bike_data, aes(x = bike_data$hum, y = bike_data$cnt )) + geom_jitter
() + geom_smooth(method = loess, formula = y ~ x)
>
> ggplot(bike_data, aes(x = bike_data$windspeed, y = bike_data$cnt )) + geom_
jitter() + geom_smooth(method = loess, formula = y ~ x)
```

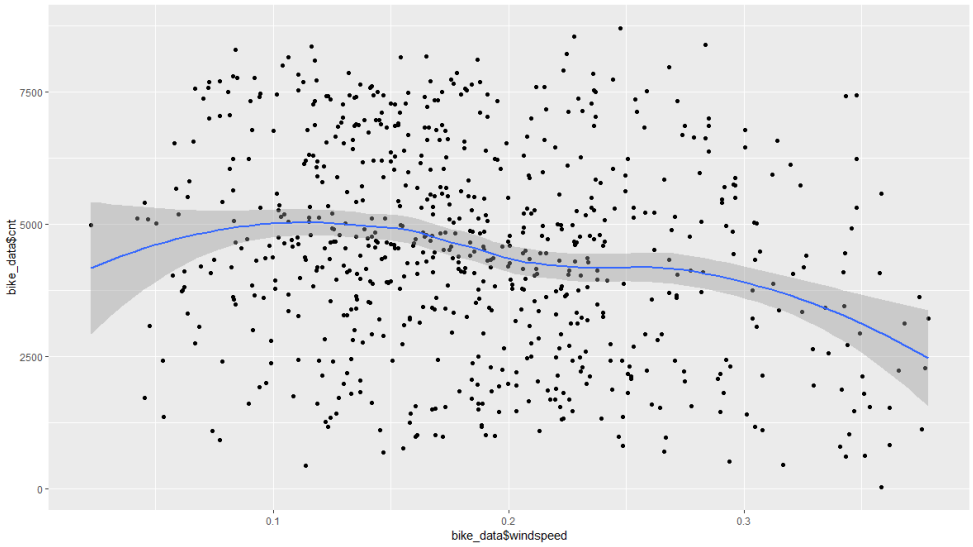
- Relation between count and cnt, its showing parabolic lines peoples don't rent bike on high temperature days. Graph line increase but reached at a peak point where renting rate is high and after increase in temp also decreased in cnt.



- Humidity is directly proportional to count sometimes but little bit when there in excess humidity peoples seems to less interested in renting bike.



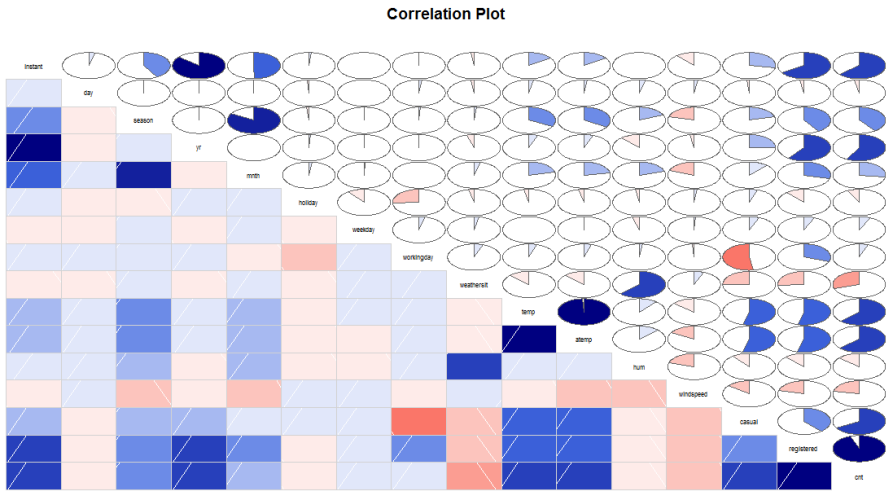
- Windspeed also directly proportional to count , graph is same as humidity .



2.3 Feature Selection

Before performing any type of modeling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that. Below will build correlation graph and see vif.

- Correlation Grapgh



We can see that cnt is highly correlated with casual and registered and temp and atemp are also highly correlated. Lets do another test.

Here we will drop *atemp* it is highly correlated and *instant* as it is nothing just as a index.

With that we will also drop *day* variable it is also no fruitful to our data because of no relation.

It will decrease features of our model due to high so many levels.

We will also drop two most highly correlated variables which are *casual* and *registered*.

If we wouldn't remove those variable , they will show same *cnt* variable in prediction values.

