PREDICATION OF BIKE RENTAL COUNT

Satyam Sharma 03-07-2019

Contents

Chapter 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 need to predict count of bike rental which seems the regression statement.

1.2 Data

Our aim is to predict count of bike rental .

Data Details:

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_max- t_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

Below is the sample data for the same:

Table 1.1: Column 1 - 5

Instant	dteday	season	yr	mnth
1	01-01-2011	1	0	1
2	02-01-2011	1	0	1
3	03-01-2011	1	0	1
4	04-01-2011	1	0	1
5	05-01-2011	1	0	1
6	06-01-2011	1	0	1
7	07-01-2011	1	0	1

Table 1.2: Column 6 – 10

holiday	weekday	workingday	weathersit	Temp	atemp	hum	windspeed	casual	registered
0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654
0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670
0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229
0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454
0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518
0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518
0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362

This is the column we need to correctly predict

Table 1.3: Column 11

Cnt
985
801
1349
1562
1600
1606
1510

Chapter 2

Methodology

2.1 Pre-processing

It is a data mining technique that transforms raw data into an understandable format. Raw data(real world data) is always incomplete and that data cannot be sent through a model. That would cause certain errors. That is why we need to pre-process data before sending through a model.

<u>Steps in Data Preprocessing</u>

Here are the steps:

- Import libraries
- Read data
- Checking for missing values
- Checking for categorical data
- Outlier Analysis
- Feature Selection

2.1.1 Missing Value Analysis

There are no missing values in the dataset. No need to use any technique to handle it.

```
> colSums(sapply(ana_data, is.na))
    dteday     season         yr       mnth    holiday    weekday workingday
         0          0          0          0          0          0          0
weathersit       temp      atemp        hum  windspeed        cnt
         0          0          0          0          0          0
> sum(is.na(ana_data)) / (nrow(ana_data) *ncol(ana_data))
[1] 0
```

2.1.2 Type Conversion

2.1.2.1 Taking a glance at data:

Table 2.1: Head of data

	instant	dteday	season	yr	mnth	holiday	weekday
1	1	2011-01-01	1	0	1	0	6
2	2	2011-01-02	1	0	1	0	0
3	3	2011-01-03	1	0	1	0	1
4	4	2011-01-04	1	0	1	0	2
5	5	2011-01-05	1	0	1	0	3
6	6	2011-01-06	1	0	1	0	4

	workingday	weathersit	temp	atemp	hum	windspeed
1	0	2	0.344167	0.363625	0.805833	0.1604460
2	0	2	0.363478	0.353739	0.696087	0.2485390
3	1	1	0.196364	0.189405	0.437273	0.2483090
4	1	1	0.200000	0.212122	0.590435	0.1602960
5	1	1	0.226957	0.229270	0.436957	0.1869000
6	1	1	0.204348	0.233209	0.518261	0.0895652

	casual	registered	cnt
1	331	654	985
2	131	670	801
3	120	1229	1349
4	108	1454	1562
5	82	1518	1600

6 88 1518 1606

2.1.2.2 Structure of data:

Data Types:

Before Conversion:

```
'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 ...
```

Below features should be categorical but are having numeric type and date is not in formatted So we will need to convert them to appropriate type:

dteday
season
yr
mnth
holiday
weekday
workingday
weathersit

2.1.2.3 Data type conversion:

- Changing below features to factor:
  Season
  Month
  Year
  Holiday
  Weekday
  Working day
  Weathers it
- Changed “dteday” to date format
- Removed unnecessary features
- Removed instant, Casual, register

After Conversion:

