BIKE RENTING

BY

Deepak Kulkarni

12/09/2019

TABLE OF CONTENTS

CHAPTER 1

Introduction

1.1 Problem Statement

The objective of the proposed work is to predict the bike rental count which is based on the Seasonal settings like temperature, wind speed, humidity etc and also environmental settings.

1.2 Dataset

The dataset provided has a combination of categorical and numerical data type. The target variable is total number of count of bikes which is a continuous variable.
The dataset provided has all together 16 variables (15 independent and 01 dependent) with 731 observations.
There is one categorical variable dteday and 15 numerical variables. Out of which one is target variable(cnt).

Table1.1: Sample data to predict bike rent

instant	dteday	season	yr	mnth	holiday	weekday
1	01-01-2011	1	0	1	0	6
2	02-01-2011	1	0	1	0	0
3	03-01-2011	1	0	1	0	1
4	04-01-2011	1	0	1	0	2
5	05-01-2011	1	0	1	0	3
6	06-01-2011	1	0	1	0	4
7	07-01-2011	1	0	1	0	5
8	08-01-2011	1	0	1	0	6

workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	2	0.3441	0.3636	0.8058	0.160446	331	654	985
0	2	0.3634	0.3537	0.6960	0.248539	131	670	801
1	1	0.1963	0.1894	0.4372	0.248309	120	1229	1349
1	1	0.2	0.21212	0.59043	0.160296	108	1454	1562
1	1	0.2269	0.22927	0.43695	0.1869	82	1518	1600
1	1	0.2043	0.23320	0.51826	0.089565	88	1518	1606
1	2	0.1965	0.20883	0.49869	0.168726	148	1362	1510
0	2	0.165	0.1622	0.5358	0.266804	68	891	959

As it can be seen from table1.1 the following predictors are used to predict the count (cnt) shown in table 1.2

Table 1.2: Predictor variables

Sl no	Predictor
1	instant
2	dteday
3	Season
4	yr
5	mnth
6	holiday
7	weekday
8	workinday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered
16	cnt

CHAPTER 2

Methodology

2.1 DATA PREPROCESSING

The first step of the data analysis is the data pre processing. It involves finding out whether there are any missing values present in the given dataset. If the missing data in the dataset is more than 30% then ignore the predictor which has those missing values. On the other hand if the given data has les than 30% of missing values then impute the missing values by mean, median or any other method.

With reference to the given dataset there are no missing values in the data as shown in fig 2.1