```
vifcor(bike_data[,-16], th = 0.9)
1 variables from the 15 input variables have collinearity problem:

atemp

After excluding the collinear variables, the linear correlation coefficien
ts ranges between:
min correlation ( hum ~ instant ):  7.544427e-05
max correlation ( yr ~ instant ):  0.8669562

---------- VIFs of the remained variables --------
    Variables          VIF
1     instant 1.755804e+05
2         day 3.077550e+02
3      season 4.134124e+00
4          yr 1.322422e+05
5        mnth 4.327133e+04
6     holiday 1.102950e+00
7     weekday 1.048113e+00
8  workingday 3.152986e+00
9  weathersit 2.047081e+00
10       temp 2.738029e+00
11        hum 2.047654e+00
12  windspeed 1.191565e+00
13     casual 3.621165e+00
14 registered 5.939648e+00
```

By above result we need to remove *casual*, *registered* and *atemp* variable as they are highly correlation, hence will give no good prediction model. Instance column is also of no use.

```
vif(df)
    Variables      VIF
1         day 1.014600
2      season 3.963615
3          yr 2.410294
4        mnth 3.391187
5     holiday 1.096210
6     weekday 1.044821
7  workingday 1.079793
8  weathersit 2.028748
9        temp 2.410973
10        hum 2.046136
11  windspeed 1.185952
12        cnt 4.974007
```

Its look like our data is fit, move to modelling now.

3 Modelling

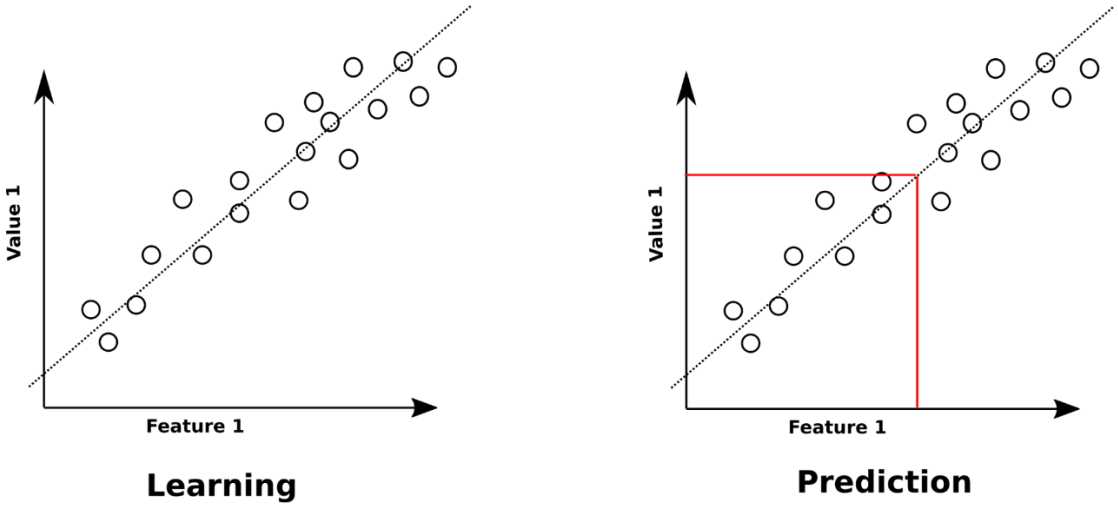
As our problem statement suggest that we have to do prediction by using our useful predictor variable.
Our problem is regression problem, we have to predict count of daily rent of bike.
For regression we have to use linear regression or decision tree regression algorithms.

3.1 Linear Regression

Linear Regression is a regression algorithm (but you probably figured that out). This algorithm's principle is to find a linear relation within your data (see below). Once the linear relation is found, predicting a new value is done with respect to this relation.

Main advantages :

- very simple algorithm
- doesn't take a lot of memory
- quite fast
- easy to explain



Main drawbacks:

• requires the data to be linearly spread (see « Polynomial Regression » if you think you need a polynomial fitting)

• is unstable in case features are redundant, i.e. if there is multicollinearity (note that, in that case you should have a look to « Elastic-Net or « Ridge-Regression »).

As we will create model first look for the code below.

