

Predication of bike rental count

Satyam Sharma 03-07-2019



Contents

[Chapter 1 2](#_Toc13006294)

[Introduction 2](#_Toc13006295)

[1.1 Problem Statement 2](#_Toc13006296)

[1.2 Data 2](#_Toc13006297)

[Chapter 2 4](#_Toc13006298)

[Methodology 4](#_Toc13006299)

[2.1 Pre-processing 4](#_Toc13006300)

[Chapter 3 11](#_Toc13006301)

[Conclusion 11](#_Toc13006302)

[3.1 Model Evaluation 11](#_Toc13006303)

[3.2 Model Selection 12](#_Toc13006304)

[Appendix A - R code 13](#_Toc13006305)

[Appendix B - Python code 25](#_Toc13006306)

[Plots 30](#_Toc13006307)

[References 35](#_Toc13006308)

# 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.

**Steps in Data Preprocessing**

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 ...

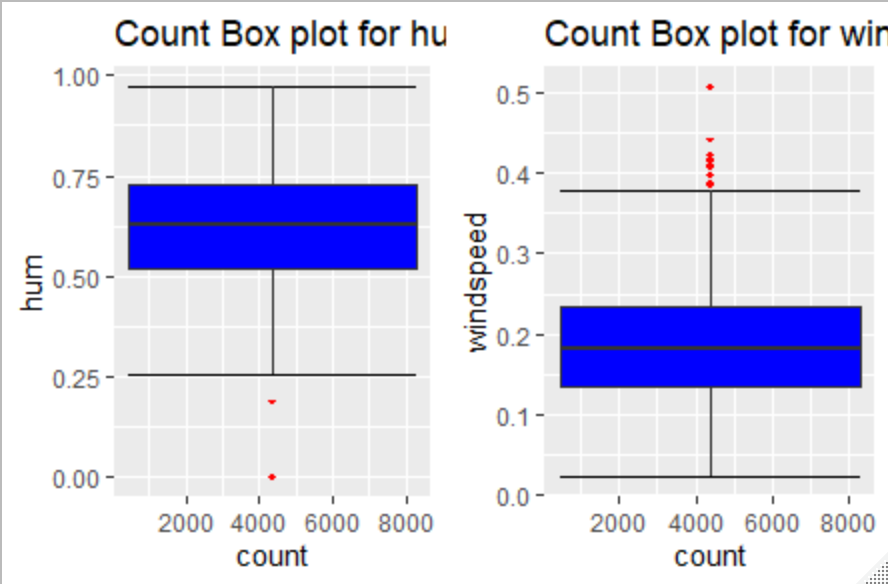
$ windspeed : num 0.16 0.249 0.248 0.16 0.187 ...

$ 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

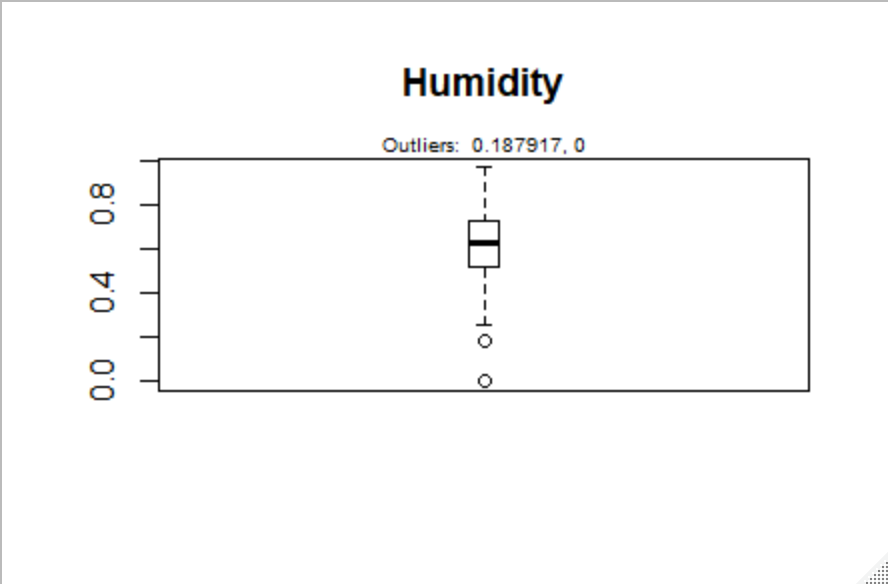
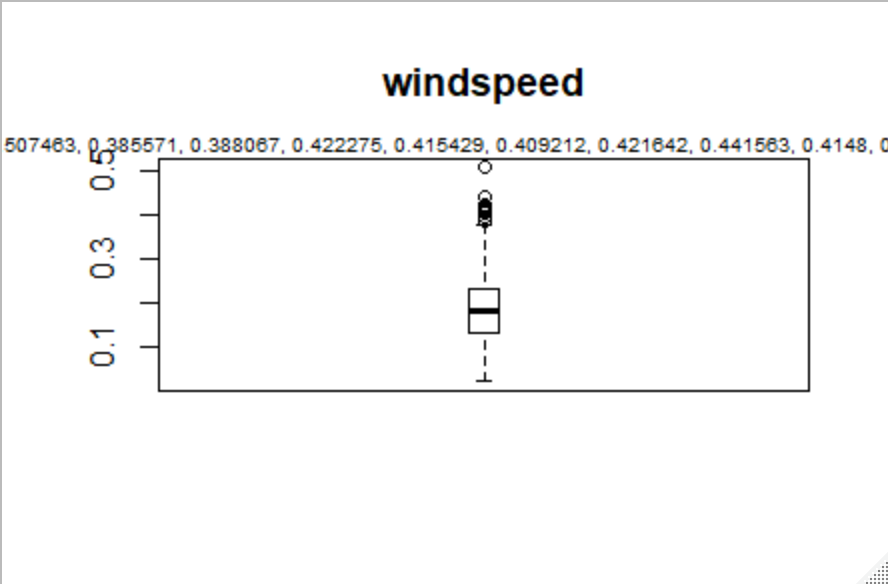