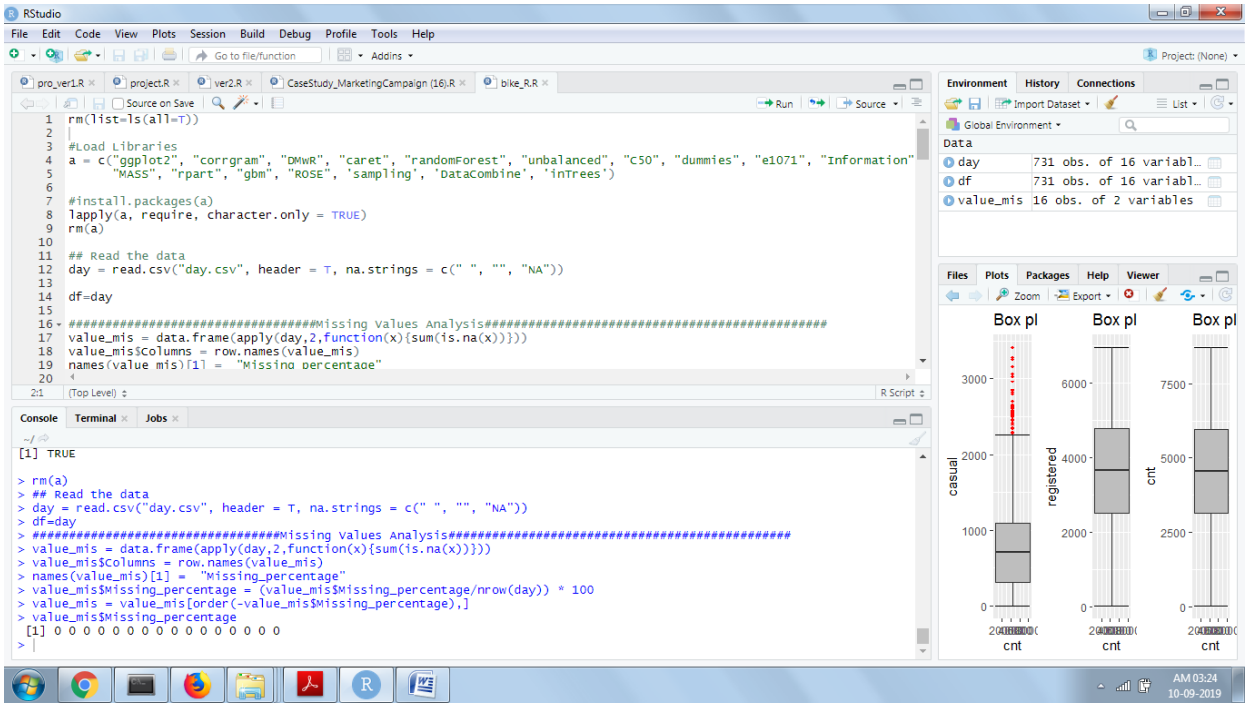


fig 2.1: Missing value analysis

2.2 OUTLIER ANALYSIS

In data analysis there is possibility that there are outliers present in the given dataset. An **outlier** is an observation that lies an abnormal distance from other values in a random sample from a population. Examination of the **data** for unusual observations that are far removed from the mass of **data**. These points are often referred to as **outliers**.

Outliers affect the mean value of the **data** but have little **effect** on the median or mode of a given set of **data**.

There are different ways to recognize the presence of outliers. One of the method used is boxplot as shown in fig 2.2. The red colour dots in the figure2.2 indicate the presence of outliers.

It is clear from the figure that predictor wind speed has more number of outliers whereas humidity has lesser number of outliers.

Outliers are imputed either by using mean , mode or median based on the requirement