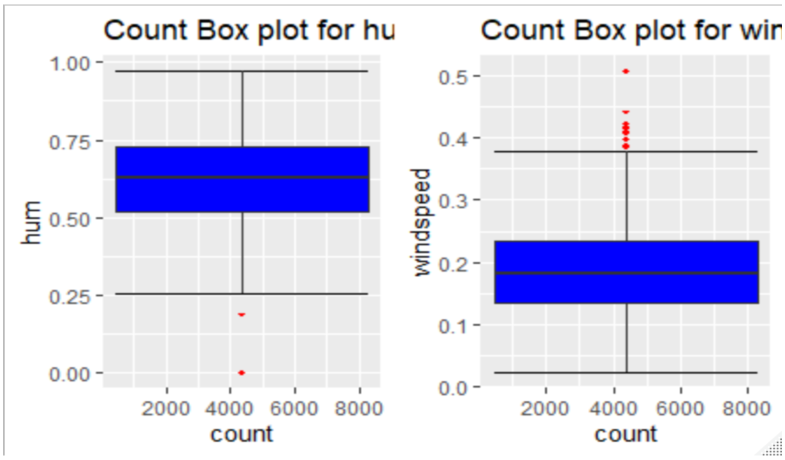
```
> str(ana_data)
'data.frame':	731 obs. of  13 variables:
 $ dteday    : Factor w/ 31 levels "01","02","03",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ season    : Factor w/ 4 levels "1","2","3","4": 1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ mnth      : Factor w/ 12 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
 $ weekday   : Factor w/ 7 levels "0","1","2","3",..: 7 1 2 3 4 5 6 7 1 2 ...
 $ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...
```

```
$ weathersit: Factor w/ 3 levels "1","2","3": 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 ...
$ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

2.1.3 Outlier Analysis

2.1.3.1 Count Plot:

Table 2.2



Here we can see some outliers in red colour.

2.1.3.2 Checking for Humidity and Windspeed feature:

Table 2.3

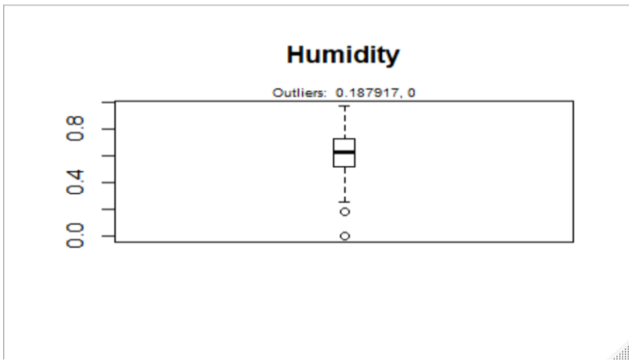
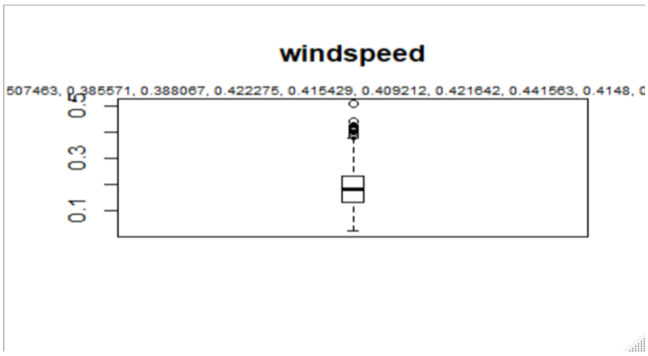


Table 2.4



Only two values are lying outside in humidity and some values in Windspeed.

2.1.3.3 Cook’s distance method to detect influence of outlier:

Calculate Cook's distance - It computes the influence exerted by each data point (row) on the predicted outcome. Cook’s distance greater than 4 times the mean may be classified as influential.

Plots:

Table 2.5

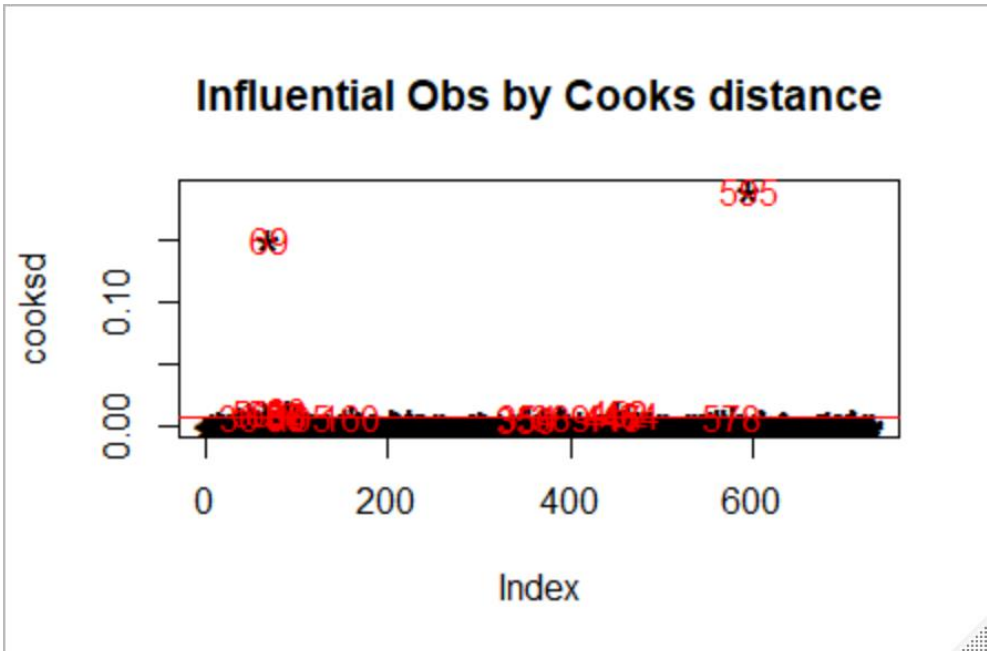
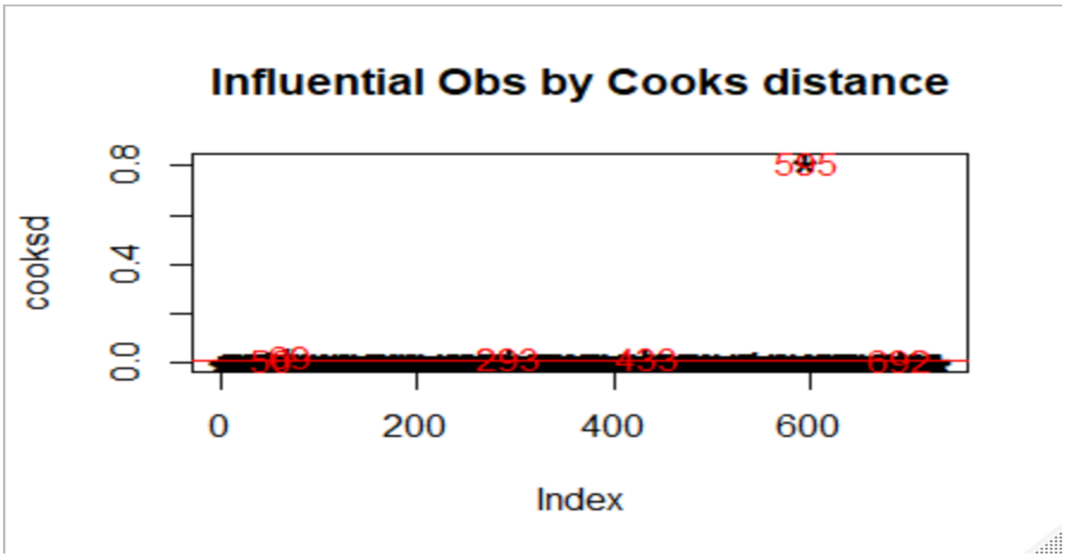


Table 2.6



Influence:

Humidity

Mean = 0.6278941

	weathersit	temp	atemp	hum	windspeed	cnt
36	2	0.233333	0.243058	0.929167	0.161079	1005
50	1	0.399167	0.391404	0.187917	0.507463	1635
65	2	0.376522	0.366252	0.948261	0.343287	605
69	3	0.389091	0.385668	0.000000	0.261877	623
86	2	0.253043	0.250339	0.493913	0.184300	1693
87	1	0.264348	0.257574	0.302174	0.212204	2028

