Predicting Bike Rental Counts using R and Python

By
Subhra Kanchan Pattnaik

Submitted on Date: December 10 2018

Contents

Introduction

1.1 Problem Statement

Predicting the count of the rental bikes on daily based on the environmental and seasonal settings. The aim is to predict the use of rental bikes in future which can be determined from the data provided. We would like to predict the count of rental bikes based on the environment in order to know that in which particular season people are opting for more rental bikes and based on which more bikes can be hired.

1.2 Data

Here the task is to build a classification model based on season or the type of environment which will classify the count of rental bikes for every season or environment. Below is the sample data provided in order to provide solution to above statement.

Table 1.1 : Below table contains the sample data which contains total of 16 variables where table 1.1 contains 1st 8 variables and table 1.2 contains another 8 variables.

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
5	1/5/2011	1	0	1	0	3	1
6	1/6/2011	1	0	1	0	4	1
7	1/7/2011	1	0	1	0	5	1

Table 1.2: another 8 Variables

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
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606
2	0.196522	0.208839	0.498696	0.168726	148	1362	1510

Below is the description of variables mentioned in the above table:

instant: Record index
dteday: Date
season: Season (1:springer, 2:summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted fromHoliday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted fromFreemeteo) 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

Chapter 2

Methodology

2.1 Pre Processing

In order to develop a model for a dataset we need to 1st understand the data, structure of data because understanding a data will only help us to identify the variables which are required and which are not required and then we can clean the data and the cleaned data can be used for the final model and predicting purpose. The process of exploring and cleaning of data is often called as **Exploratory Data Analyses**. After the Problem statement is identified then we need to analyze the data and we need to identify the independent and dependent variables also we need to identify the categorical and numerical variable. So these are all the steps of the exploratory Data Analyses also need to identify those variables which are not required. Hence in order to clean the Data and to make the data meaningful we need preprocessing steps. Also we need to check the variable types.

According to the dataset "Day.CSV", we can identify that the cnt variable present is numerical and we are considering "cnt" variable as dependent variable and other 15 variables as independent variable.

2.1.1 Outlier Analyses

The 1st preprocessing analyses after doing Exploratory Data Analyses is **Missing Value Analyses.**
1st step is converting the data into a proper data type that is all numeric data and categorical data into a proper data type.
Missing Value analyses is used to find any values which are missing and after that we can apply the KNN OR MEAN OR MEDIAN imputation on those missing values in order to build a Model on top of it.

Below is the R code where the data has been converted into proper data type and then we have calculated the missing values.

R-Code:

#set the directory

```
rm(list=ls())
setwd("E:/Newfolder/R_experiments/Email_marketing_conversionmaster/Email_marketing_conversion-master")
x=c("ggplot2", "corrgram", "DMwR", "Caret", "ramdomForest", "unbalanced", "c50", "dummies", "e1071","Information", "MASS", "rpart", "gbm", "ROSE")
lapply(x, require, character.only = TRUE)
rm(x)
```

#we have loaded the "day" data below:

```
bike_prediction =read.csv("day.csv", header = T,na.strings = c(" ", "", "NA"))
str(bike_prediction)
names(bike_prediction)
```

#converting the data into proper data type:

```
bike_prediction$holiday = as.factor(as.character(bike_prediction$holiday))
str(bike_prediction)
bike_prediction$weekday = as.factor(as.character(bike_prediction$weekday))
bike_prediction$workingday = as.factor(as.character(bike_prediction$workingday))
bike_prediction$weathersit = as.factor(as.character(bike_prediction$weathersit))
missing_val = data.frame(apply(bike_prediction, 2,function(x){sum(is.na(x))}))
View(bike_prediction)
bike_prediction[69,2]
library(dplyr)
bike_prediction$season = factor(format(bike_prediction$season, format="%A"),levels = c("1", "2","3","4"), labels = c("Spring","Summer","Fall", "Winter"))
table(bike_prediction$season)
table(bike_prediction$weathersit)
bike_prediction$holiday = factor(format(bike_prediction$holiday, format="%A"), levels = c("0","1"), labels =c("working_day", "holiday"))
bike_prediction$weathersit = factor(format(bike_prediction$weathersit, format="%A"), levels = c("1", "2","3","4"), labels = c("Good:Clear/Sunny","Moderate:Cloudy/Mist","Bad: Rain/Snow/Fog","Worse: Heavy Rain/Snow/Fog"))
bike_prediction$yr = factor(format(bike_prediction$yr, format="%A"),levels = c("0", "1") , labels = c("2011","2012"))
table(bike_prediction$yr)
```

If the Missing value is more than 30% then we should not impute the variable with any of the imputation methods such as mean, median or knn imputations.

After checking we found that there are no missing values for the data provided.

#Packages are loaded

```
Loading required package: ggplot2
Loading required package: corrgram
```

```
Loading required package: DMwR
Loading required package: lattice

Attaching package: ‘lattice’

The following object is masked from ‘package:corrgram’:

    panel.fill

Loading required package: grid
Loading required package: Caret
Loading required package: ramdomForest
Loading required package: unbalanced
Loading required package: mlr
Loading required package: ParamHelpers
Loading required package: foreach
Loading required package: doParallel
Loading required package: iterators
Loading required package: parallel
Loading required package: c50
Loading required package: dummies
dummies-1.5.6 provided by Decision Patterns

Loading required package: e1071

Attaching package: ‘e1071’

The following object is masked from ‘package:mlr’:

    impute

Loading required package: Information
Loading required package: MASS
Loading required package: rpart
Loading required package: gbm
Loaded gbm 2.1.4
Loading required package: ROSE
Loaded ROSE 0.0-3

[[1]]
[1] TRUE

[[2]]
[1] TRUE
```