```
> lm_model = lm(cnt ~., data = df)
> summary(lm_model)

Call:
lm(formula = cnt ~ ., data = df)

Residuals:
    Min      1Q  Median      3Q     Max
-4131.6  -449.0    38.6   532.1  3037.2

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1530.17     233.20   6.562 1.02e-10 ***
season        521.81      55.18   9.456  < 2e-16 ***
yr           2033.25      66.02  30.799  < 2e-16 ***
mnth          -39.15      17.25  -2.270  0.02352 *
holiday      -523.50     202.83  -2.581  0.01005 *
weekday        67.09      16.42   4.085 4.90e-05 ***
workingday    123.00      72.69   1.692  0.09108 .
weathersit   -607.00      81.42  -7.455 2.59e-13 ***
temp         5192.81     196.77  26.391  < 2e-16 ***
hum          -999.64     329.24  -3.036  0.00248 **
windspeed   -2456.44     488.64  -5.027 6.29e-07 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 880.8 on 720 degrees of freedom
Multiple R-squared:  0.7961,   Adjusted R-squared:  0.7933
F-statistic: 281.1 on 10 and 720 DF,  p-value: < 2.2e-16
```

We got R-squared value around **80%** approx.
Now, We will Check for accuracy of model below.

```
> predictions_LR = predict(lm_model, df[,1:10])
> library(MLmetrics)
> MAPE(df[,11], predictions_LR)
[1] 0.1757791
```

Here our percentage is **17.57%**, it means our accuracy is **82.43%.** We got decent accuracy.
Our values are stored in ‘predicions_LR’ .

3.2 Decision Tree Regression

The Decision Tree algorithm is a classification and regression algorithm. It subdivides learning data into regions having similar features. Descending the tree allows the prediction of the class or value of the new input data point.

Main advantages:

- quite simple
- easy to communicate about
- easy to maintain
- few parameters are required and they are quite intuitive
- prediction is quite fast

Main drawbacks:

- can take a lot of memory (the more features you have, the deeper and larger your decision tree is likely to be)
- Naturally overfits a lot (it generates high-variance models, it suffers less from that if the branches are pruned, though)
- Not capable of being incrementally improved

Let's build our decision Tree

```
> fit = rpart(cnt ~ ., data = train, method = 'anova')
> fit
n= 585

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

 1) root 585 2165444000 4543.979
   2) temp< 0.432373 230  477611000 3039.230
     4) season< 3 151  207572600 2448.550
       8) yr< 0.5 84   28352980 1762.500 *
       9) yr>=0.5 67   90116600 3308.672
        18) temp< 0.2927355 33   21515930 2584.303 *
        19) temp>=0.2927355 34   34479100 4011.735 *
     5) season>=3 79  116653200 4168.253
      10) yr< 0.5 40   21327960 3504.025 *
      11) yr>=0.5 39   59576840 4849.513 *
   3) temp>=0.432373 355  829642400 5518.887
     6) yr< 0.5 169  124955300 4264.669
      12) weathersit>=2.5 8    1031700 2320.500 *
      13) weathersit< 2.5 161   92182730 4361.273 *
     7) yr>=0.5 186  197289200 6658.473
      14) hum>=0.771458 26   34459440 5445.846 *
      15) hum< 0.771458 160  118385000 6855.525 *
```

Above we can see our decision tree, how it divides data within range. Now we move to predictions.

```
> # predictions = predict(fit, test[,-12])
> predictions_TR = predict(fit, test[,-11])
> library(MLmetrics)
> MAPE(test[,11], predictions_TR)
[1] 0.187154
```

We are getting **18.71%** of error in our prediction values or we are getting **81.39%** of accuracy in our model.
Let's store these value in csv file.

4 Conclusion

4.1 Model Evaluation

Our model was of regression model, we used only two machine learning algorithm.
Which is Linear and decision tree regression.

- Linear Regression:

In Linear Regression we got **83% Accuracy** and **17% Error** by mean absolute percentage error.

- Decision Tree:

In Decision Tree, we got **18.71%** of error in our prediction values or we are getting **81.39%** of accuracy in our model.

4.2 Model Selection

By this we can see that we can select either of two models they are almost same in means of accuracy and fits to our model.

5 Codes

5.1 R Code

```
# Remove Lists
rm(list = ls())

# Set Working Directory
setwd("D:/Bike Renting")

# Get Working Directory
getwd()

# Load Data
bike_data <- read.csv("day.csv", header = T)

# Load Library
library(tidyr)
library(plyr)
library(psych)
library(ggplot2)
library(rpart)
library(usdm)
library(rpart)
library(MASS)
library(MLmetrics)
library(DMwR)