Not much influence on any numeric data by outliers.

Windspeed

Mean = 0.1904862

2.1.4 Feature Selection

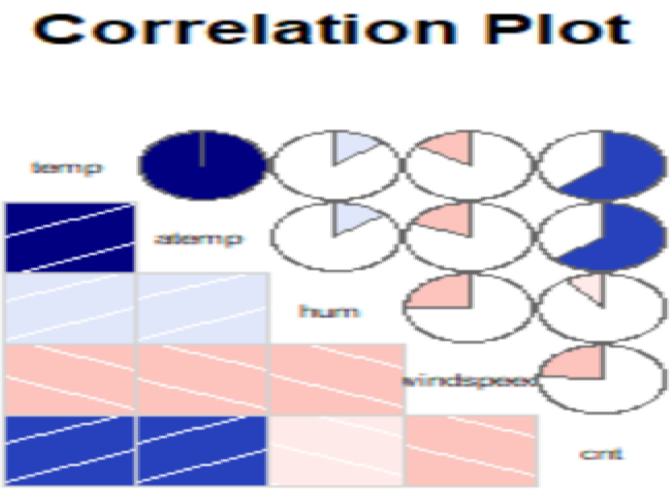
2.1.4.1 Correlation Summery

	temp	atemp	hum	windspeed	cnt
temp	1.0000000	0.9917016	0.1269629	-0.1579441	0.6274940
atemp	0.9917016	1.0000000	0.1399881	-0.1836430	0.6310657
hum	0.1269629	0.1399881	1.0000000	-0.2484891	-0.1006586
windspeed	-0.1579441	-0.1836430	-0.2484891	1.0000000	-0.2345450
cnt	0.6274940	0.6310657	-0.1006586	-0.2345450	1.0000000

We have atemp with highest correlation with count.

2.1.4.2 Correlation plot

Table 2.7 Correlation plot



2.1.4.3 Dimension Reduction:
“atemp” is highly correlated so we will remove atemp.

2.2 Modelling

2.2.1 Model Selection
Problem statement tells that this is a regression problem. So we will try some regression algorithms and evaluate the results.

Decision tree

Since the process of constructing these decision trees assume no distributional patterns in the data (non-parametric), characteristics of the input data are usually not given much attention. We consider some characteristics of input data and their effect on the learning performance of decision trees. Preliminary results indicate that the performance of decision trees can be improved with minor modifications of input data.

Random forest

Random Forest is an ensemble machine learning technique capable of performing both regression and classification tasks using multiple decision trees and a statistical technique called **bagging.**

Linear Regression

Learning a linear regression model means estimating the values of the coefficients used in the representation with the data that we have available.

2.2.2 Decision Tree

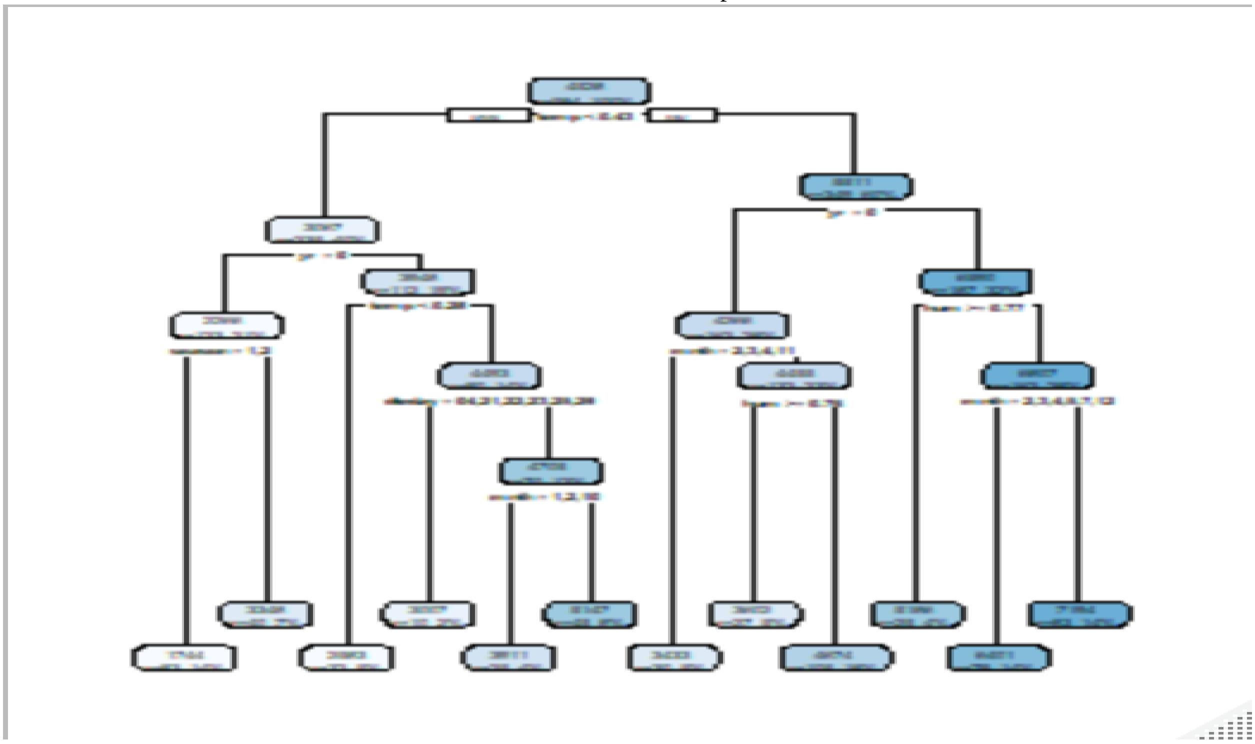
```
fit = rpart(cnt ~ ., data = train, method = "anova")
```