[[3]]
[1] TRUE

[[4]]
[1] FALSE

[[5]]
[1] FALSE

[[6]]
[1] TRUE

[[7]]
[1] FALSE

[[8]]
[1] TRUE

[[9]]
[1] TRUE

[[10]]
[1] TRUE

[[11]]
[1] TRUE

[[12]]
[1] TRUE

[[13]]
[1] TRUE

[[14]]
[1] TRUE

Warning messages:
1: In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :
there is no package called ‘Caret’
2: In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :
there is no package called ‘ramdomForest’
3: In library(package, lib.loc = lib.loc, character.only = TRUE, logical.return = TRUE, :
there is no package called ‘c50’

```
> bike_prediction$dteday = as.Date(bike_prediction$dteday)
> bike_prediction$season = as.factor(bike_prediction$season)
> bike_prediction$yr = as.factor(bike_prediction$yr)
> bike_prediction$mnth = as.factor(bike_prediction$mnth)
> bike_prediction$holiday = as.factor(bike_prediction$holiday)
> bike_prediction$weekday = as.factor(bike_prediction$weekday)
> bike_prediction$workingday = as.factor(bike_prediction$workingday)
> bike_prediction$weathersit = as.factor(bike_prediction$weathersit)
> numeric_index = sapply(bike_prediction,is.numeric)
> numeric_index
  instant     dteday     season         yr       mnth    holiday    weekday
     TRUE      FALSE      FALSE      FALSE      FALSE      FALSE      FALSE
workingday weathersit       temp      atemp        hum  windspeed     casual
    FALSE      FALSE       TRUE       TRUE       TRUE       TRUE       TRUE
registered        cnt
     TRUE       TRUE

#outlier Analyses which works only for numerical variable
numeric_index = sapply(bike_prediction,is.numeric)
numeric_index
numeric_data = bike_prediction[,numeric_index]
cnames = colnames(numeric_data)

#loop to remove outliers from all the variables
df=bike_prediction
for(i in cnames){
  print(i)
  val = bike_prediction[,i][bike_prediction[,i] %in% boxplot.stats(bike_prediction[,i])$out]
bike_prediction = bike_prediction[which(!bike_prediction[,i] %in% val),]
}
[1] "instant"
[1] "temp"
[1] "atemp"
[1] "hum"
[1] "windspeed"
[1] "casual"
[1] "registered"
[1] "cnt"
```

#We can see below that there is no missing values for the bike_prediction fo all the variables. Now we will go to Outlier analyses where we will be using boxplot.

```
sum(is.na(bike_prediction))
[1] 0
for(i in cnames){
```

```
  val = bike_prediction[,i][bike_prediction[,i] %in% boxplot.stats(bike_prediction[,i])$out]
bike_prediction[,i][bike_prediction[,i] %in% val] = NA
}
#Now as we can see some missing values are generated across 1.e. nearly 21
#Hence now we need to replace or impute the missing avlues with the help of KNN
bike_prediction = knnImputation(bike_prediction, k = 3)
```

> sum(is.na(bike_prediction))
[1] 0

#Outlier analyses can be applied only in numerical values hence
> **for(i in cnames){**
+ val = bike_prediction[,i][bike_prediction[,i] %in% boxplot.stats(bike_prediction[,i])$out]
+ bike_prediction[,i][bike_prediction[,i] %in% val] = NA
+ }
> #Now as you can see some missing valuesa re generated accros i.e 21 that is initially 731 observations are there and 676 observatios and 16 variables are present as of now.

> cnames
[1] "instant" "temp" "atemp" "hum" "windspeed" "casual"
[7] "registered" "cnt"
> #Hence now we need to replace or impute the missing values with the help of KNN
> bike_prediction = knnImputation(bike_prediction, k = 3)

Below are the boxplots

```
> library(ggplot2)
> ggplot(bike_prediction,aes(yr,cnt)) +
+   geom_boxplot(fill = c("#8DD3C7","#FFFFB3")) +
+   theme_classic() +
+   labs(title = "Boxplot of rental bikes per year") +
+   scale_x_discrete(labels = c("2011","2012"))
```

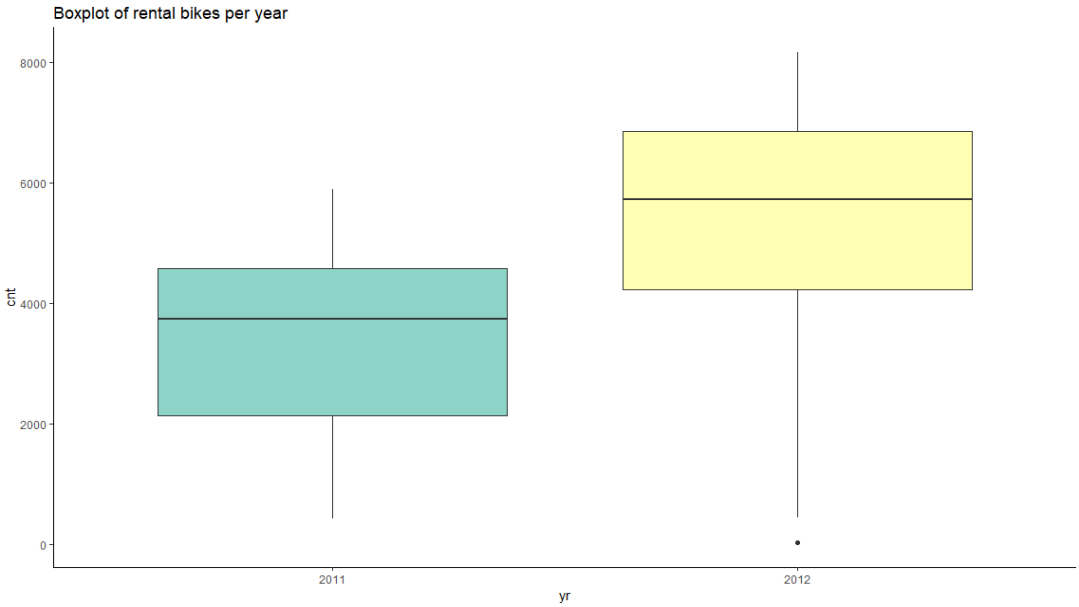


FIG. 2.1 FIGURE REPRESENTS BOXPLOT FOR RENTAL BIKES PER YEAR

```
ggplot(bike_prediction,aes(mnth,cnt)) +
+ geom_boxplot(fill = c("#8DD3C7","#FFFFB3",
"#8DD3C7","#FFFFB3","#8DD3C7","#FFFFB3",
"#8DD3C7","#FFFFB3","#8DD3C7","#FFFFB3", "#8DD3C7","#FFFFB3")) +
+   theme_classic() +
+   labs(title = "Boxplot of rental bikes per month")
```