Table 2.4



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

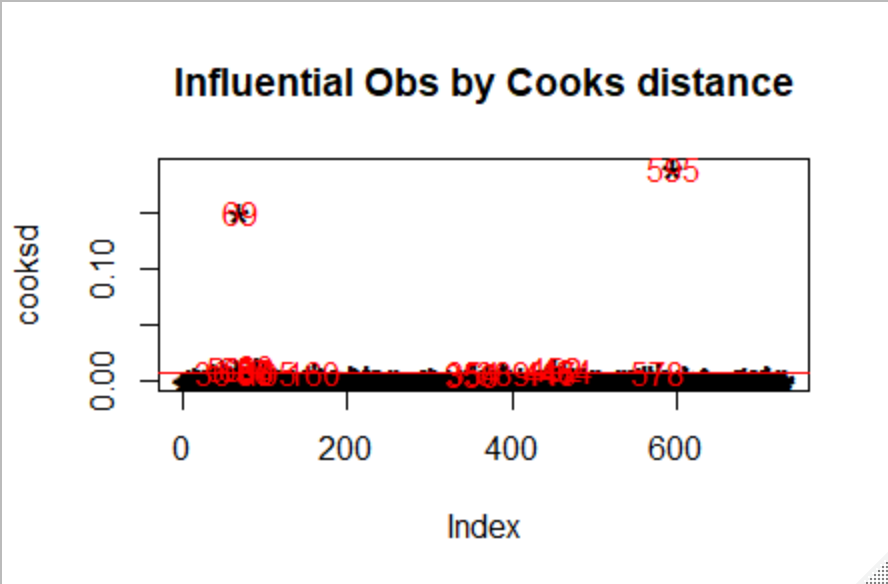
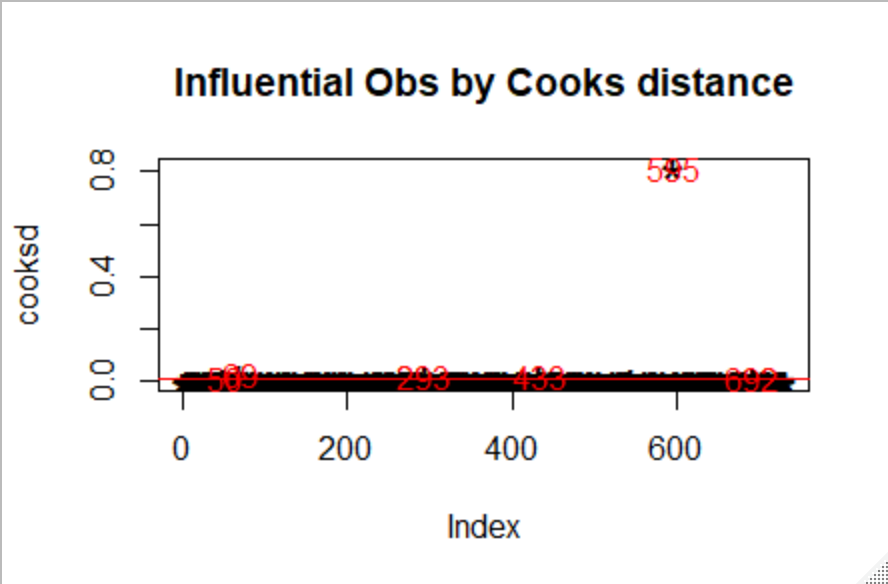


Table 2.6



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

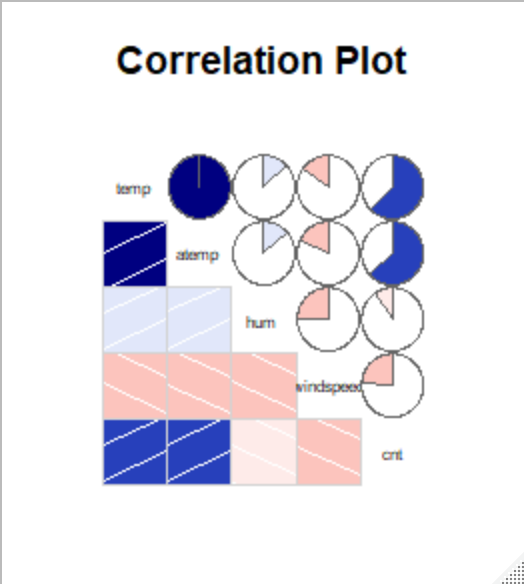
windspeed -0.1579441 -0.1836430 -0.2484891 1.0000000 -0.2345450

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

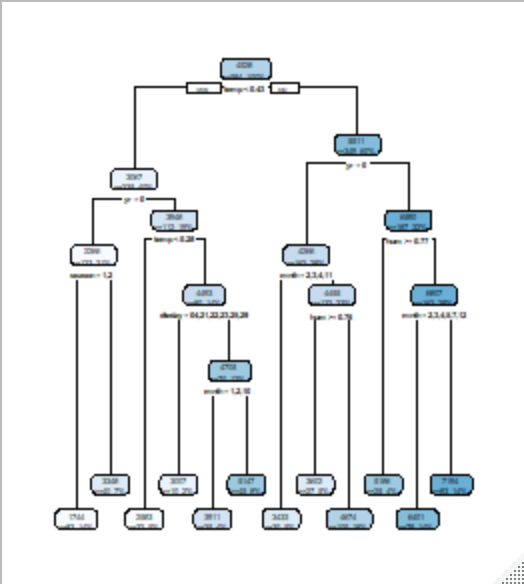
1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494 1744.494