```
predict_DT = predict(fit, test[,-12])
```

```
predict_DT
```

```
1        2        3        5        11       12       14       16
1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494
      29       33       34       45       47       51       59       64
1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494
      69       82       87       88       89       91       93      101
1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 3433.300
     108      123      127      133      136      137      142      147
3433.300 4674.476 4674.476 3601.667 3601.667 3601.667 4674.476 4674.476
```

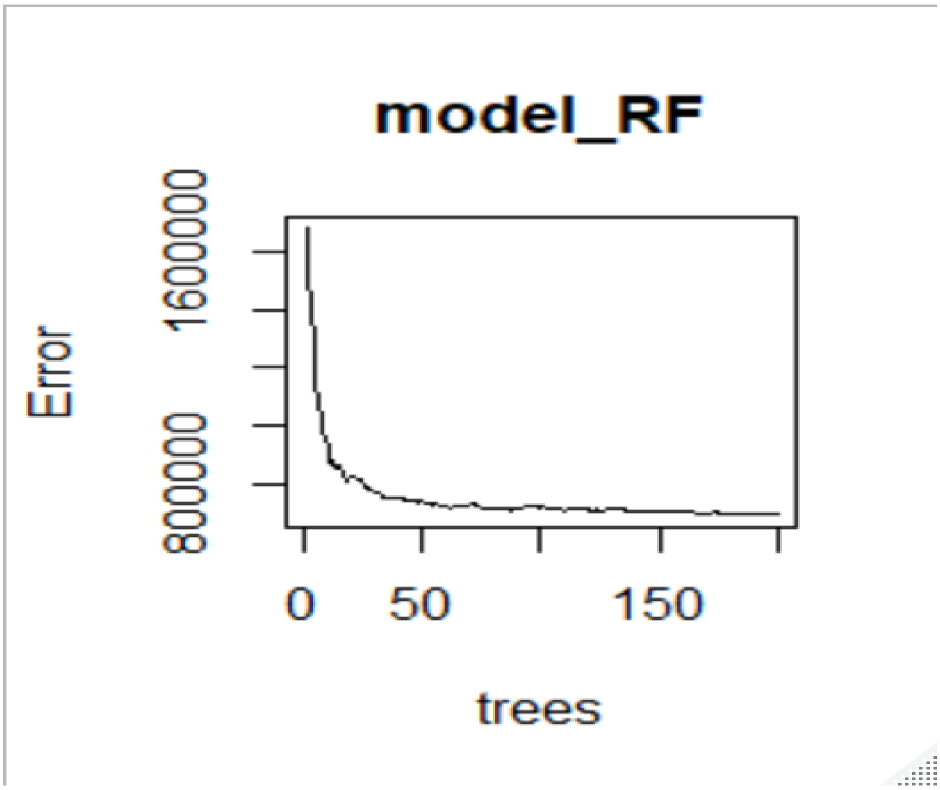
Table 2.8: Decision Tree plot



2.2.3 Random Forest

```
> model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)
>
> predict_RF = predict(model_RF, test[,-12])
> predict_RF
       1        2        3        5       11       12       14       16 
2323.695 2468.643 1785.763 2082.783 1740.501 1715.639 2007.872 1977.994 
      29       33       34       45       47       51       59       64 
1690.118 2134.616 1876.356 2628.353 2418.973 2170.187 2421.479 2440.629 
      69       82       87       88       89       91       93      101 
2524.818 2692.034 2823.906 3098.882 2907.529 2812.077 3470.137 4131.685 
     108      123      127      133      136      137      142      147 
4143.128 4176.832 4676.765 3999.282 4511.369 4281.241 4783.147 4663.816 
     151      161      163      172      173      174      177      178 
```

Table 2.9



2.2.4 Linear Regression

```
> predict_LR
        7        15        25        28        34        40        43 
 926.7891 1496.0479 1086.5167  521.0595 1538.5854  986.8367 1645.6495 
       51        54        56        64        69        74        78 
1914.0308 2204.0104 1132.7494 2015.4172 1656.9729 2101.3222 3132.1761 
       81        83        84        86        90        93        96 
3413.6573 1931.8333 3071.5969 2163.2749  805.2645 3496.8268 3774.7736 
      102       107       112       118       123       124       126 
3093.1229 3763.6615 2366.9700 3316.0431 3956.5056 3220.1329 4452.1042 
      131       135       137       138       142       146       147 
4962.8672 3764.9955 3968.5270 3615.1892 4181.8705 5219.3303 4808.6915 
```

Table 2.10

Chapter 3

Conclusion

3.1 Model Evaluation

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 Wine Data, the latter two, 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.

3.1.1 Mean Absolute Error (MAE)

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.