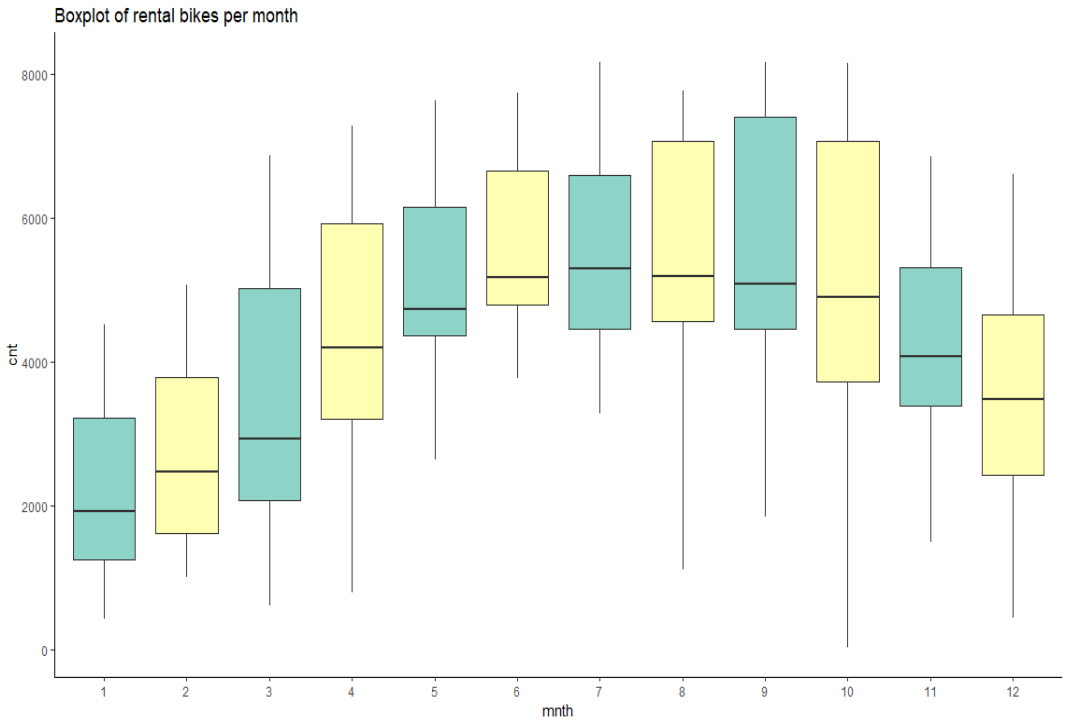


FIG. 2.2 FIGURE REPRESENTS BOXPLOT FOR RENTAL BIKES PER MONTH

```
ggplot(bike_prediction,aes(holiday,cnt)) +
    geom_boxplot(fill = c("#8DD3C7","#FFFFB3")) +
    theme_classic() +
    labs(title = "Boxplot of rental bikes by holiday") +
    scale_x_discrete(labels = c("no","yes"))
```

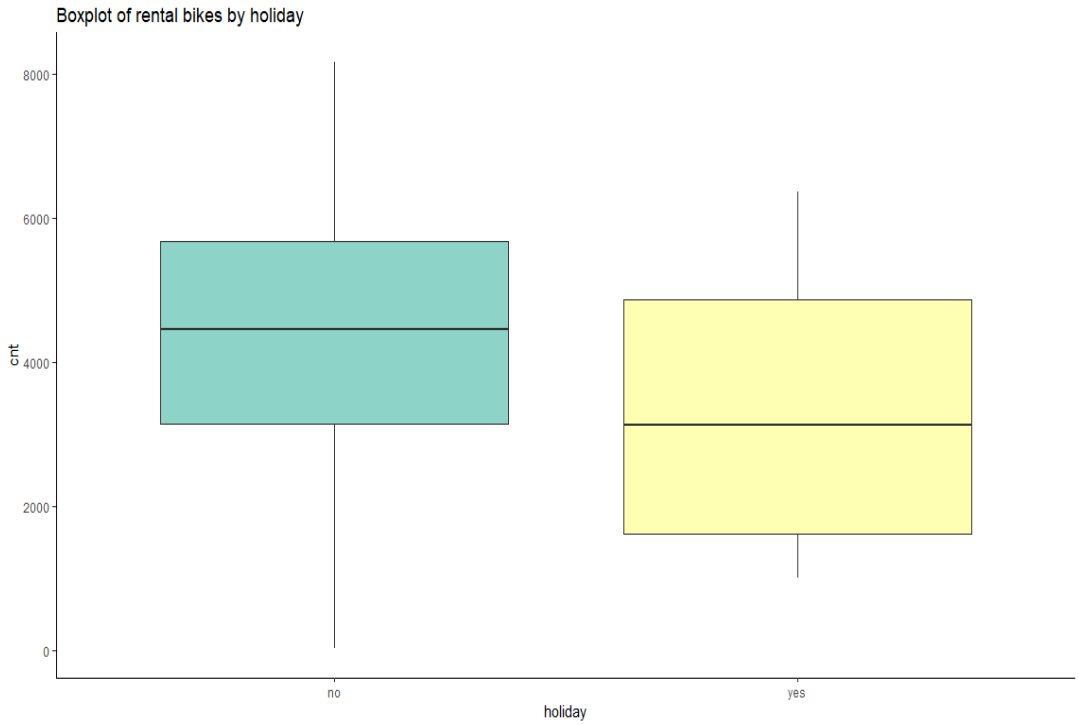


FIG. 2.3 FIGURE REPRESENTS BOXPLOT FOR RENTAL BIKES BY HOLIDAY

```
ggplot(bike_prediction,aes(cnt,temp)) +
    geom_point() +
    geom_smooth(method = "loess") +
    theme_classic()
```