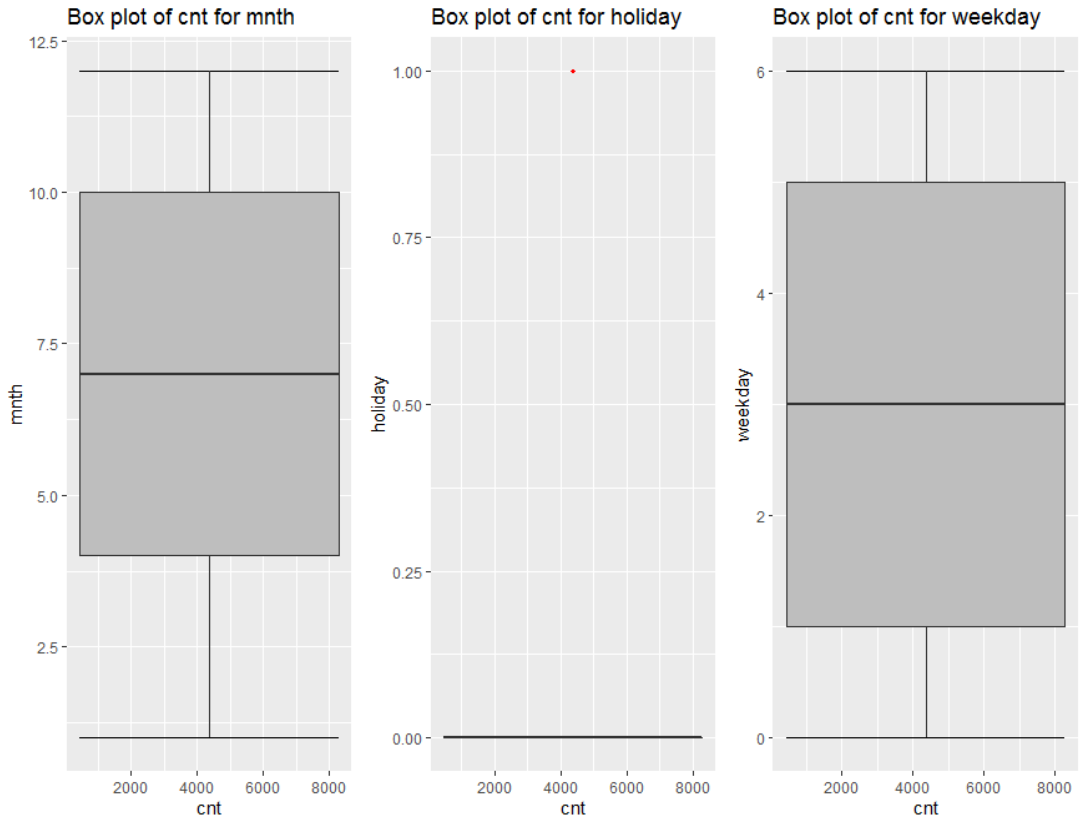


fig 2.2: Outlier analysis

2.3 FEATURE SELECTION

In the given data it is very important to select the desired predictors. One of the method used is correlation analysis. If the predictors are correlated then such predictors should be ignored. A sample correlation plot is shown in figure 2.3.

In the figure blue colour indicates that the predictors are positively correlated. The red colour indicates that the predictors are negatively correlated. Hence those predictors could be ignored while building the model.

In the given dataset the variables temp and atemp are positively correlated. Hence any one of them can be ignored.

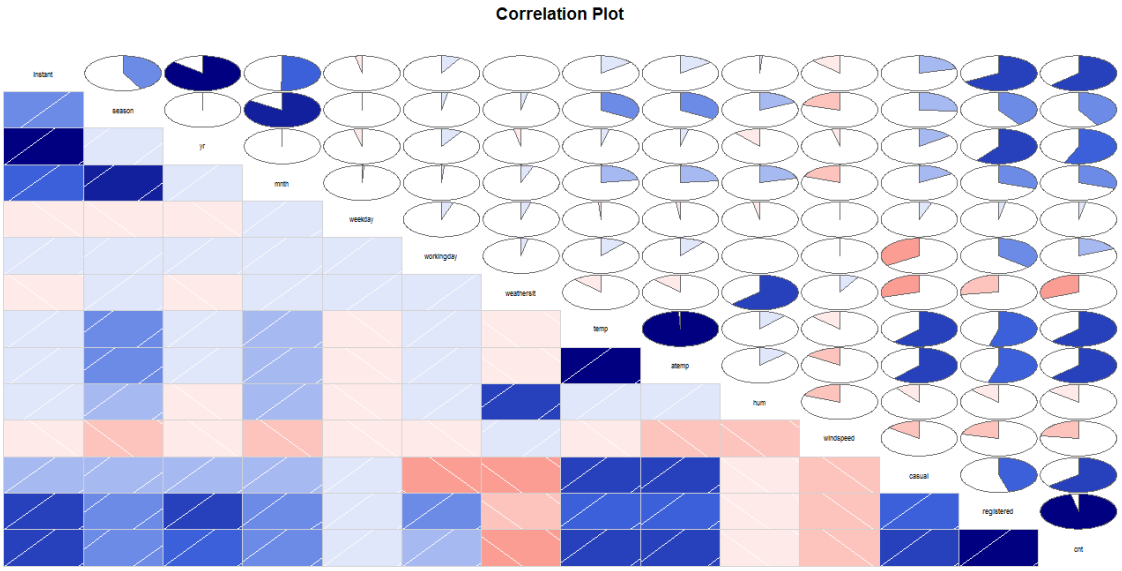


fig2.3: Correlation plot

After correlation plot is obtained it is a good practice to eliminate the predictors which are highly correlated. This step is referred as feature selection.