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

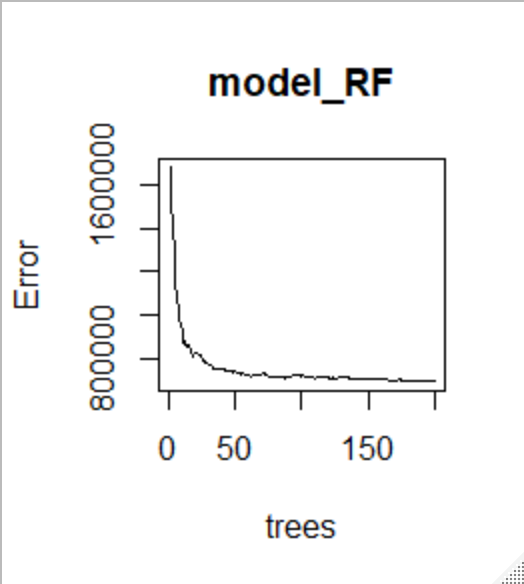
2524.818 2692.034 2823.906 3098.882 2907.529 2812.077 3470.137 4131.685

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)

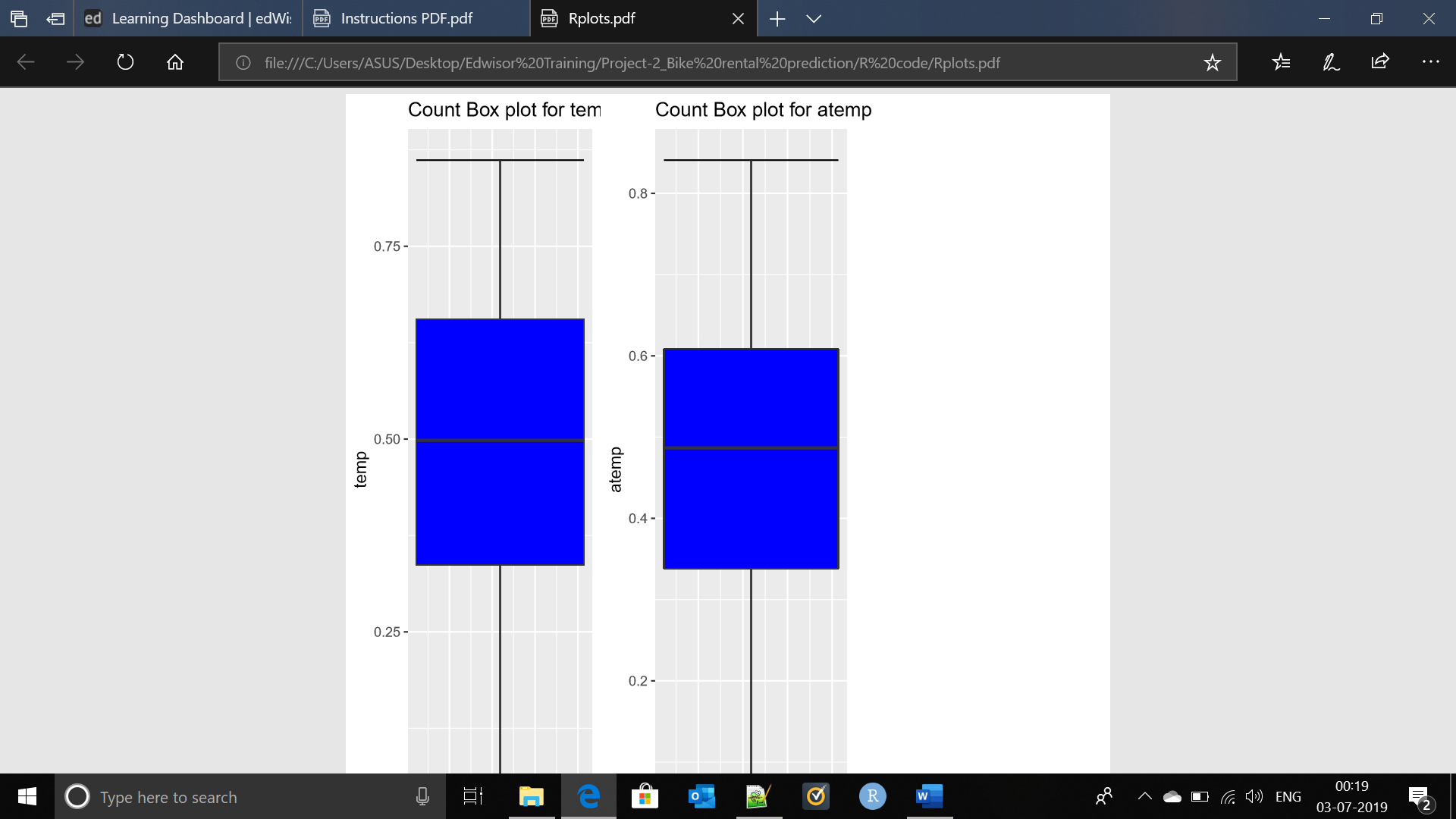
print("MAPE Linear Refression=",z)