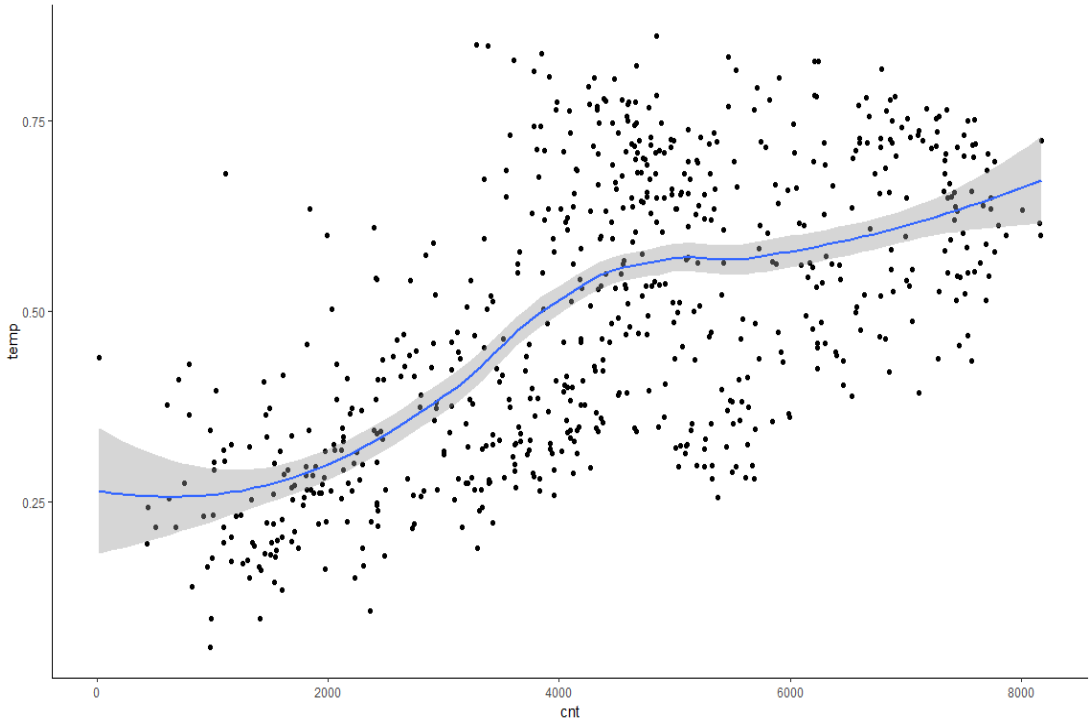


FIG. 2.4 FIGURE REPRESENTS RELATIONSHIP BETWEEN TEMP AND CNT OF BIKES VARIABLES

```
ggplot(bike_prediction,aes(weathersit,cnt)) +
    geom_boxplot(fill =c("#8DD3C7","#FFFFB3", "#8DD3C7") )+
    theme_classic() +
    labs(title = "Boxplot of rental bikes by weathersit") +
    scale_x_discrete(labels = c("Cloudy","Mist","Light Rain"))
  unique(bike_prediction$cnt)
```

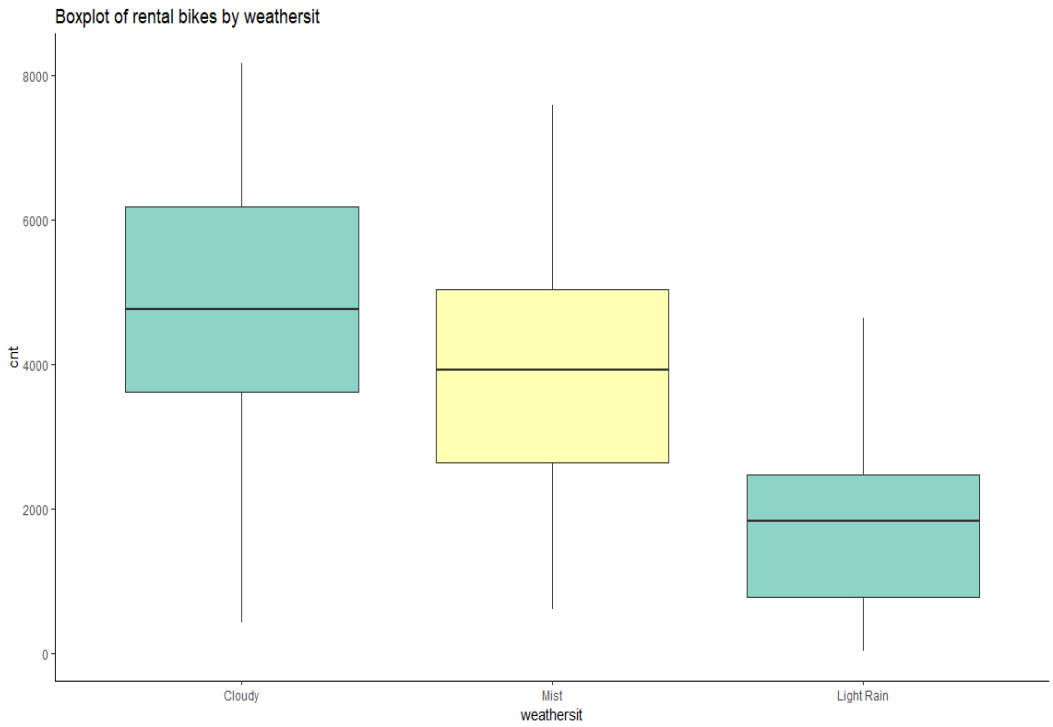


FIG. 2.5 FIGURE REPRESENTS BOXPLOT FOR RENTAL BIKES PER YEAR

2.1.2 Feature Selection

Feature Selection is a technique which allows us to predict the relevant features from the raw data. In general it allows us to filter those data or variables which are needed for use in Model.

Feature selection and Feature Engineering or extractions are two jargons which are used, where feature selection is extracting the subset of data whereas feature engineering is creating a new variable out of the old variable. Feature selection thus reduces the redundant data and hence applies the model with simplicity. It is also called Dimensionality reduction.

There are 2 different methods that are correlation analyses and chi square methods. So correlation is applied only on continuous data where the independent and dependent variable should be highly correlated. If two independent variable are highly correlated then we can reject one independent variable. Chi-Square test is used only for categorical value.

#feature analyses-correlation analyses--whether any redundant variable is there between independent variable then we need to remove it
 # Now we will be sealing with correlation plot-- we are not reducing the observation by using outlier analyses and we will be using correlation

```
corrgram(bike_prediction[,numeric_index], order = F,
        upper.panel = panel.pie, text.panel= panel.txt, main ="correlation plot")
```

Below is the correlation plot where we are considering only numeric variable and avoiding all categorical variable as the correlation plot can be plotted with continues variables only.