2.4 SAMPLING (To be performed if dataset is too huge)

Out of the given data only some of the data is selected over which the model could be built. It is called sampling process. Ones the sampling is carried out the given dataset is divided into training and testing data. Usually 80% is training and 20% is testing data.

The model is applied with a training data and is trained for different values of dataset. Then testing data is applied to the model and its functionality is verified in terms of accuracy,error rate.

2.5 MODEL SELECTION

Model selection plays a crucial step in data analysis. With respect to bike renting problem target variable is continuous. Hence it is a prediction problem. Therefore we need to make use of modelling techniques like decision trees, linear regression. In this project decision tree and linear regression has been used to model the system.

The accuracy is calculated in both the cases and compared the results.

The mean absolute percentage error (**MAPE**) is a statistical **measure** of how accurate a forecast system is. It measures this accuracy as a percentage, and can be **calculated** as the average absolute percent error for each time period minus actual values divided by actual values.

CHAPTER 3

Conclusion

3.1 MODEL EVALUATION

Since this is a prediction analysis problem wherein target variable is numerical or continuous either decision tree or regression could be employed. In the proposed problem decision tree is used and MAPE is calculated.

As discussed in 2.3 one of the variable i.e. temp is discarded because while implementing the model temp has resulted in higher value of MAPE than atemp.

Also by imputing the outliers with median, has resulted in higher value of MAPE. Hence mean is used.

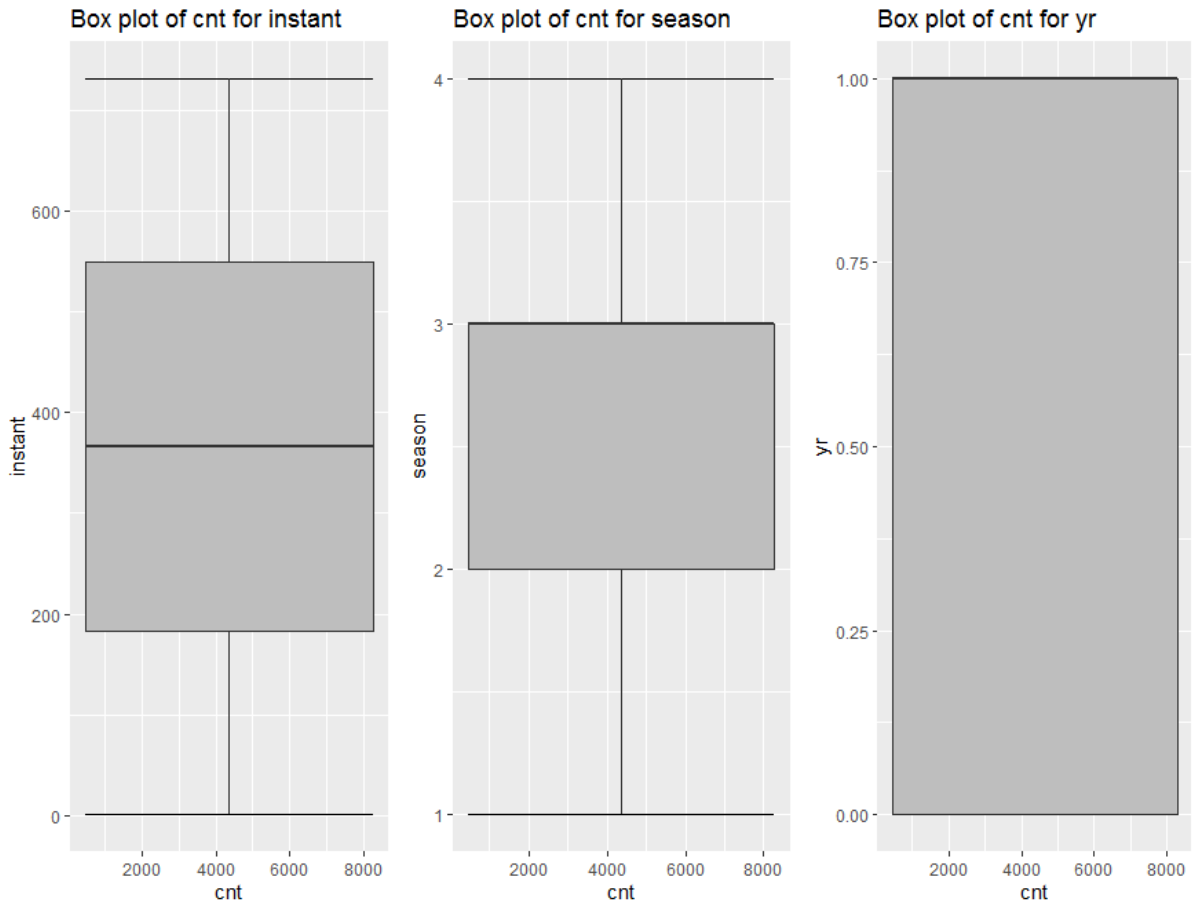
The following observations are made with python code