```
> MAPE = function(y, yhat){
+
+   mean(abs((y - yhat)/y))*100
+
+ }
>
> MAPE(test[,12], predict_DT)
[1] 14.55029

>
>
>
> MAPE(test[,12], predict_RF)
[1] 13.23267

>
>
>
> MAPE(test_lr[,64],  predict_LR)
[1] 18.75454
```

3.2 Model Selection

Least is second one using Random forest so we will use that for prediction.

Appendix A - R code

```
rm(list=ls(all=T))

setwd("C:/Users/ASUS/Desktop/Edwisor Training/Project-2_Bike rental prediction/R code")

#Sample to install a package
#install.packages('ggplot2')

x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",

      "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','fastDummies')

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

rm(x)

#Load CSV
bike_data = read.csv("day.csv", header = T, na.strings = c(" ", "", "NA"))

#Load data to another dataframe to avoid modification in original loaded data
ana_data=bike_data

######Exploratory Data Analysis##################
#Taking a glance through the data and it's feature types
#str(ana_data)

#dim(ana_data)
#We have 16 features and 731 obervations
#taking a glance at data
#head(ana_data)
#As we can see lot of variables are actually should be categorical but are having integer type
#So we will need to convert them to appropriate type
```

```
#Season
#
#Data type conversion

ana_data$season=as.factor(ana_data$season)

ana_data$mnth=as.factor(ana_data$mnth)

ana_data$yr=as.factor(ana_data$yr)

ana_data$holiday=as.factor(ana_data$holiday)

ana_data$weekday=as.factor(ana_data$weekday)

ana_data$workingday=as.factor(ana_data$workingday)

ana_data$weathersit=as.factor(ana_data$weathersit)

#Removing unnessary features from dataset as they are of no use
#for us
#As per discription about features
#instant: Record index
#casual: count of casual users
#registered: count of registered user
ana_data=subset(ana_data,select = -c(instant,casual,registered))

#converting dates
d1=unique(ana_data$dteday)

#str(d1)

df=data.frame(d1)

ana_data$dteday=as.Date(df$d1,format="%Y-%m-%d")

#has been converted to date format
#Let's have a look
```

```
str(ana_data$dteday)

#head(df)
#changing to categorical
df$d1=as.Date(df$d1,format="%Y-%m-%d")

ana_data$dteday=format(as.Date(df$d1,format="%Y-%m-%d"), "%d")

#str(ana_data$dteday)

ana_data$dteday=as.factor(ana_data$dteday)



#All required features have been converted now ...let's recheck
#str(ana_data)

#Successful conversion


###Missing Values Analysis################################################



# 1. checking for missing value

colSums(sapply(ana_data, is.na))
sum(is.na(ana_data)) / (nrow(ana_data) *ncol(ana_data))
#There are no missing values in data

########Checking duplicate values######
cat("The number of duplicated rows are", nrow(ana_data) - nrow(unique(ana_data)))
#There are no duplicate rows present in the dataset
```

##############Outlier Analysis##########

```
# 1.BoxPlots - Distribution and Outlier Check
#Check numeric data first
numeric_index = sapply(ana_data,is.numeric)
numeric_data = ana_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(ana_data))+

           stat_boxplot(geom = "errorbar", width = 0.5) +

           geom_boxplot(outlier.colour="red", fill = "blue" ,outlier.shape=18,

                        outlier.size=1, notch=FALSE) +

           theme(legend.position="bottom")+

           labs(y=cnames[i],x="count")+

           ggtitle(paste("Count Box plot for",cnames[i])))

}



gridExtra::grid.arrange(gn1,gn2,ncol=3)

gridExtra::grid.arrange(gn3,gn4,ncol=2)

#we can see that some of the outliers are there (in red colour)
```

```
#####Outlier analysis and treatment for Humidity#####
#Detecting Outlier on the boxplot

outlier_values <- boxplot.stats(ana_data$hum)$out
boxplot(ana_data$hum, main="Humidity", boxwex=0.1)
mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)

###Cook's distance approach
#select feature
#Coefficient
mean(ana_data$windspeed)
mod <- lm(hum ~ ., data=ana_data)

#calculate cook's distance - It computes the influence
#exerted by each data point (row) on the predicted outcome.
#cook's distance greater than 4 times
#the mean may be classified as influential.

cooksd <- cooks.distance(mod)
#cooksd
#Plot
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd, na.rm=T),names(cooksd),""),
col="red")  # add labels

#finding Influence
influential <- as.numeric(names(cooksd)[(cooksd > 4*mean(cooksd, na.rm=T))])

head(ana_data[influential, ])

#The function outlierTest from car package
#gives the most extreme observation based on the given model
#install.packages('car')
library('car')
car::outlierTest(mod)

#here we can see in the output that 69th observation
```

```
#is most influenced

#Treating Outlier by using Capping
x <- ana_data$hum
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]
#Commenting this because outliers removal was affecting Model accuracy
#We can neglect those Outliers because they are having minor impact
#ana_data$hum=x

#Checking outliers now on the plot
#outlier_values <- boxplot.stats(ana_data$hum)$out
#boxplot(ana_data$hum, main="Humidity", boxwex=0.1)
#mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)

#Now there are no Outliers present

#####Outlier analysis and treatment for windspeed#####
#Detecting Outlier on the boxplot

outlier_values <- boxplot.stats(ana_data$windspeed)$out
boxplot(ana_data$windspeed, main="windspeed", boxwex=0.1)
mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)

#We can see 13 outliers there
###Cook's distance approach
#select feature
#Coefficient

mod <- lm(windspeed ~ ., data=ana_data)

#calculate cook's distance - It computes the influence
#exerted by each data point (row) on the predicted outcome.
#Calculating Cook's distance
```

```
cooksd <- cooks.distance(mod)

#Plot
plot(cooksd, pch="*", cex=2, main="Influential Obs by Cooks distance")  # plot cook's distance
abline(h = 4*mean(cooksd, na.rm=T), col="red")  # add cutoff line
text(x=1:length(cooksd)+1, y=cooksd, labels=ifelse(cooksd>4*mean(cooksd, na.rm=T),names(cooksd),""),
col="red")  # add labels

#finding Influence
influential <- as.numeric(names(cooksd)[(cooksd > 4*mean(cooksd, na.rm=T))])
#head(ana_data[influential, ])

#The function outlierTest from car package
#gives the most extreme observation based on the given model
#install.packages('car')
#library('car')
car::outlierTest(mod)

#here we can see in the output that 595th observation
#is most influenced

#Treating Outlier by using Capping
x <- ana_data$hum
qnt <- quantile(x, probs=c(.25, .75), na.rm = T)
caps <- quantile(x, probs=c(.05, .95), na.rm = T)
H <- 1.5 * IQR(x, na.rm = T)
x[x < (qnt[1] - H)] <- caps[1]
x[x > (qnt[2] + H)] <- caps[2]

#Commenting this because outliers removal was affecting Model accuracy
#We can neglect those Outliers because they are having minor impact
#ana_data$windspeed=x