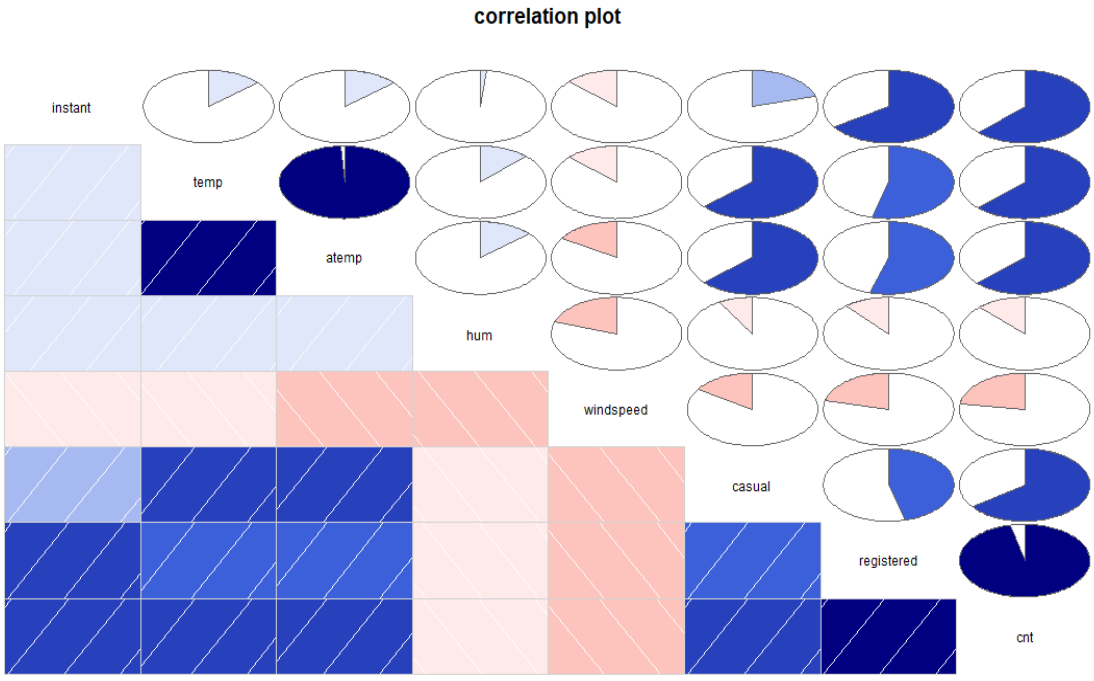


FIG. 2.6 FIGURE REPRESENTS CORRELATION PLOT BETWEEN ALL THE CONTINUOUS VARIABLES

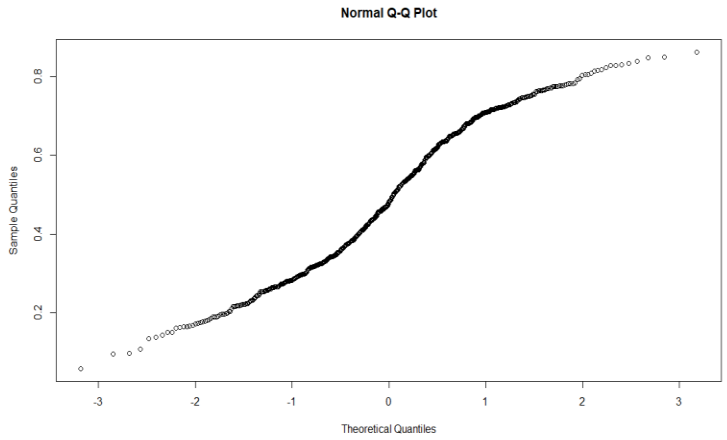
Secondly we are using categorical variable which cleans only categorical variable, but since our data is continues variable and the dependent variable is also continues hence we will not be using Chi-Square test.

2.1.3 Feature Scaling

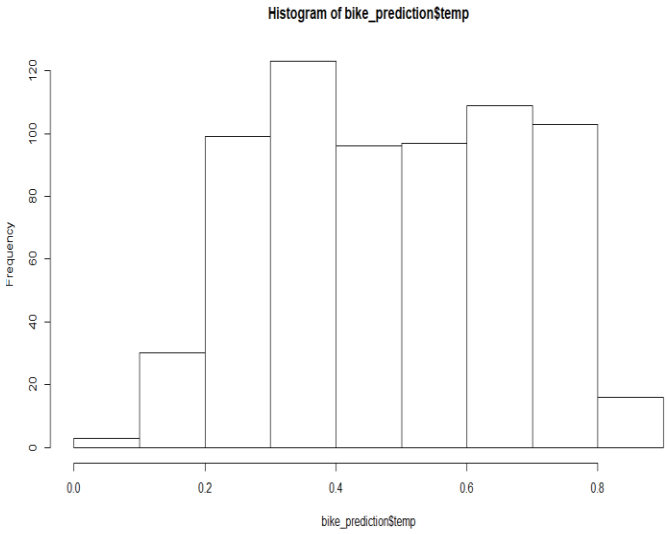
Another preprocessing technique is Feature Scaling it is performed only on continuous variable. It can be divided into two parts that is normalization and standardization. Normalization is the process of reducing unwanted variation either within or between variables.

Standardization is also another preprocessing technique. If the data is normally distributed then we go for standardization otherwise we choose normalization.
Below is how we choose we need to choose normalization or standardization.
Histogram is a better metrics to see the data is normalized or not.

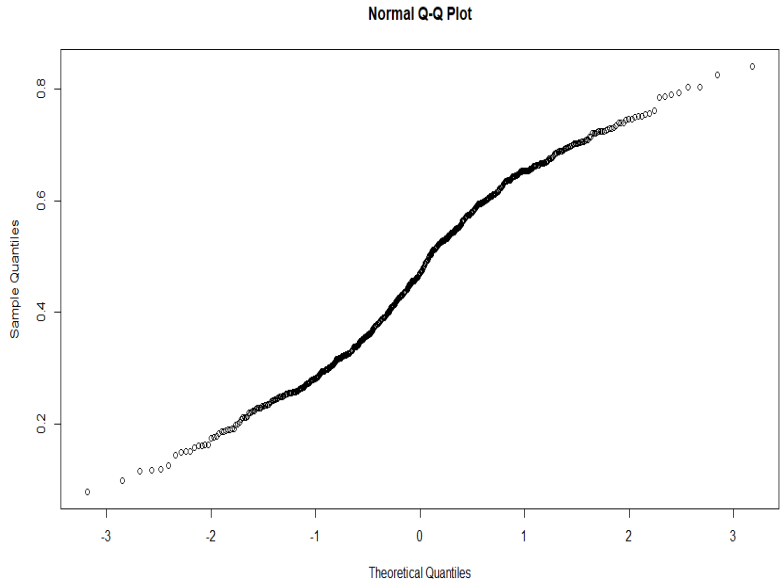
>qqnorm(bike_prediction$temp)



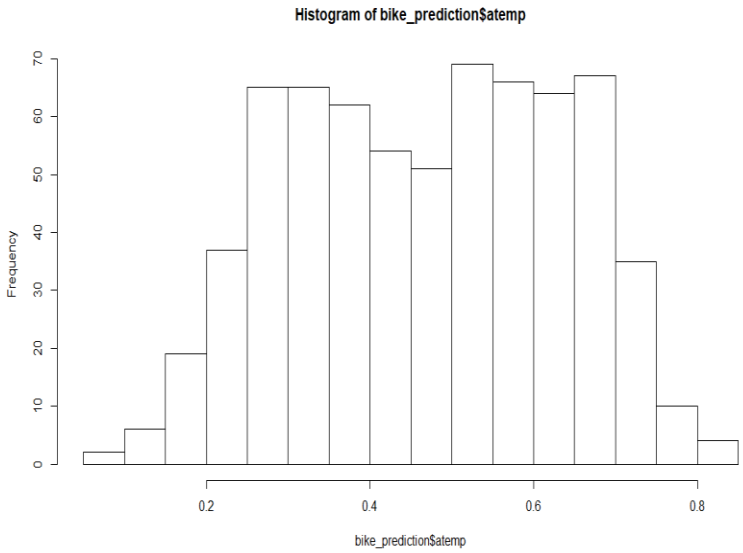
> hist(bike_prediction$temp)