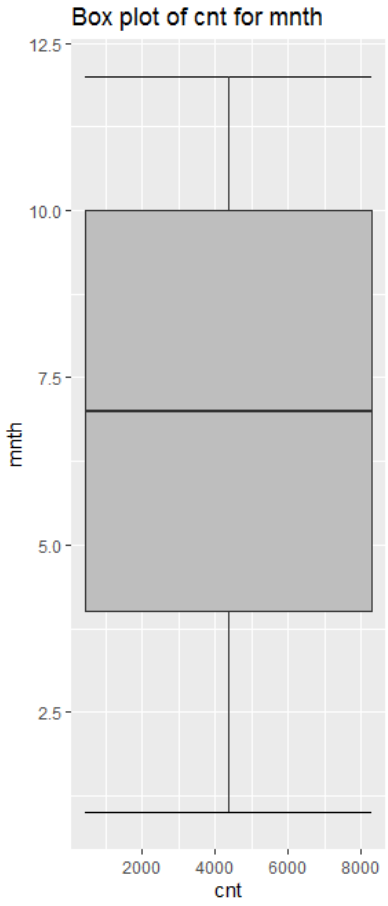
METHOD USED	MAPE	ACCCURACY
Decision Tree	20.75%	79.25%
Linear Regression	14.31%	85.69%

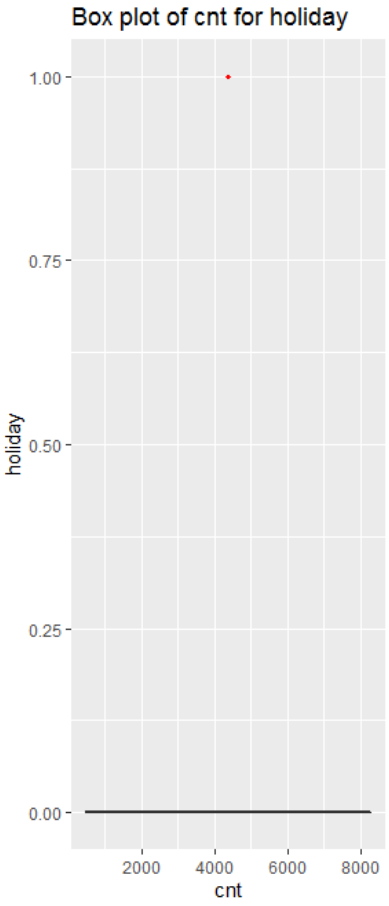
The following observations are made with R code