#Checking outliers now on the plot
#outlier_values <- boxplot.stats(ana_data$windspeed)$out
#boxplot(ana_data$hum, main="windspeed", boxwex=0.1)
#mtext(paste("Outliers: ", paste(outlier_values, collapse=", ")), cex=0.6)
```

```
#Now there are no Outliers present

####################Feature Selection################

#Finding features with high correlation
#colnames(data)
rel = cor(ana_data[,numeric_index])
#rel
#Here we can summarize correlation of numeric data with count
#feature


## Correlation Plot

corrgram(ana_data[,numeric_index], order = F,

    upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")


## Dimension Reduction####


#Remove atemp
ana_data = subset(ana_data,select = -c(atemp))


####### Models ########################
```

```
#####Data shuffeling and train test split#####
rmExcept("ana_data")

##We can use below lines as well:
#head(ana_data)

#shuffle_index = sample(1:nrow(ana_data))

#head(shuffle_index)

#ana_data <- ana_data[shuffle_index, ]

#head(ana_data)

#########Train_test split#########
#install.packages("rpart.plot")
library(rpart.plot)



train_index = sample(1:nrow(ana_data), 0.8 * nrow(ana_data))

train = ana_data[train_index,]

test = ana_data[-train_index,]

#Models

###########Decision tree regression  #################
#Anova for regression tree
#class for classification tree
#rpart is the function to apply Dicision tree

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

predict_DT = predict(fit, test[,-12])
```

```
predict_DT
#test[,-12]

#plot tree
rpart.plot(fit,extra = 101)

#############Random Forest Model##########################

#model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 250)
#model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 300)
#model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 150)
#model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 170)
#model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)
#Perfect eror fall below

model_RF = randomForest(cnt ~ ., train, importance = TRUE, ntree = 200)

predict_RF = predict(model_RF, test[,-12])
predict_RF
plot(model_RF)

################Linear Regression#################



#converting multilevel categorical variable into binary dummy variable

cnames= c("dteday","season","mnth","weekday","weathersit")

data_lr=ana_data[,cnames]

cnt=data.frame(ana_data$cnt)

names(cnt)[1]="cnt"

#creating dummy columns
```

```
data_lr <- fastDummies::dummy_cols(data_lr)

data_lr= subset(data_lr,select = -c(dteday,season,mnth,weekday,weathersit))
#head(data_lr)

#Appending datset after Dummies

d3 = cbind(data_lr,ana_data)
#head(d3)

#Droping some features

d3= subset(d3,select = -c(dteday,season,mnth,weekday,weathersit,cnt))

#d3
data_lr=cbind(d3,cnt)




#dividind data into test and train

train_index = sample(1:nrow(data_lr), 0.8 * nrow(data_lr))

train_lr = data_lr[train_index,]

test_lr = data_lr[-train_index,]



#Linear regression model making

model_lm = lm(cnt ~., data = train_lr)

predict_LR = predict(model_lm,test_lr[,-64])
```

```
predict_LR

#summary(model_lm)

################evaluating MApe value##############

MAPE = function(y, yhat){
  mean(abs((y - yhat)/y))*100
}

MAPE(test[,12], predict_DT)

MAPE(test[,12], predict_RF)

MAPE(test_lr[,64],  predict_LR)

#14.55029
#13.23267
#18.07619

#so least is second one using Random forest , so we will use that
#for prediction
```

```
##########extacting predicted values output from Random forest model######################

results <- data.frame(test, pred_cnt = predict_RF)

write.csv(results, file = 'RF output R .csv', row.names = FALSE, quote=FALSE)

rm(list=ls(all=T))
```

Appendix B - Python code

```
##########################################################################################
################################

#################################################   Bike Rental Prediction
############################################

##########################################################################################
################################

############importing Libraries##########

import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
import datetime as dt
```

```
#from sklearn.cross_validation import train_test_split
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
import statsmodels.api as sm
from sklearn.ensemble import RandomForestRegressor
from  matplotlib import pyplot

############working directory#################
os.chdir("C:/Users/ASUS/Desktop/Edwisor Training/Project-2_Bike rental prediction/Python code")

###########loading file####################
bike_data = pd.read_csv("day.csv")
bike_train = bike_data

############exploratory data analysis######################
####Type Conversion#####

bike_train['season']= bike_train['season'].astype('category')
bike_train['yr']=bike_train['yr'].astype('int')
bike_train['mnth']=bike_train['mnth'].astype('category')
bike_train['holiday']=bike_train['holiday'].astype('int')
bike_train['workingday']=bike_train['workingday'].astype('int')
bike_train['weekday']=bike_train['weekday'].astype('category')
bike_train['weathersit']=bike_train['weathersit'].astype('category')
d1=bike_train['dteday'].copy()
for i in range (0,d1.shape[0]):
    d1[i]=dt.datetime.strptime(d1[i], '%Y-%m-%d').strftime('%d')
bike_train['dteday']=d1
bike_train['dteday']=bike_train['dteday'].astype('category')

##Dropping columns unnecessary####

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

print("Data types :", bike_train.dtypes)

##Check discription####
```

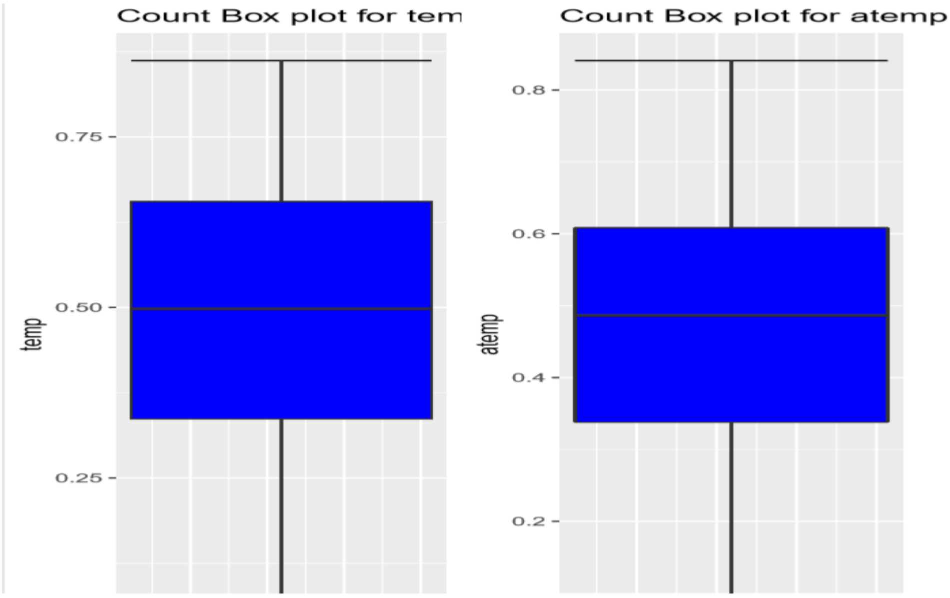
```
print("Description of Data")
print(bike_train.describe())


##We can see that no missing values are there in Data but we will check once##
############Missing value analysis#################

##Checking Null values#

response = bike_train.isnull().values.any()

print("Null :",response)

##There are no missing values ###

#No missing values###

################Outlier Analysis###################

#saving numeric values#

cnames=["temp","atemp","hum","windspeed",]