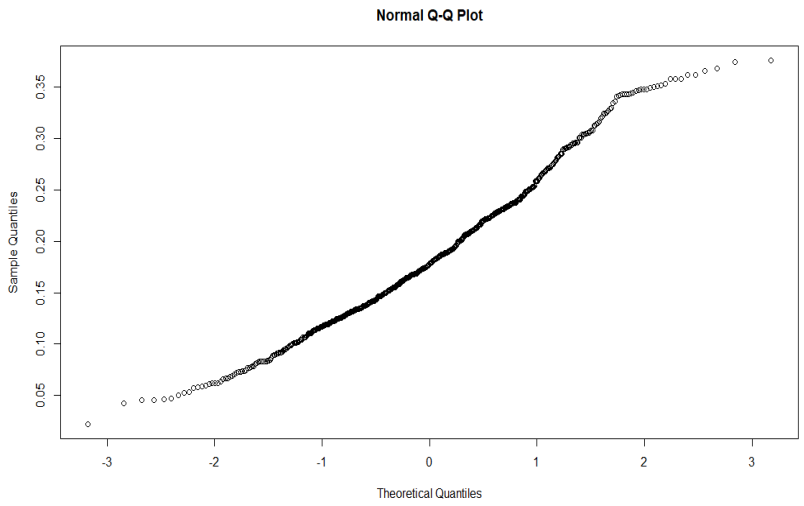
> qqnorm(bike_prediction$atemp)



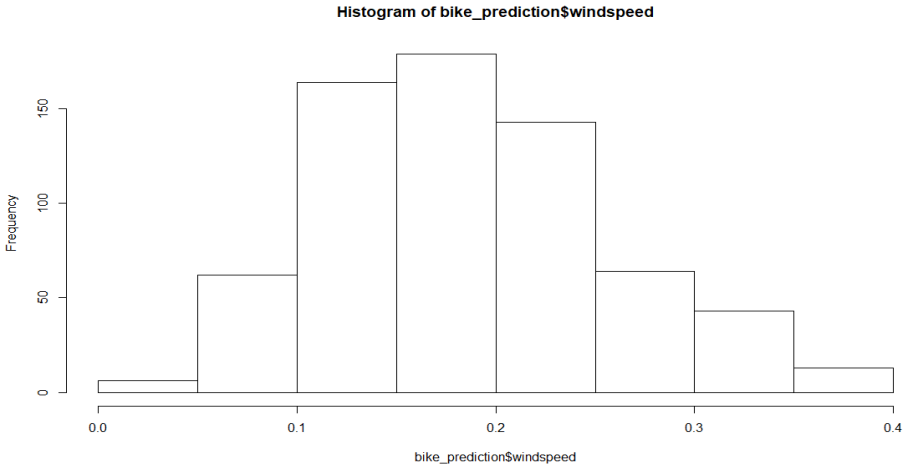
> hist(bike_prediction$atemp)



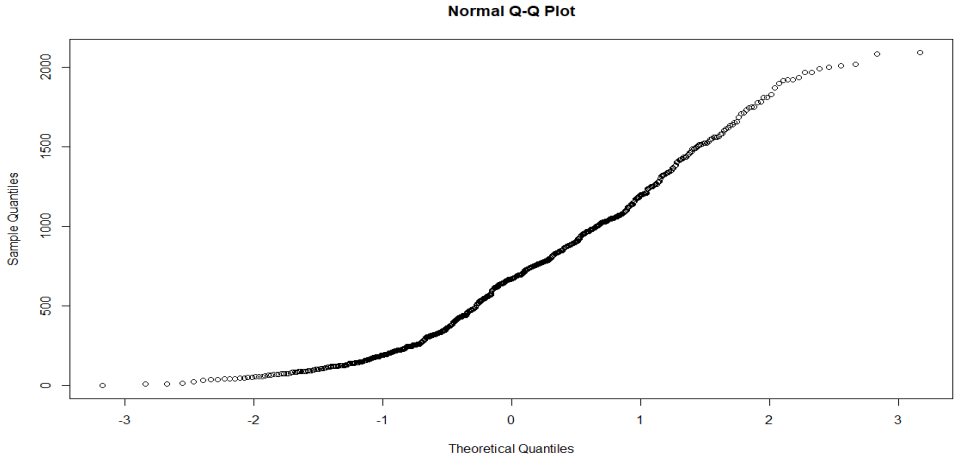
> qqnorm(bike_prediction$windspeed)



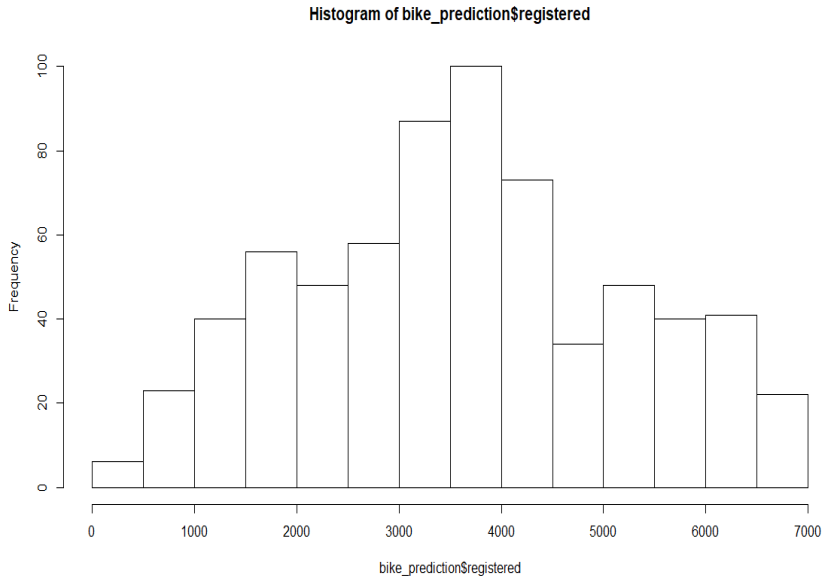
> hist(bike_prediction$windspeed)

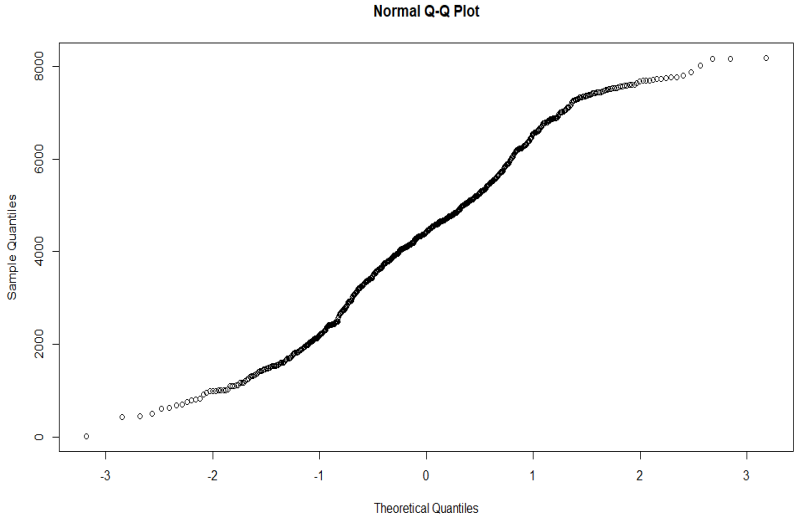


> qqnorm(bike_prediction$registered)