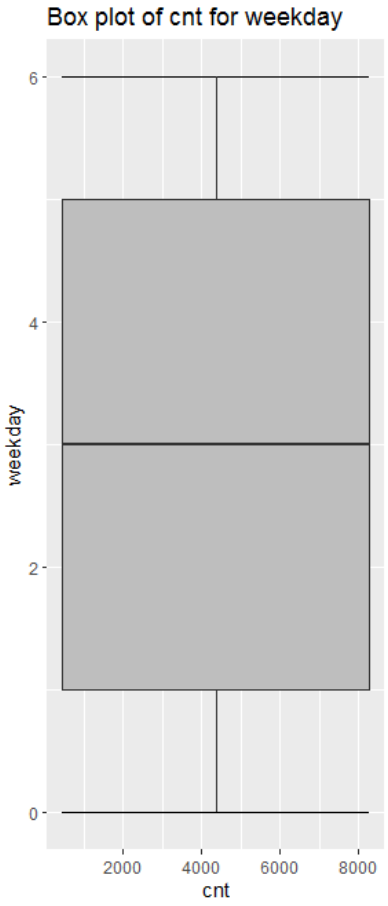
METHOD USED	MAPE	ACCCURACY
Decision Tree	19.95%	80.05%
Linear Regression	14.29%	85.71%

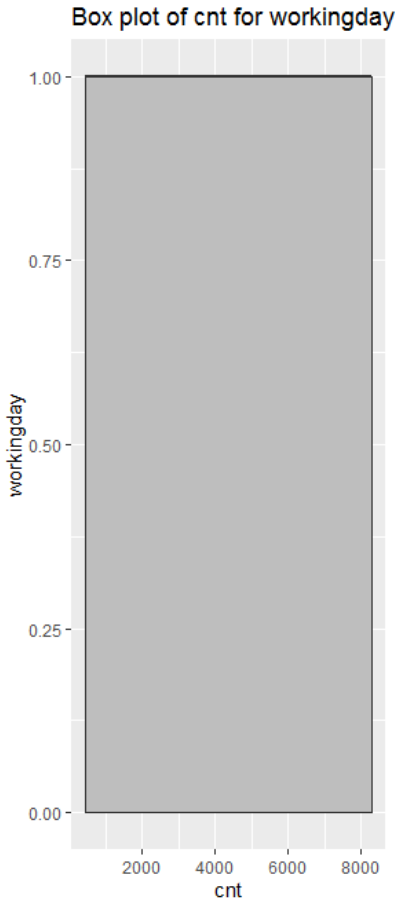
APPENDIX A

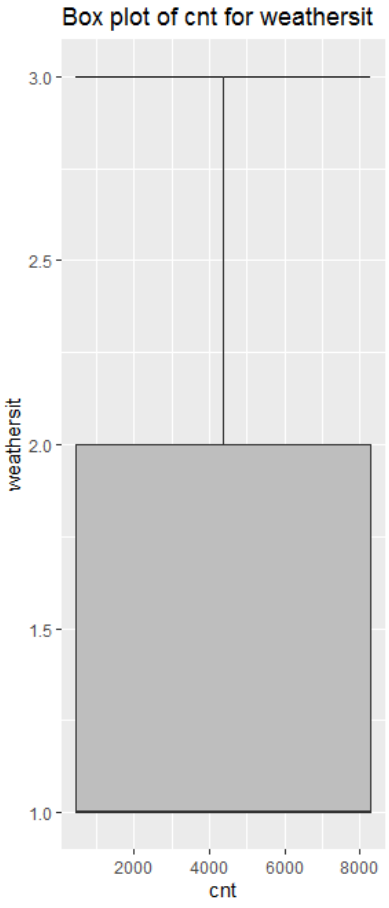

Box plot of cnt for instant
instant
0
200
400
600
2000
4000
6000
8000
cnt
Box plot of cnt for season
season
1
2
3
4
2000
4000
6000
8000
cnt
Box plot of cnt for yr
yr
0.00
0.25
0.50
0.75
1.00
2000
4000
6000
8000
cnt