# Explore Data
str(bike_data)

# Missing Value Analysis
sum(is.na(bike_data))

#Multi Histogram for finding out Outlier

multi.hist(bike_data$temp, main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")

multi.hist(bike_data$atemp, main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")

multi.hist(bike_data$hum, main = NA, dcol = c("blue", "red"),  # Outlier
```

```
        dlty = c("solid", "solid"), bcol = "linen")

multi.hist(bike_data$windspeed, main = NA, dcol = c("blue", "red"),   # Outlier
        dlty = c("solid", "solid"), bcol = "linen")

multi.hist(bike_data$cnt, main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")

# We can see some skewness in 'hum'  and  'windspeed'
#
# # Split 'dteday' column into new Day column
bike_data <- separate(data = bike_data,col = "dteday",into = c("year","month","day"))
bike_data$year <- NULL
bike_data$month <- NULL

## Convert day to int from
bike_data$day <- as.numeric(as.character(bike_data$day))
#
# Plot BoxPlot For them and lets go with Outlier Analysis

numeric_index = sapply(bike_data,is.numeric)
numeric_data = bike_data[,numeric_index]
cnames = colnames(numeric_data)

for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]), x = "cnt"), data = subset(bike_data))+
        stat_boxplot(geom = "errorbar", width = 0.5) +
        geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                     outlier.size=1, notch=FALSE) +
        theme(legend.position="bottom")+
        labs(y=cnames[i],x="cnt")+
        ggtitle(paste("Box plot of cnt for",cnames[i])))
}
#gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
#gridExtra::grid.arrange(gn13,gn14,gn16,ncol=3)
gridExtra::grid.arrange(gn11,gn12,ncol=2)

# hum and windspeed have outliers, Lets remove those values
outlier_val <- c('hum' , 'windspeed')
#
for(i in outlier_val){
  val = bike_data[,i][bike_data[,i] %in% boxplot.stats(bike_data[,i])$out]
  #print(length(val))
```

```
  bike_data[,i][bike_data[,i] %in% val] = NA
}
#
bike_data$hum[is.na(bike_data$hum)] = mean(bike_data$hum, na.rm = T)
bike_data$windspeed[is.na(bike_data$windspeed)] = mean(bike_data$windspeed, na.rm = T)
#

##
boxplot(bike_data$hum)
multi.hist(bike_data$hum, main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")

boxplot(bike_data$windspeed)
multi.hist(bike_data$windspeed, main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")

#Visualization

##Int value
ggplot(bike_data, aes(x = bike_data$day, y = bike_data$cnt )) + geom_col(show.legend = TRUE)
ggplot(bike_data, aes(x = bike_data$season, y = bike_data$cnt )) + geom_col(show.legend =
TRUE)
ggplot(bike_data, aes(x = bike_data$yr, y = bike_data$cnt )) + geom_col(show.legend = TRUE )
ggplot(bike_data, aes(x = bike_data$mnth, y = bike_data$cnt )) + geom_col(show.legend = TRUE
)
ggplot(bike_data, aes(x = bike_data$holiday, y = bike_data$cnt )) + geom_col(show.legend =
TRUE )
ggplot(bike_data, aes(x = bike_data$weekday, y = bike_data$cnt )) + geom_col(show.legend =
TRUE )
ggplot(bike_data, aes(x = bike_data$workingday, y = bike_data$cnt )) + geom_col(show.legend
= TRUE )
ggplot(bike_data, aes(x = bike_data$weathersit, y = bike_data$cnt )) + geom_col(show.legend =
TRUE )

##Numeric Value
ggplot(bike_data, aes(x = bike_data$temp, y = bike_data$cnt )) + geom_jitter() +
geom_smooth(method = loess, formula = y ~ x)

ggplot(bike_data, aes(x = bike_data$atemp, y = bike_data$cnt )) + geom_jitter() +
geom_smooth(method = loess, formula = y ~ x)

ggplot(bike_data, aes(x = bike_data$hum, y = bike_data$cnt )) + geom_jitter() +
geom_smooth(method = loess, formula = y ~ x)
```

```
ggplot(bike_data, aes(x = bike_data$windspeed, y = bike_data$cnt )) + geom_jitter() +
geom_smooth(method = loess, formula = y ~ x)

# Correlation graph
library(corrgram)
corrgram(bike_data[,numeric_index], order = F,
       upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
vifcor(bike_data[,-16], th = 0.9)

# Lets make towards Model
# Drop some nouse variable
#df <- bike_data[, -c(1,11,14:15)]
df <- bike_data[, -c(1:2,11,14:15)]

vifcor(df[-11], th = 0.9)

vif(df)

# Linear Regression
lm_model = lm(cnt ~., data = df)

summary(lm_model)