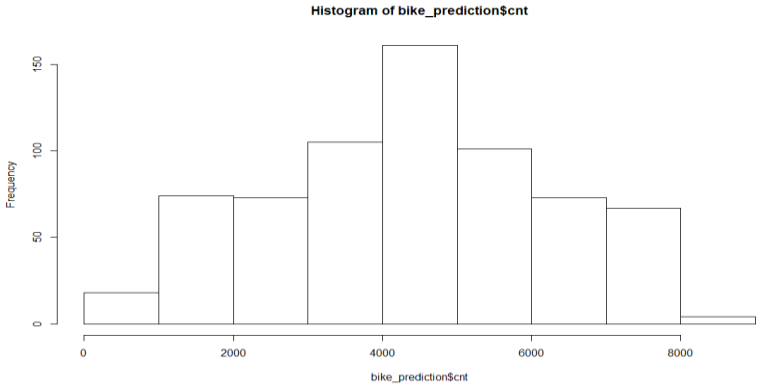


> hist(bike_prediction$registered)





>hist(bike_prediction$cnt)



Below is the formula for the Normalization. After applying the formula we will see that all the numerical value will range in between 0 to 1.

```
> for(i in cnames){
+   print(i)
+   bike_prediction[,i] = (bike_prediction[,i] - min(bike_prediction[,i]))/(max(bike_prediction[,i] - min(bike_prediction[,i])))
+ }
[1] "instant"
[1] "temp"
[1] "atemp"
[1] "hum"
[1] "windspeed"
[1] "casual"
[1] "registered"
[1] "cnt"
> View(bike_prediction)
> bike_prediction =read.csv("day.csv", header = T,na.strings = c(" ", "", "NA"))
> View(bike_prediction)
> bike_prediction$dteday = as.Date(bike_prediction$dteday)
> bike_prediction$season = as.factor(bike_prediction$season)
> bike_prediction$yr = as.factor(bike_prediction$yr)
> bike_prediction$mnth = as.factor(bike_prediction$mnth)
> bike_prediction$holiday = as.factor(bike_prediction$holiday)
> bike_prediction$weekday = as.factor(bike_prediction$weekday)
> bike_prediction$workingday = as.factor(bike_prediction$workingday)
> bike_prediction$weathersit = as.factor(bike_prediction$weathersit)
> numeric_index = sapply(bike_prediction,is.numeric)
> numeric_index
  instant     dteday     season         yr       mnth    holiday    weekday
     TRUE      FALSE      FALSE      FALSE      FALSE      FALSE      FALSE
workingday weathersit       temp      atemp        hum  windspeed     casual
    FALSE      FALSE       TRUE       TRUE       TRUE       TRUE       TRUE
registered        cnt
     TRUE       TRUE
> numeric_data = bike_prediction[,numeric_index]
> cnames = colnames(numeric_data)
> cnames
[1] "instant"    "temp"       "atemp"      "hum"        "windspeed"  "casual"
[7] "registered" "cnt"
>
```

Below is the method of standardization if the target variable s normalized then we need to apply the below formula.

```
> bike_prediction[,i] = (bike_prediction[,i] - mean(bike_prediction[,i]))/sd(bike_prediction[,i])
```

```
> View(bike_prediction)
> bike_prediction =read.csv("day.csv", header = T,na.strings = c(" ", "", "NA"))
> install.packages("DataCombine")
Installing package into ‘C:/Users/Subhra/Documents/R/win-library/3.5’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/DataCombine_0.2.21.zip'
Content type 'application/zip' length 118805 bytes (116 KB)
downloaded 116 KB

package ‘DataCombine’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Subhra\AppData\Local\Temp\RtmpiI8jC0\downloaded_packages
> library(DataCombine)
> rmExcept("bike_prediction")
Removed the following objects:

cnames, df, factor_data, factor_index, i, numeric_data, numeric_index, val, x
```

2.2 Modeling

2.2.1 Model Selection

Model Development on the DATA SET

We can use the decision tree regression method, we cannot apply decision tree classification method as the data set is continuous variable hence we are using decision tree regression method. Out of 16 variables we have 15 independent variable and one dependent variable that is “cnt” variable is dependent variable. We need to predict the bike count base on the weather, and seasons or environment. Also as our target variable is continues hence we are choosing simple random sampling method.

Below is the R code describing the test and training data and then applying the sampling method in it.

2.2.2 Decision Regression

In order to develop a model we need to divide the dataset into train and test data.

```
> set.seed(1234)
> install.packages("caret")
Installing package into ‘C:/Users/Subhra/Documents/R/win-library/3.5’
(as ‘lib’ is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/3.5/caret_6.0-81.zip'
Content type 'application/zip' length 6133562 bytes (5.8 MB)
downloaded 5.8 MB
```

package ‘caret’ successfully unpacked and MD5 sums checked

The downloaded binary packages are in
> #######Model Development#######
> library(caret)

Attaching package: ‘caret’

The following object is masked from ‘package:mlr’:

> train.index = createDataPartition(bike_prediction$cnt, p = .80, list = FALSE)
> train = bike_prediction[train.index,]
> test = bike_prediction[-train.index,]

bike_prediction$dteday = as.Date(bike_prediction$dteday)
bike_prediction$season = as.factor(bike_prediction$season)
bike_prediction$yr = as.factor(bike_prediction$yr)
bike_prediction$mnth = as.factor(bike_prediction$mnth)
bike_prediction$holiday = as.factor(bike_prediction$holiday)
bike_prediction$weekday = as.factor(bike_prediction$weekday)
bike_prediction$workingday = as.factor(bike_prediction$workingday)
bike_prediction$weathersit = as.factor(bike_prediction$weathersit)

#Checking the missing value
missing_val = data.frame(apply(bike_prediction, 2,function(x){sum(is.na(x))}))
sum(is.na(bike_prediction))
[1] 0
View(bike_prediction)

#As our target variabe is continuous hence we are using simple random sampling Method and Decision Regression Method

train_index = sample(1:nrow(bike_prediction), 0.8*nrow(bike_prediction))
train = bike_prediction[train_index,]
test = bike_prediction[-train_index,]
fit = rpart(cnt ~.,data = train, method = "anova")
predictions_DT = predict(fit, test[,-16])

> train_index = sample(1:nrow(bike_prediction), 0.8*nrow(bike_prediction))
> train = bike_prediction[train_index,]
> test = bike_prediction[-train_index,]
> fit = rpart(cnt ~.,data = train, method = "anova")
> fit
n= 584