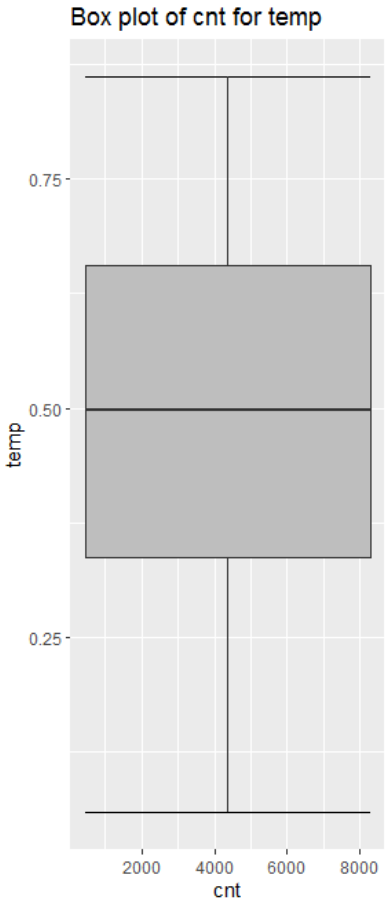

Box plot of cnt for mnth
mnth
12.5
10.0
7.5
5.0
2.5
2000
4000
6000
8000
cnt


Box plot of cnt for holiday
holiday
1.00
0.75
0.50
0.25
0.00
2000
4000
6000
8000
cnt


Box plot of cnt for weekday
weekday
6
4
2
0
2000
4000
6000
8000
cnt


Box plot of cnt for workingday
workingday
1.00
0.75
0.50
0.25
0.00
2000
4000
6000
8000
cnt


Box plot of cnt for weathersit
weathersit
3.0
2.5
2.0
1.5
1.0
2000
4000
6000
8000
cnt