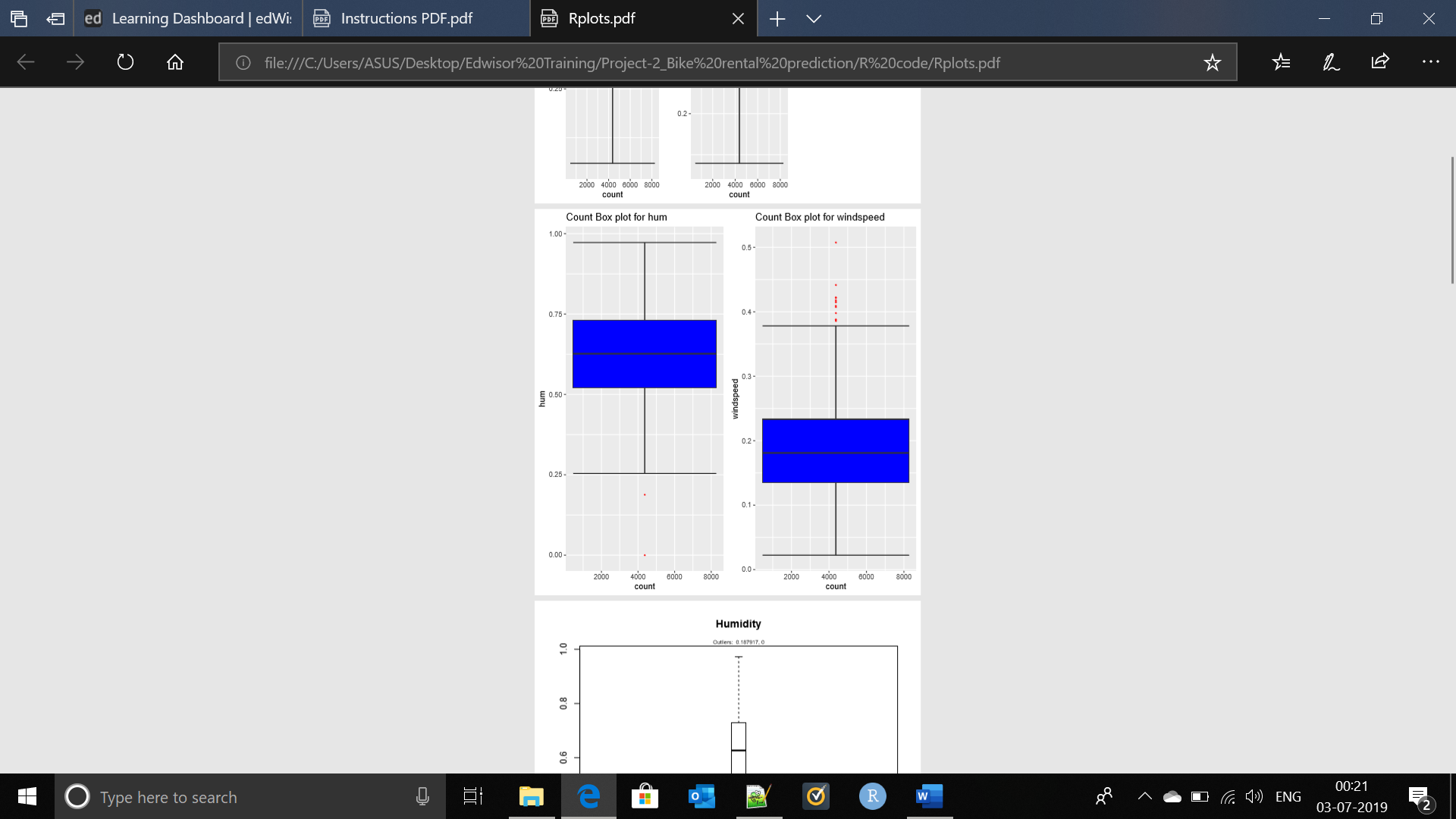
result=pd.DataFrame(test.iloc[:,0:11])

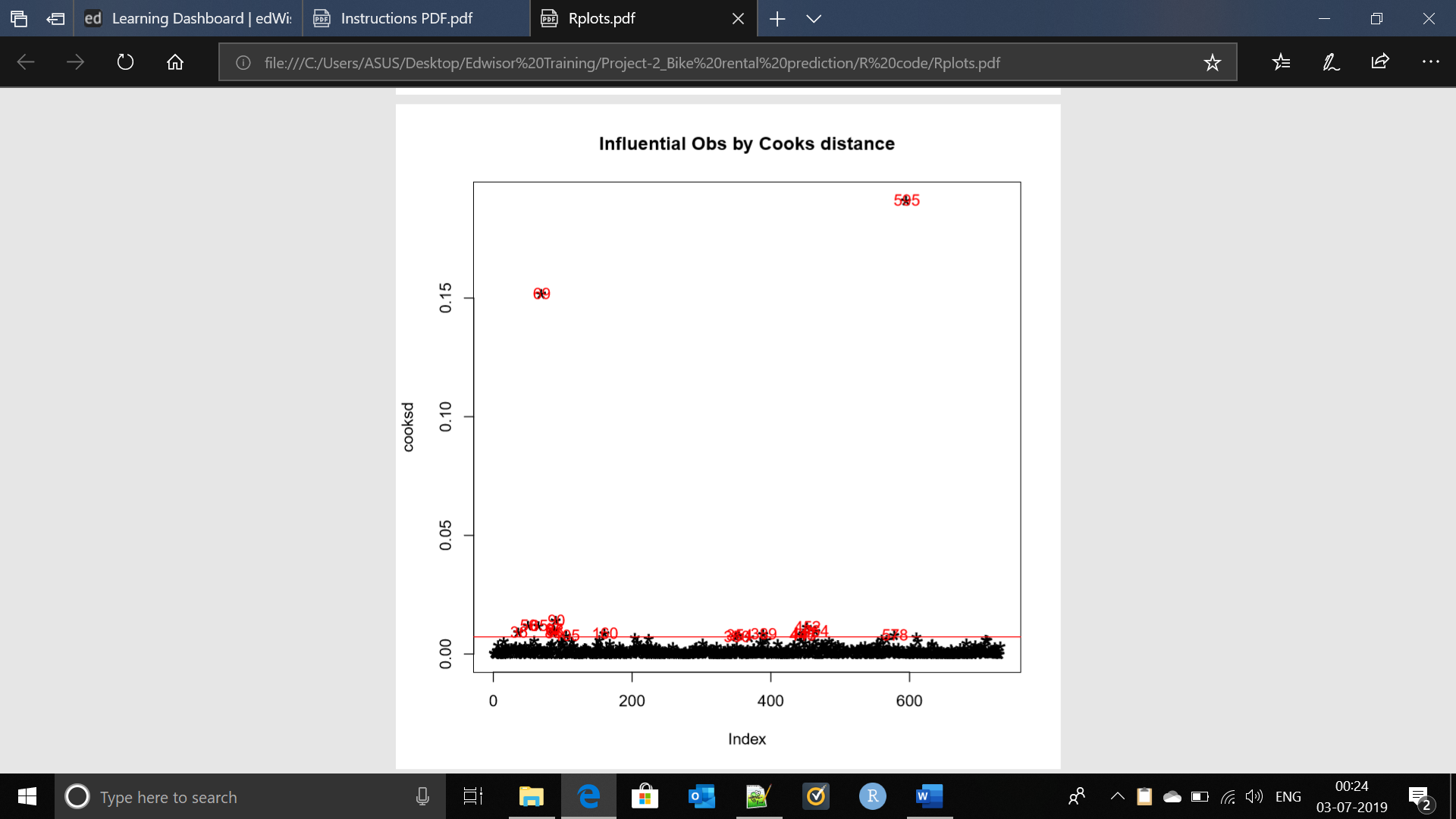
result['pred\_cnt'] = (RF\_Predictions)