#ploting boxplotto visualize outliers#

plt.boxplot(bike_train['temp'])
plt.show()

plt.boxplot(bike_train['atemp'])
plt.show()

plt.boxplot(bike_train['hum'])
plt.show()


plt.boxplot(bike_train['windspeed'])
plt.show()
```

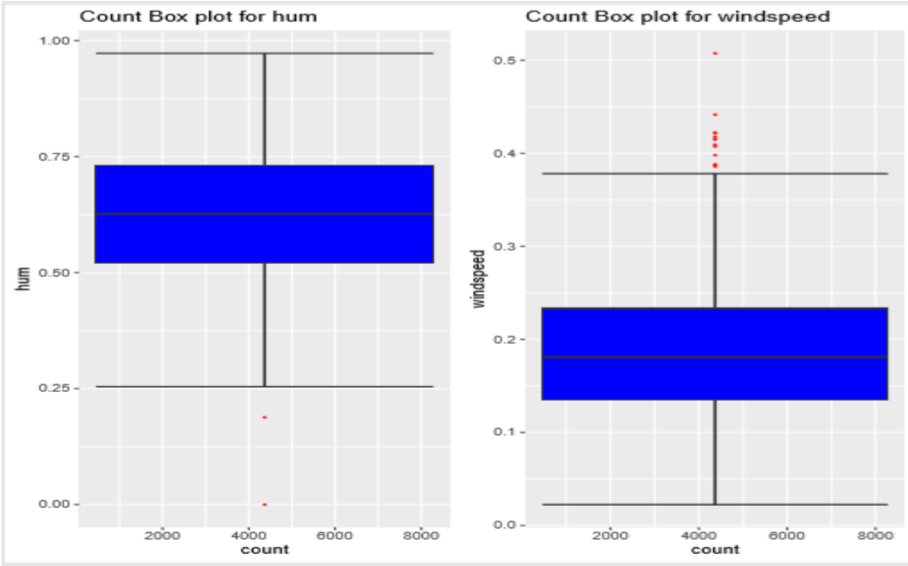
```
#############Feature Selection ##################

df_corr = bike_train

#Set the width and hieght of the plot

f, ax = plt.subplots(figsize=(7, 5))

#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)
plt.show()

#droping corelated variable
bike_train = bike_train.drop(['atemp'], axis=1)

###########Modeling ###############################

#dividing data into train and test
train, test = train_test_split(bike_train, test_size=0.2)
```

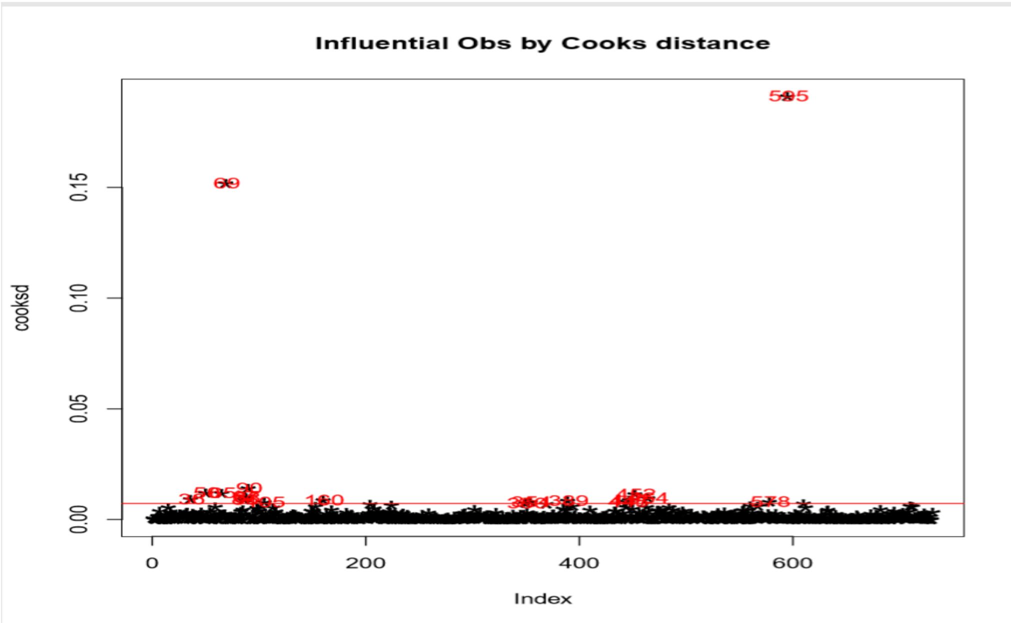
```
######c50#######
fit_DT = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:11], train.iloc[:,11])
predictions_DT = fit_DT.predict(test.iloc[:,0:11])

#random forest
RFmodel = RandomForestRegressor(n_estimators = 200).fit(train.iloc[:,0:11], train.iloc[:,11])
RF_Predictions = RFmodel.predict(test.iloc[:,0:11])


#linear regression
#creating dummy variable
data_lr=bike_train.copy()
cat_names = ["season", "dteday", "weathersit", "mnth","weekday"]
for i in cat_names:
    temp = pd.get_dummies(data_lr[i], prefix = i)
    data_lr = data_lr.join(temp)
fields_to_drop = ['dteday', 'season', 'weathersit', 'weekday', 'mnth','cnt']
data_lr = data_lr.drop(fields_to_drop, axis=1)
data_lr=data_lr.join(bike_train['cnt'])

trainlr, testlr = train_test_split(data_lr, test_size=0.2)
model = sm.OLS(trainlr.iloc[:,63], trainlr.iloc[:,0:63]).fit()
predictions_LR = model.predict(testlr.iloc[:,0:63])


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


#MAPE for decision tree regression
x=MAPE(test.iloc[:,11], predictions_DT)

#MAPE for random forest regression
y=MAPE(test.iloc[:,11],RF_Predictions)
```

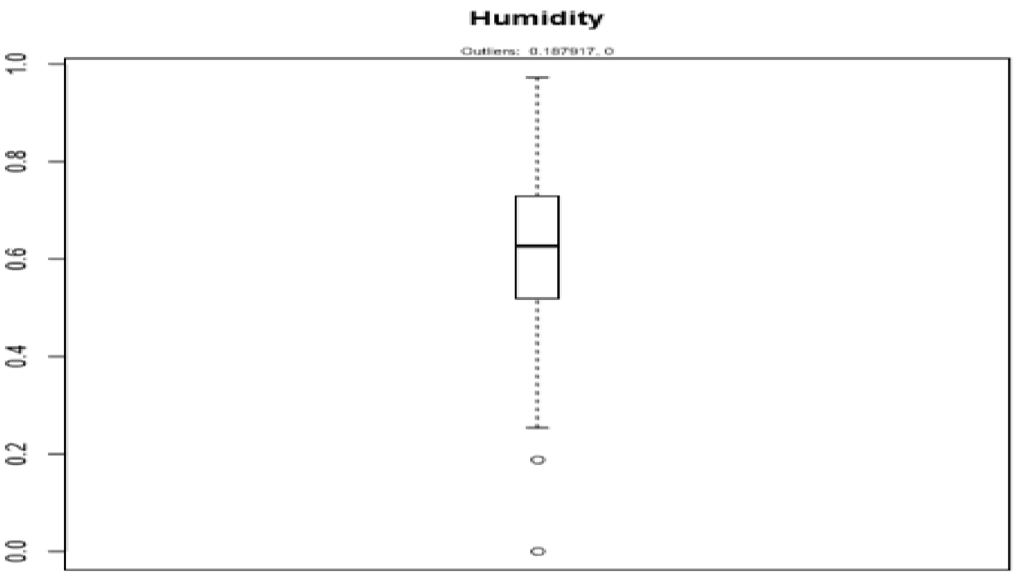
```
#MAPE for linear regression
z=MAPE(testlr.iloc[:,63], predictions_LR)