# p  0.05 Accept NULL Hypothesis and can say that this variable not relevant to us

predictions_LR = predict(lm_model, df[,1:10])
library(MLmetrics)
MAPE(df[,11], predictions_LR)

# 83%

write.csv(predictions_LR, file = 'D:/Bike Renting/countpredictLR.csv')
# Decision Tree Regression

n = nrow(df)
trainIndex = sample(1:n, size = round(0.8*n), replace=FALSE)
train = df[trainIndex ,]
test = df[-trainIndex ,]


# r part for decision tree regreession

fit = rpart(cnt ~ ., data = train, method = 'anova')
```

```
# predictions = predict(fit, test[,-12])
predictions_TR = predict(fit, test[,-11])
library(MLmetrics)


MAPE(test[,11], predictions_TR)
# 81%
write.csv(predictions_TR, file = 'D:/Bike Renting/countpredictTR.csv')
```

5.2 Python Code

```
import os
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from sklearn.cross_validation import train_test_split
import statsmodels.api as sm

#os.chdir("D:\bike")
path = r"D:\bike"

#Load Data
bike_data = pd.read_csv("day.csv")

bike_data.head(5)

# Check Missing Values
pd.DataFrame(bike_data.isnull().sum())

# Data summary
bike_data.describe()

# Data Info
bike_data.info()
```

Data Preprocessing

```
# BoxPlot
sns.boxplot(data=bike_data[['temp',
      'atemp', 'hum', 'windspeed']])

fig=plt.gcf()

#Boxplot
sns.boxplot(data=bike_data[['cnt',
      'casual']])

# Set for outliers
outlier_val = ['hum', 'windspeed']
cnames = ['instant','dteday','season','yr','mnth',
'holiday','weekday','workingday','weathersit','temp'
'atemp','hum','windspeed','casual','registered','cnt']

#Detect and delete outliers from data
for i in outlier_val:
   print(i)
   q75, q25 = np.percentile(bike_data.loc[:,i], [75 ,25])
   iqr = q75 - q25

   min = q25 - (iqr*1.5)
   max = q75 + (iqr*1.5)
   print(min)
   print(max)

   bike_data = bike_data.drop(bike_data[bike_data.loc[:,i] < min].index)
   bike_data = bike_data.drop(bike_data[bike_data.loc[:,i] > max].index)
```

Data Visualization

```
#Bar Graph
sns.factorplot(x="season",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="yr",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="mnth",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="holiday",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="weekday",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="workingday",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
sns.factorplot(x="weathersit",y="cnt",data=bike_data,kind='bar',size=5,aspect=1.5)
```

```
# Scatter Plot
plt.scatter(x="temp",y="cnt",data=bike_data,color='#ff4125')
plt.scatter(x="atemp",y="cnt",data=bike_data,color='#ff4125')
plt.scatter(x="hum",y="cnt",data=bike_data,color='#ff4125')
plt.scatter(x="windspeed",y="cnt",data=bike_data,color='#ff4125')
```

Feature Engineering

```
#hist of cnnt
%matplotlib inline
plt.hist(bike_data['cnt'], bins='auto')

# Correlation Graph
df_corr = bike_data.loc[:,cnames]

#Set the width and hieght of the plot
f, ax = plt.subplots(figsize=(7, 5))

#Generate correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool), cmap=sns.diverging_palette(220,
10, as_cmap=True),
        square=True, ax=ax)

# Drop use less variable which can be used to predict
bike_data = bike_data.drop(['instant','dteday','atemp','casual','registered'], axis=1)
```

Model Development

```
# Linear Regresssion
train, test = train_test_split(bike_data, test_size=0.2)

model = sm.OLS(bike_data.iloc[:,10], bike_data.iloc[:,0:10]).fit()

model.summary()

predictions_LR = model.predict(test.iloc[:,0:10])

def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
   return mape
#Calculate MAPE
MAPE(test.iloc[:,10], predictions_LR)
```

19.694792307271566

Here we are getting 81% accuracy

Decision Tree

```
from sklearn.tree import DecisionTreeRegressor

dtrain, dtest = train_test_split(bike_data, test_size=0.2)

fit_DT = DecisionTreeRegressor(max_depth=2).fit(bike_data.iloc[:,0:10], bike_data.iloc[:,10])

fit_DT

def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
   return mape

MAPE(dtest.iloc[:,10], predictions_DT)
```

Store LR prediction value

Because In python we are not getting good accuracy of decision tree , as in Linear Regression ,But linear regression always preferred in industry for regression problem

```
predictions_LR.to_csv('predictions_LR.csv', sep=',', encoding='utf-8', index= False)
```