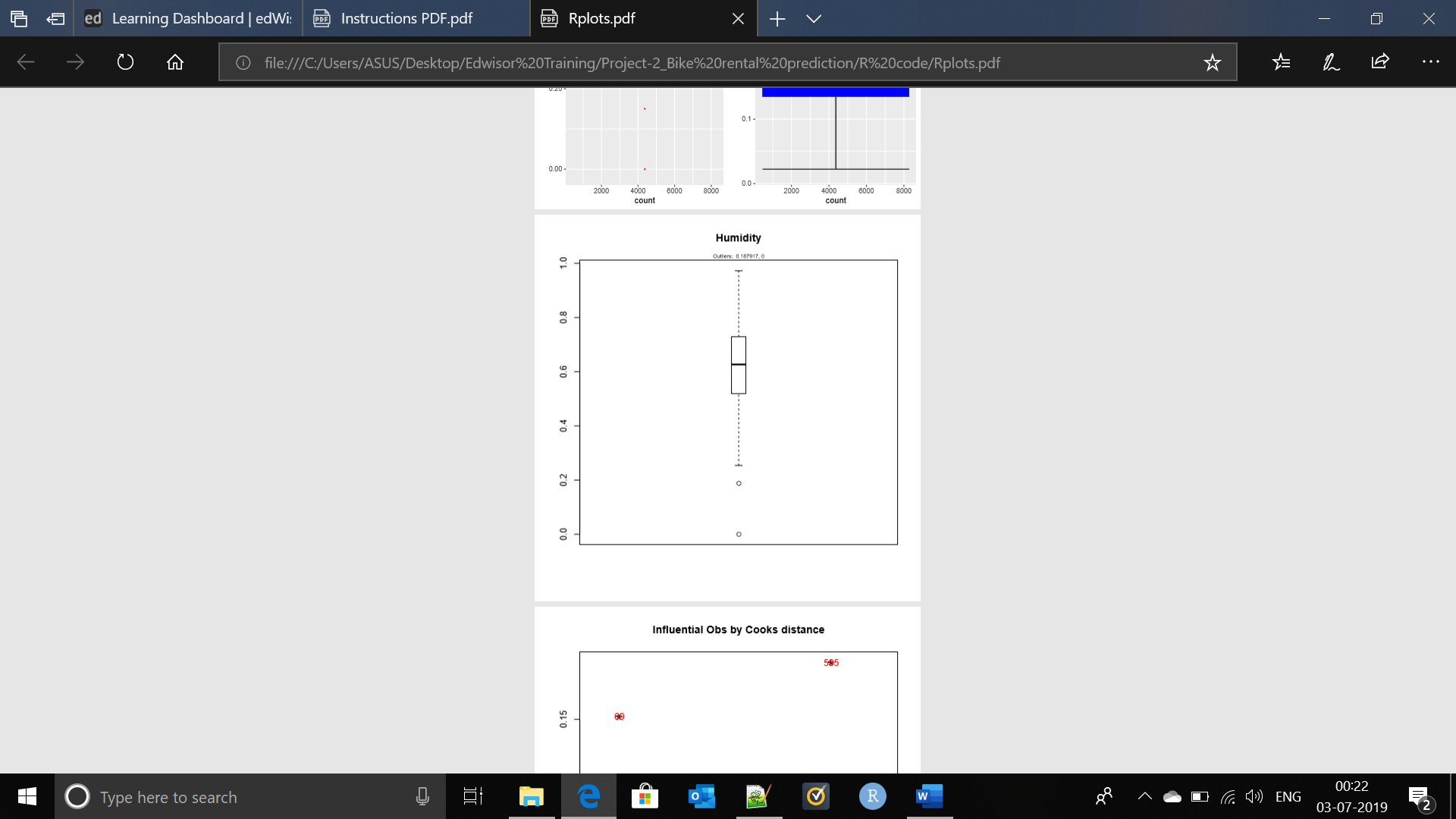
result.to\_csv("Random forest output python.csv",index=False)