print("MAPE Dicision tree=",x)
print("MAPE Random forest=",y)
print("MAPE Linear Refression=",z)


result=pd.DataFrame(test.iloc[:,0:11])
result['pred_cnt'] = (RF_Predictions)

result.to_csv("Random forest output python.csv",index=False)
```

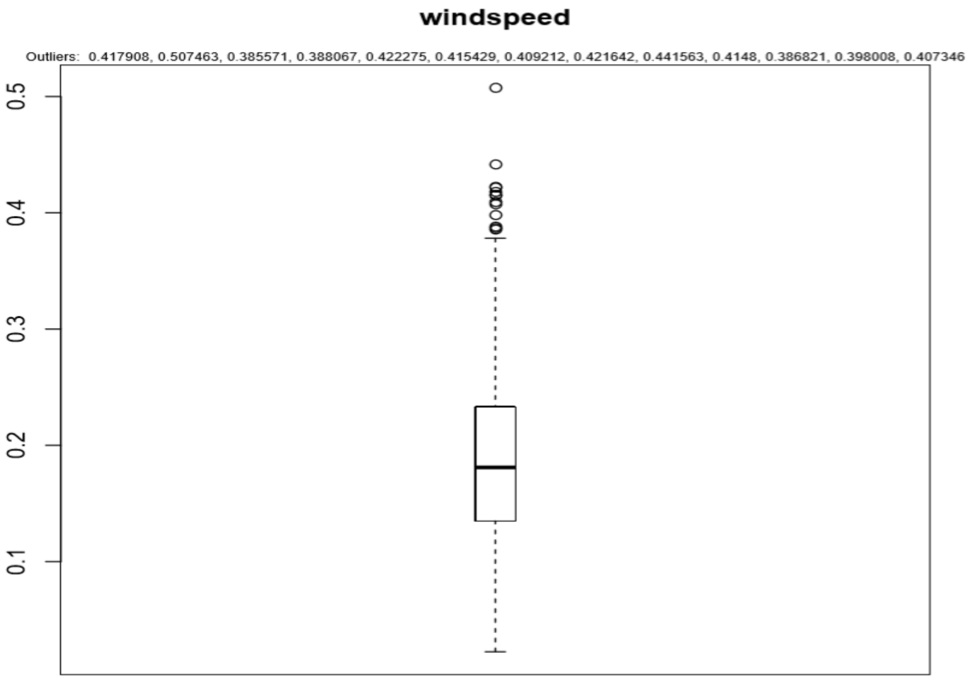
Plots

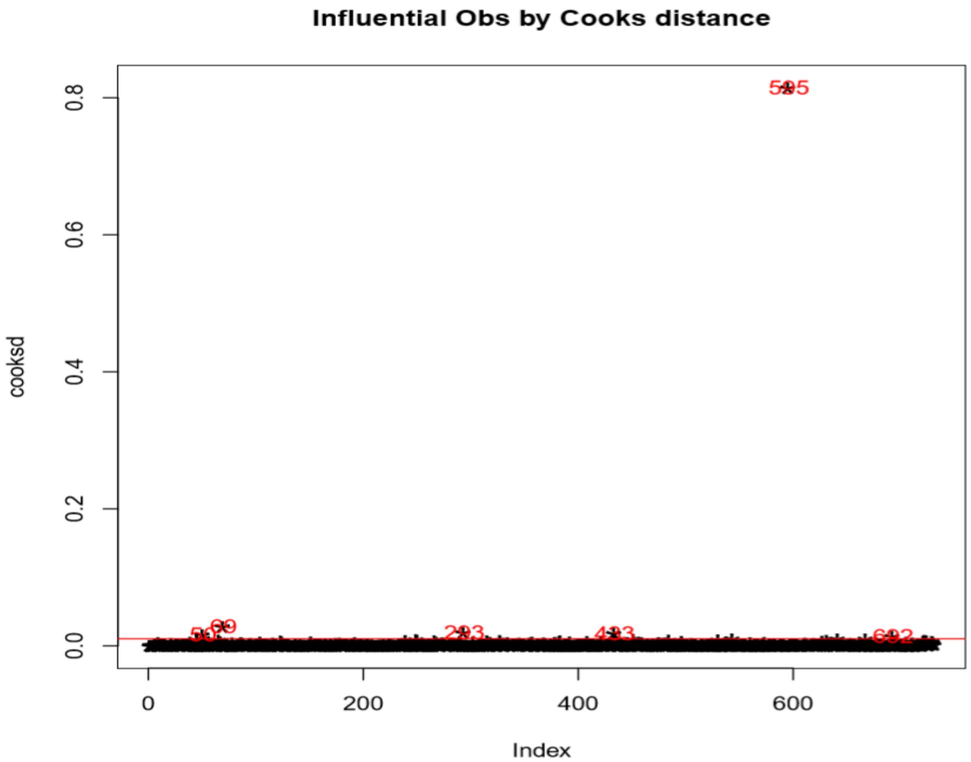


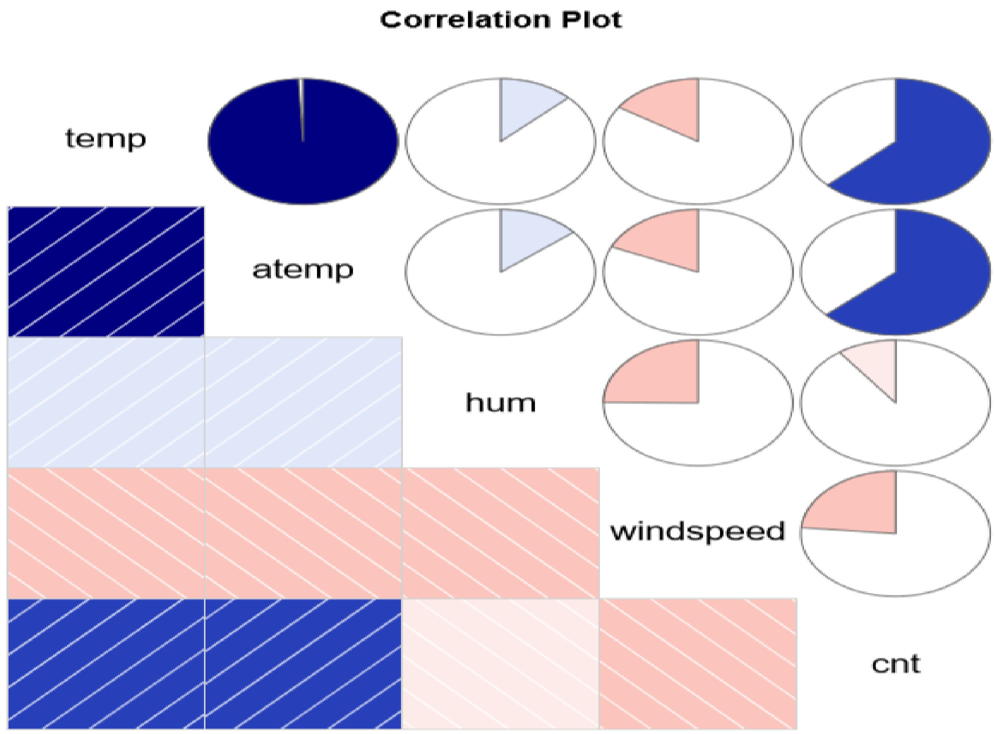

Count Box plot for hum
hum
count
Count Box plot for windspeed
windspeed
count

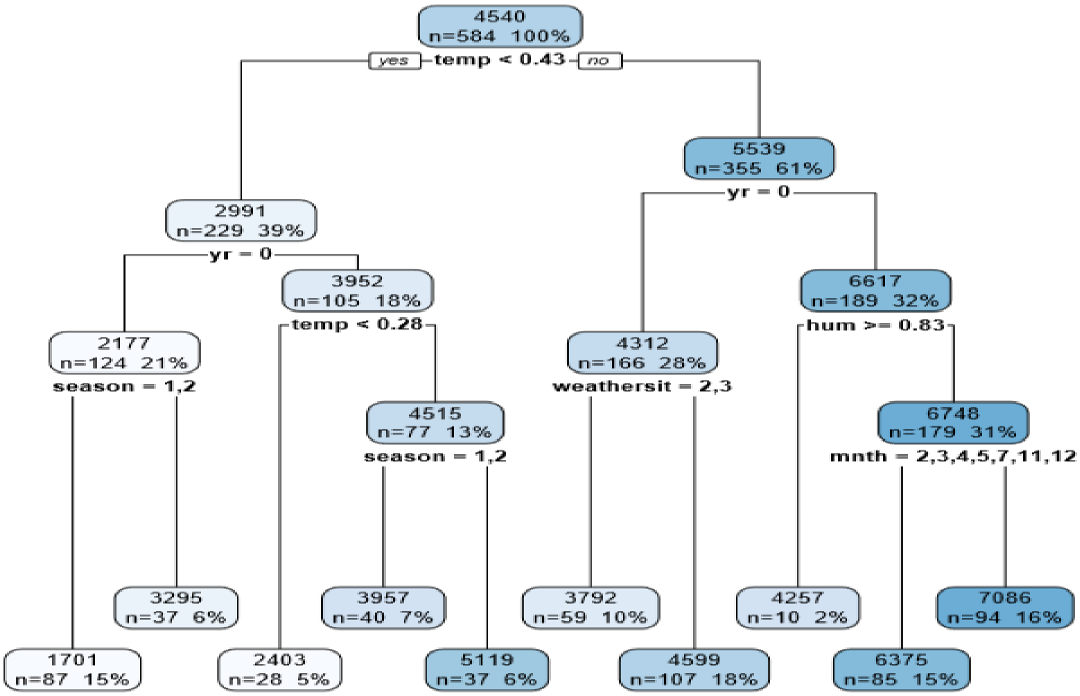

Influential Obs by Cooks distance
cooksd
Index

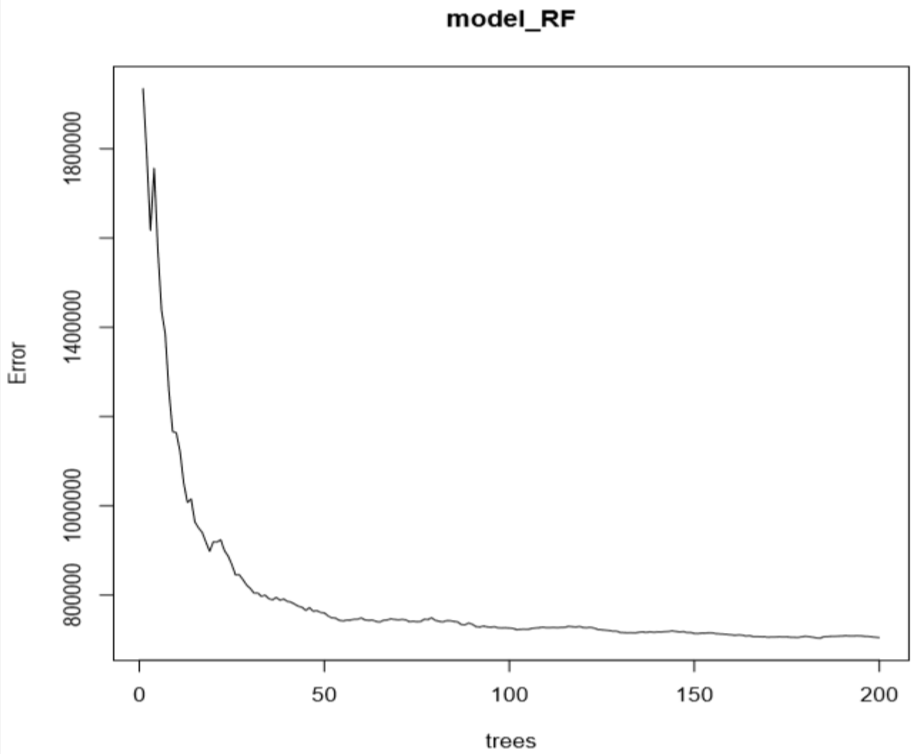

Humidity
Outliers: 0.187917, 0
1.0
0.8
0.6
0.4
0.2
0.0


windspeed
Outliers: 0.417908, 0.507463, 0.385571, 0.388067, 0.422275, 0.415429, 0.409212, 0.421642, 0.441563, 0.4148, 0.386821, 0.398008, 0.407346
0.5
0.4
0.3
0.2
0.1


Influential Obs by Cooks distance
cooksd
0.0
0.2
0.4
0.6
0.8
0
200
400
600
Index


Correlation Plot
temp
atemp
hum
windspeed
cnt


4540
n=584 100%
yes
temp < 0.43
no
5539
n=355 61%
yr = 0
2991
n=229 39%
yr = 0
3952
n=105 18%
temp < 0.28
6617
n=189 32%
hum >= 0.83
2177
n=124 21%
season = 1,2
4312
n=166 28%
weathersit = 2,3
4515
n=77 13%
season = 1,2
6748
n=179 31%
mnth = 2,3,4,5,7,11,12
3295
n=37 6%
3957
n=40 7%
3792
n=59 10%
4257
n=10 2%
7086
n=94 16%
1701
n=87 15%
2403
n=28 5%
5119
n=37 6%
4599
n=107 18%
6375
n=85 15%


model_RF
Error
1800000
1400000
1000000
800000
0
50
100
150
200
trees

References

HTTPS://WWW.DATASCIENCECENTRAL.COM/PROFILES/BLOGS/IMPLEMETATION-OF-17-CLASSIFICATION-ALGORITHMS-IN-R

HTTPS://DATA-FLAIR.TRAINING/BLOGS/CLASSIFICATION-IN-R/

HTTP://R-STATISTICS.CO/LINEAR-REGRESSION.HTML

HTTPS://HACKERNOON.COM/CHOOSING-THE-RIGHT-MACHINE-LEARNING-ALGORITHM-68126944CE1F - VERY EFFECTIVE FOR ALGORITHMS SUMMARY