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

 1) root 584 2187308000 4463.575
   2) registered< 2914.5 182  168330200 2235.176
     4) registered< 2096.5 112   39346700 1644.536
       8) registered< 1347.5 44    6322991 1092.136 *
       9) registered>=1347.5 68   10909670 2001.971 *
     5) registered>=2096.5 70   27396690 3180.200 *
   3) registered>=2914.5 402  706039800 5472.453
     6) registered< 4283 210   90186290 4469.790
      12) casual< 613.5 80   12368450 3914.588 *
      13) casual>=613.5 130   37982410 4811.454 *
     7) registered>=4283 192  173821800 6569.115
      14) registered< 5550 113   74510350 6035.310
        28) casual< 1323.5 81   17563990 5628.123 *
        29) casual>=1323.5 32    9522188 7066.000 *
      15) registered>=5550 79   21055450 7332.658 *
```

Now we can apply the model on the test Data as shown below:

```
> predictions_DT = predict(fit, test[,-16])
```

3. Conclusion

3.1 Model Evaluation:

The error metrics can be used for both classification and regression, such as MSE(Mean Square Error) is used for classification. The regression error metrics is MAPE, RMSE. So below we have applied the MAPE method is used to find the error rate in our test data and from that we can found the accuracy of the model.

3.2 MAPE

Now we need to calculate the error by using MAPE method.

#calculate MAPE

```
> mape = function(y,yhat){
+       mean(abs((y-yhat)/y))
+ }
> mape(test[,16],predictions_DT)
[1] 0.112317
> mape = function(y,yhat){
+       mean(abs((y-yhat)/y))*100
```

```
+ }
> mape(test[,16],predictions_DT)
```
[1] 11.2317

From above we can see 11.2 % error so our model is nearly 89% accuracy. Also we can use the above model as 80% above accuracy is fine for the model.

We can also apply different other models and calculate other error metrics in order to apply the Model on test Data.

4. Python Code

Python Code:

```
 import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contigency
import seaborn as sns
from random import randrange, uniform

#set working directory
os.chdir("E:\New folder\R_experiments\Email_marketing_conversion-
master\Email_marketing_conversion-master")
os.getcwd()
#load data
bike_prediction = pd.read_csv("day.csv")
bike_prediction.shape
1.Exploratory Data Analyse
2. Missing Values===preprocessing technique(a)
We need to find the missing values for each of variable
missing_value = pd.DataFrame(bike_prediction.isnull().sum())
===n missing values in the output
3. Preprocessing technique===outlier analyses
To detect outlier let us plot the outlier
matplotlib inline
plt.boxplot(bike_prediction[“instant”,”temp”,"atemp","hum","windspeed","casual","registered","
cnt]
##save numeric name
cnames("instant","temp”,"atemp","hum","windspeed","casual","registered","cnt")
#detect and delete outliers from data
for i in cnames:
print(i)
q75,q25 = np.percentile(bike_prediction.loc[:,i],[75,25])
```

```
iqr = q75 - q25
min = q25 - (iqr*1.5)
max = q75 + (iqr*1.5)
print(min)
print(max)
bike_prediction = bike_prediction.drop(bike_prediction[bike_prediction.loc[;,i]< min].index)
bike_prediction = bike_prediction.drop(bike_prediction[bike_prediction.loc[;,i] > max].index)
```
After removing the outliers some less obsevations are found which is same as R
Now the 2nd method is replace outliers with NA and try to impute as missing value
so let us reload the master data
```
q75, q25 = np.percentile(marketing_train['temp'],[75,25])
```
after finding the 75 percentile and 25th percentile of the data
```
iqr =q75-q25
```
Now calculate the upper fence and lower fence
```
minimun = q25 -(iqr*1.5)
maximum = q75 + (iqr*1.5)
bike_prediction.loc[bike_prediction['variable'] < minimum,:'temp'] = np.nan
bike_prediction.loc[bike_prediction['variable'] > maximum,:'temp'] = np.nan
#calculating missing value
missing_val = pd.Dataframe(bike_prediction.isnull().sum())
```
missing_val===we have imputed
now KNN imputation....is
```
bike_prediction = pd.DataFrame(KNN(K=3).complete(bike_prediction),columns =
bike_prediction.columns)
```
bike_prediction.isnull().sum====This is how we remove the outlier in python

=======Feature Selection -in Python
Correlation and Chi-Square test===(numerical and catagorical)

```
df_corr = bike_prediction.loc[:,cnames]
f, ax = plt.subplots(figsize=(7,5))
sns.heatmap(corr, mask =np.zero_like(corr, dtype = np.bool),
cmap=sns.diverging_palette(220,10, as_cmp=True),square=True, ax=ax)
```
Based on this we can remove the variables which are not required..
To convert the data into same range we have two methods 1. Feature Scaling and 2 another is
===normalization and standardisation

#Normality Check===histogram to check if the data is nrmally distributed or not

```
matplotlib inline
plt.hist(bike_prediction['variable'], bins='auto')
```
from here we will get if the variable is left skewed, riht skewed or anything else...

let us keep all the continuous variables to cnames and let us use loop

##sampling techniques:
2 types:probability and non probability
a. simple random samling (1 dependent)-other independent
sim_sampling = bike_prediction.sample(10)---just need to choose no. of subset observation from all the observation
==========

```
import os
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor


for i in cnames:
 print(i)
bike_prediction[i] = (bike_preddiction[i] - min(bike_prediction[i]))/(max(bike_prediction[i]) - min(bike_prediction[i]))
```

#standardisation method :

```
reload the actual data:
for i in cnames:
 print(i)
bike_prediction[i] = (bike_preddiction[i] - (bike_prediction[i].mean())/(bike_prediction[i].std()

corr = df_corr.corr()
```

==========Decision Regression=======

```
import os
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor
os.chdir("E:\New folder\R_experiments\Email_marketing_conversion-master\Email_marketing_conversion-master")
os.getcwd()

bike_prediction = pd.read_csv("day.csv")
```

Here we are not going to use stratifired sampling method a this method needs the dependent variable as catagorical thats why we are going to use simple random sampling method
Now we will be dividing the data into test and train

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

That is 20%test and 80% train
fit = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0.9],train.iloc[:,9])#Decision Tree for regression
predictions_DT = fit_DT.predict(test.iloc[:,0.9])
Now in order to evaluate the ref=gression method...mse or mpse==data based
here the data is not time base
mape
#calculate mape
def MAPE(y_true, y_pred):
mape = np.mean(np.abs((y_true - y_pred) / y_true))
return MAPE
MAPE(test.iloc[:,9],predictions_DT)

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.

http://brandonharris.io/kaggle-bike-sharing/