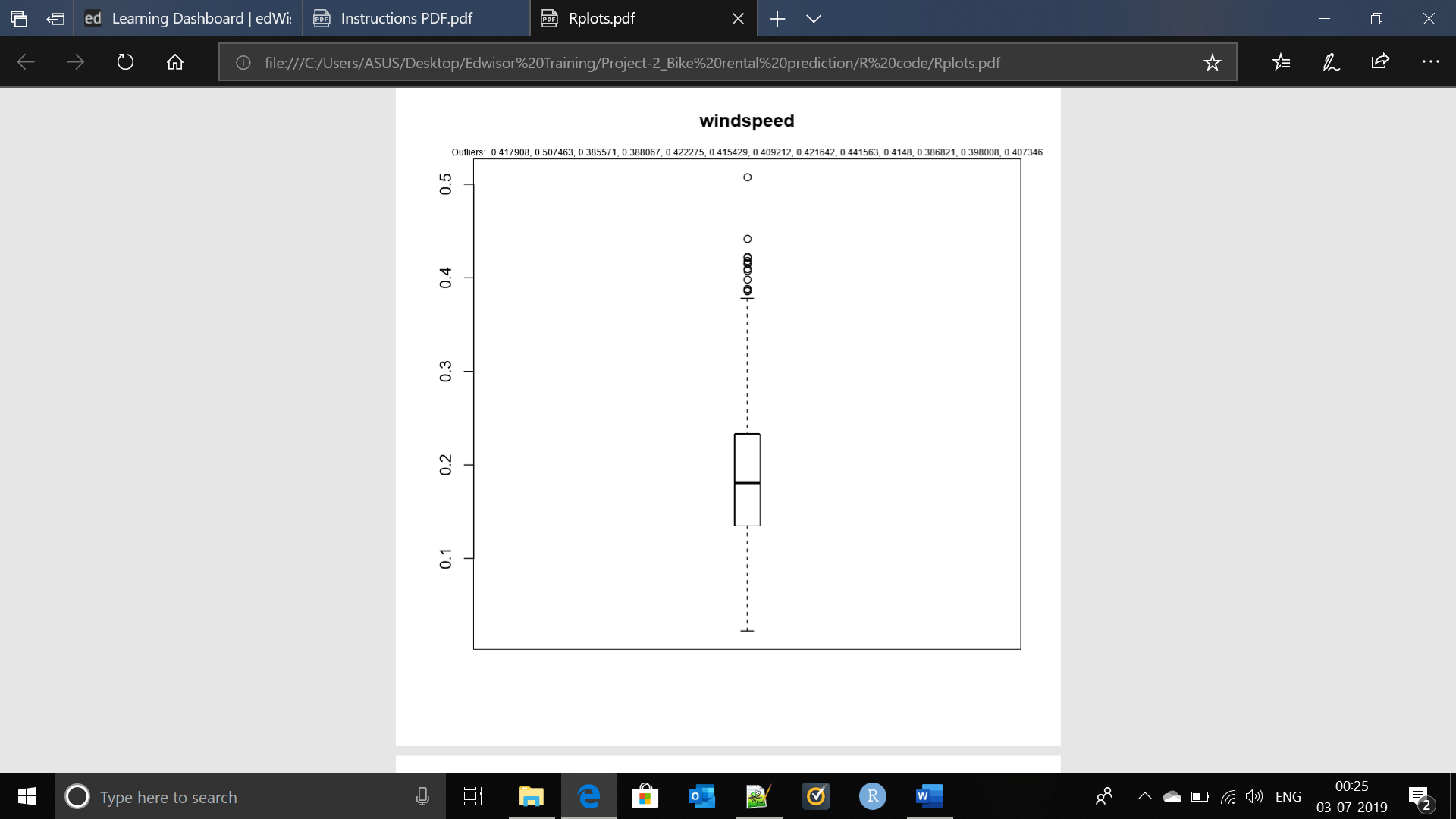
# Plots

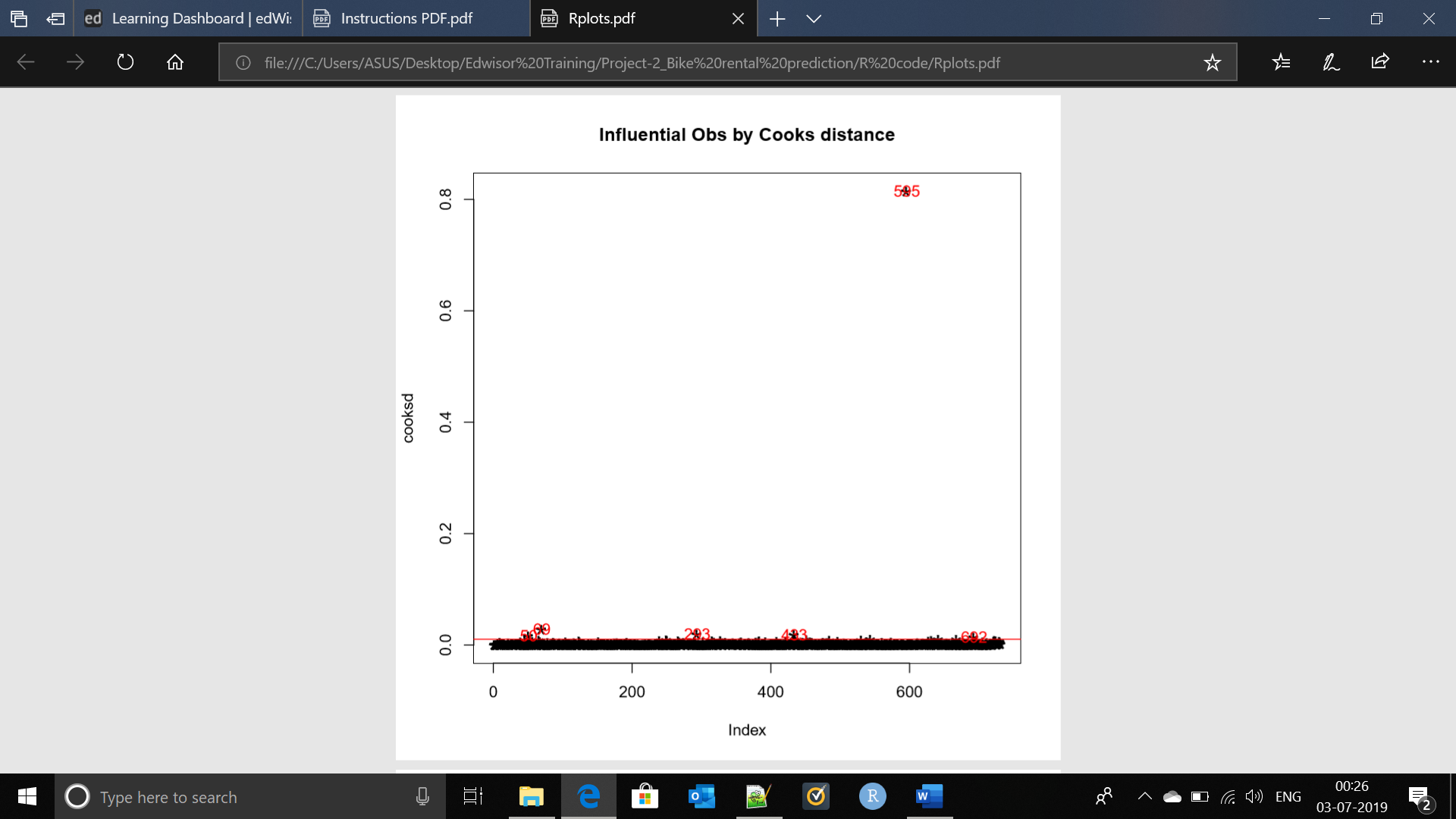


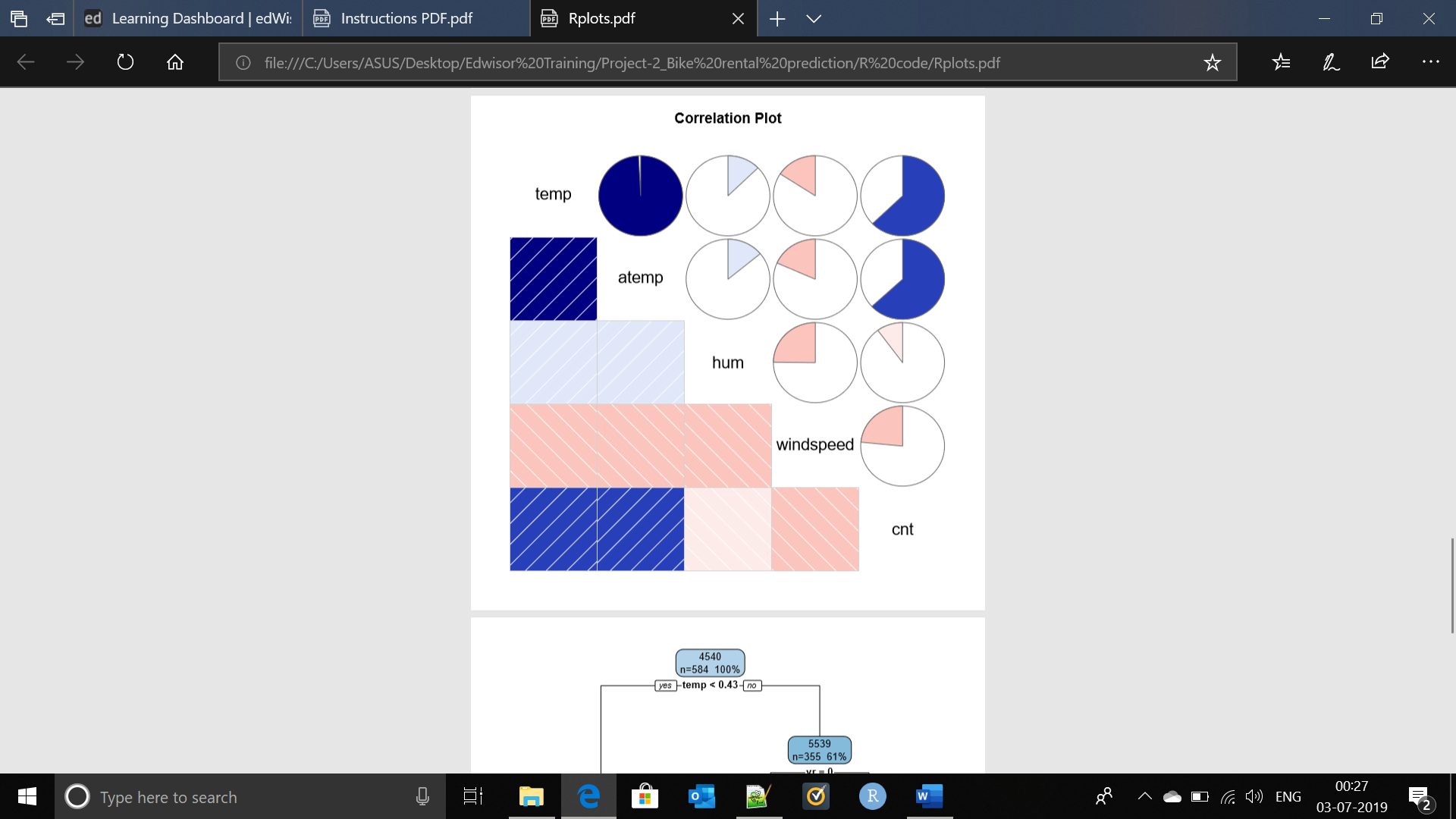


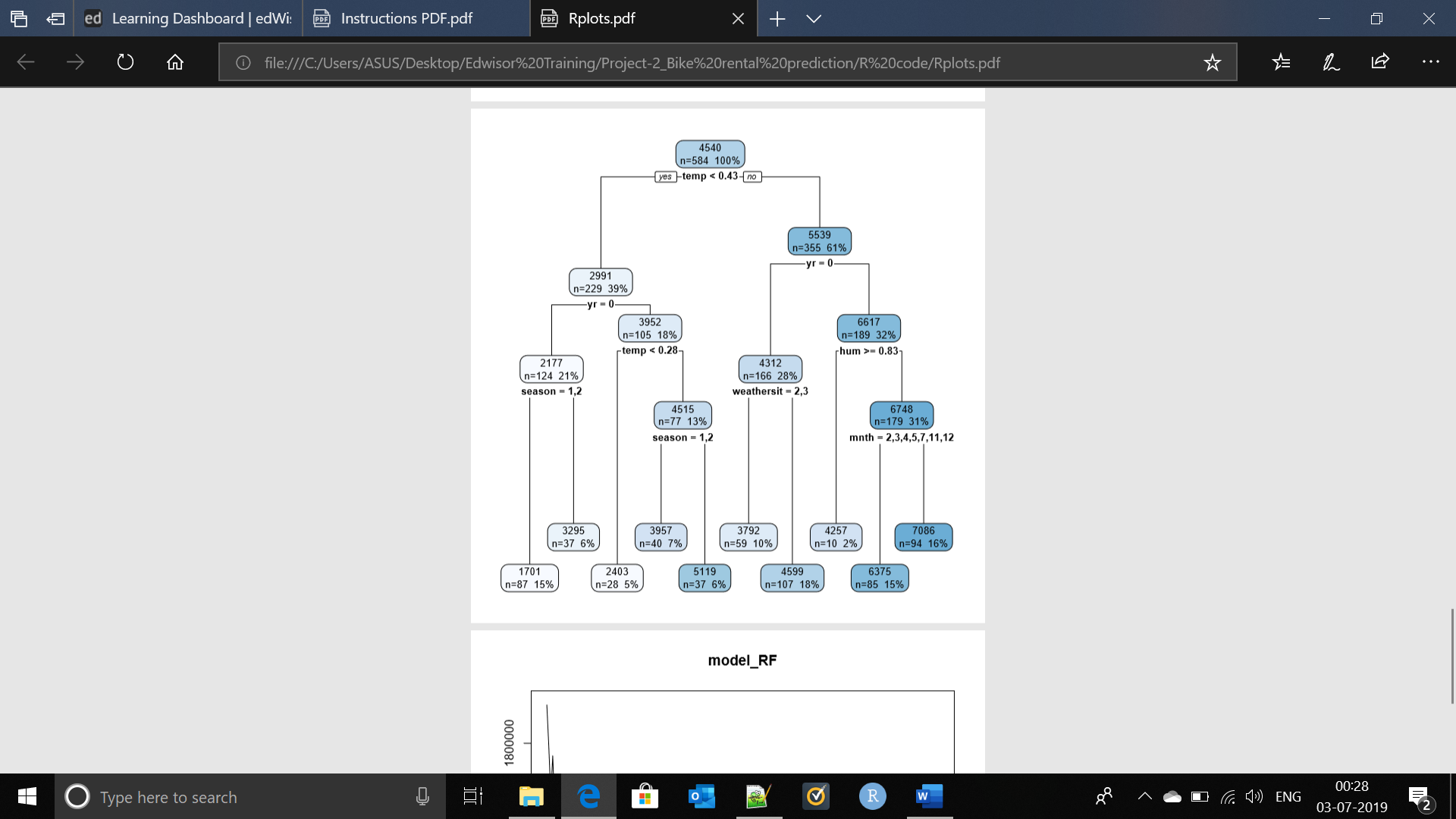


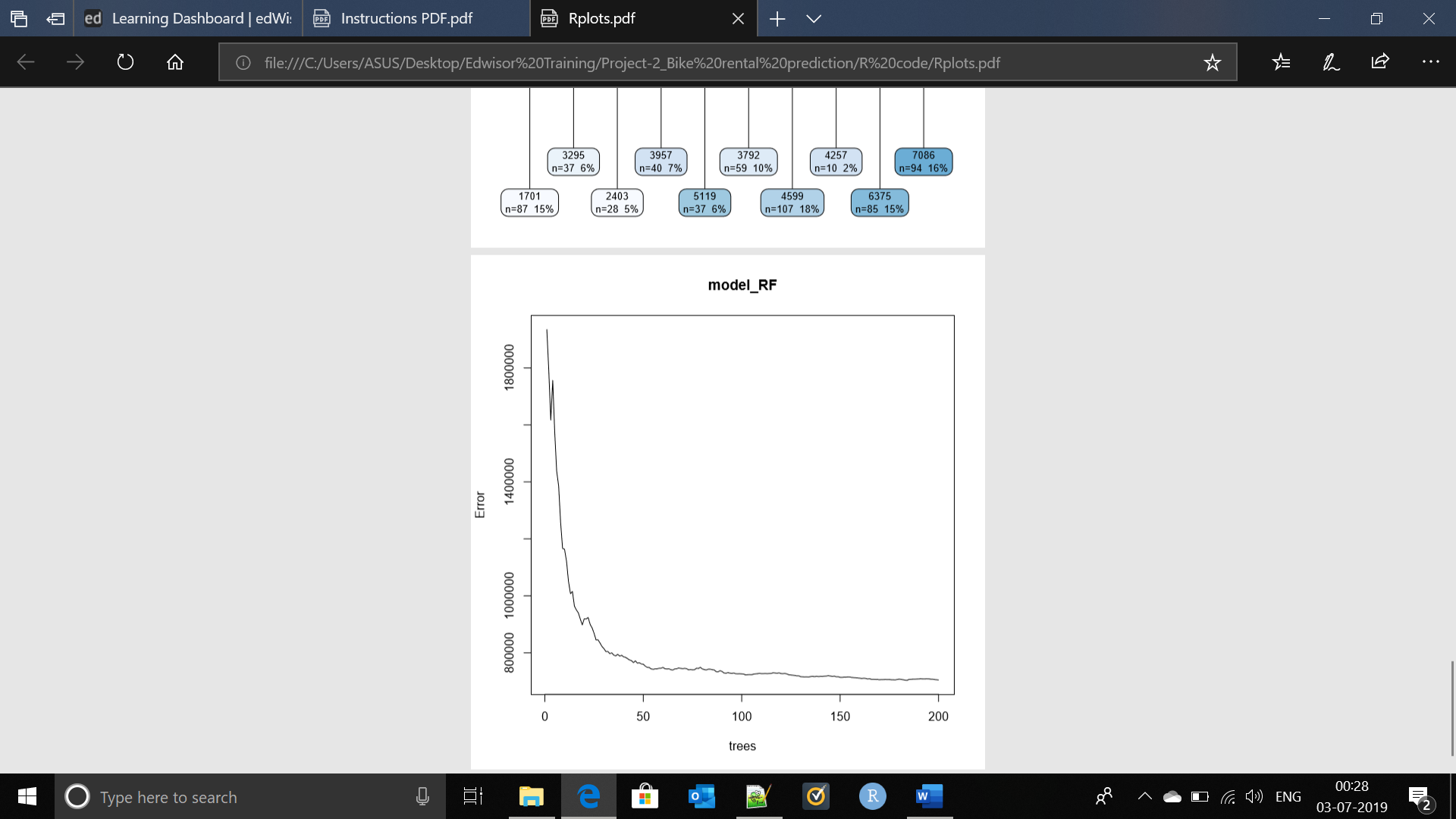












# References

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