Box plot of cnt for temp
temp
0.75
0.50
0.25
2000
4000
6000
8000
cnt

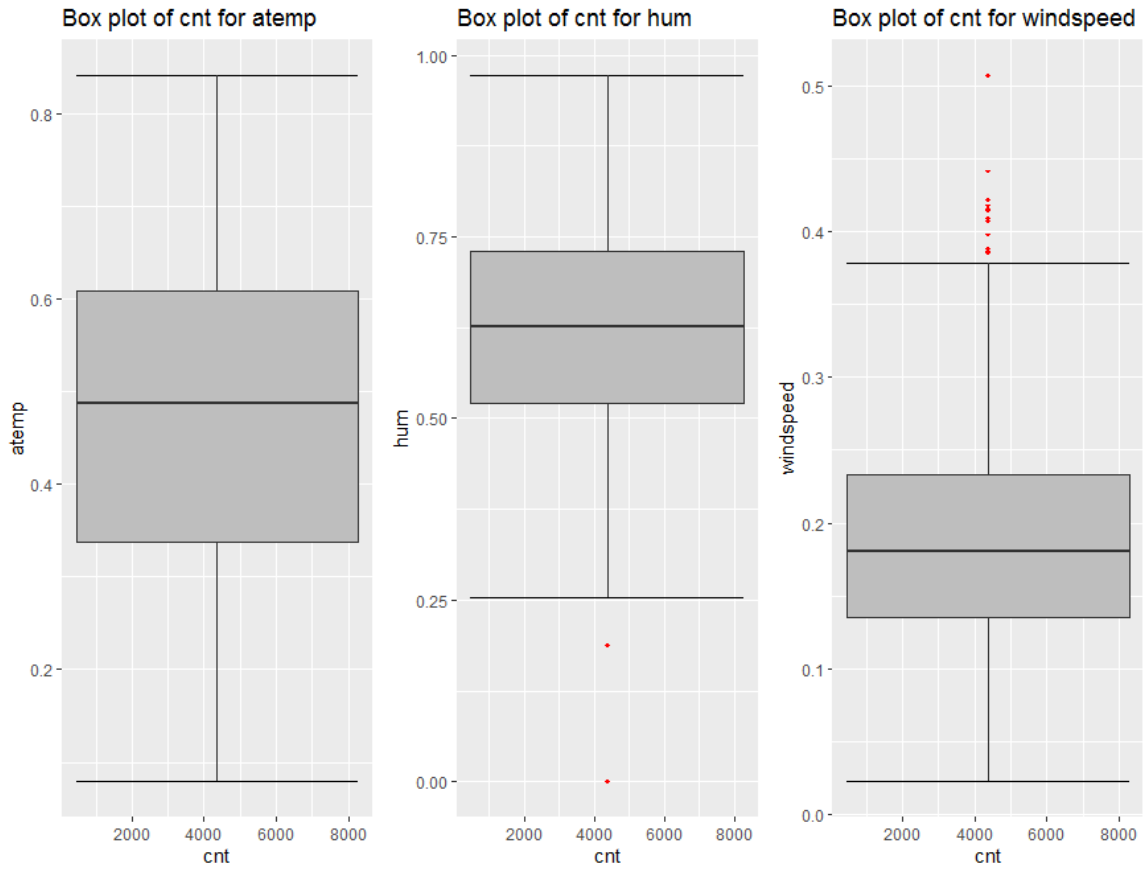


Fig A.1: Outlier Alalysis

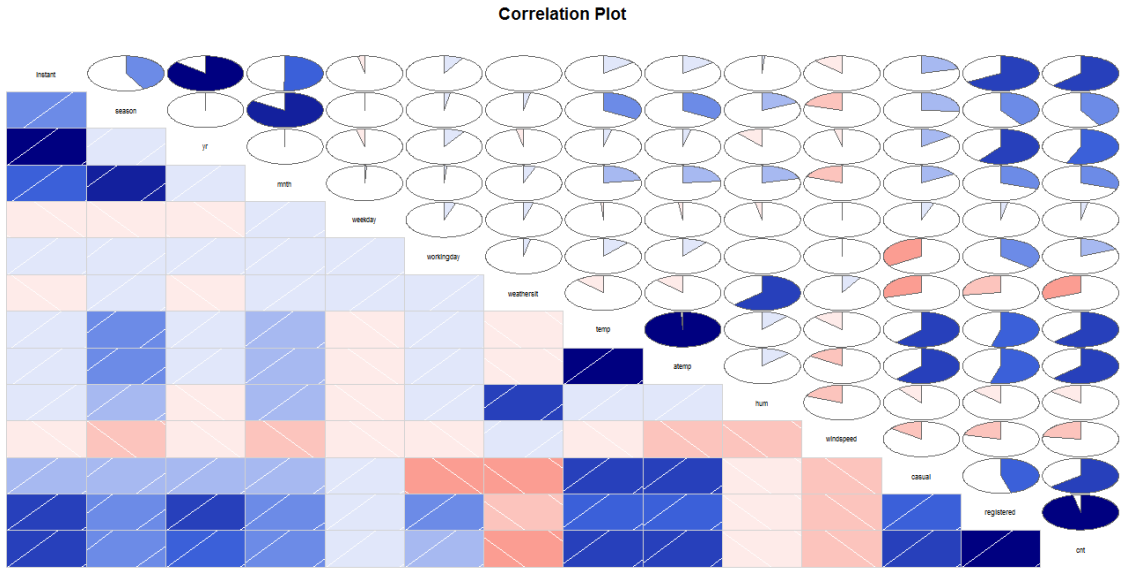


Fig A.2: Correlation Analysis

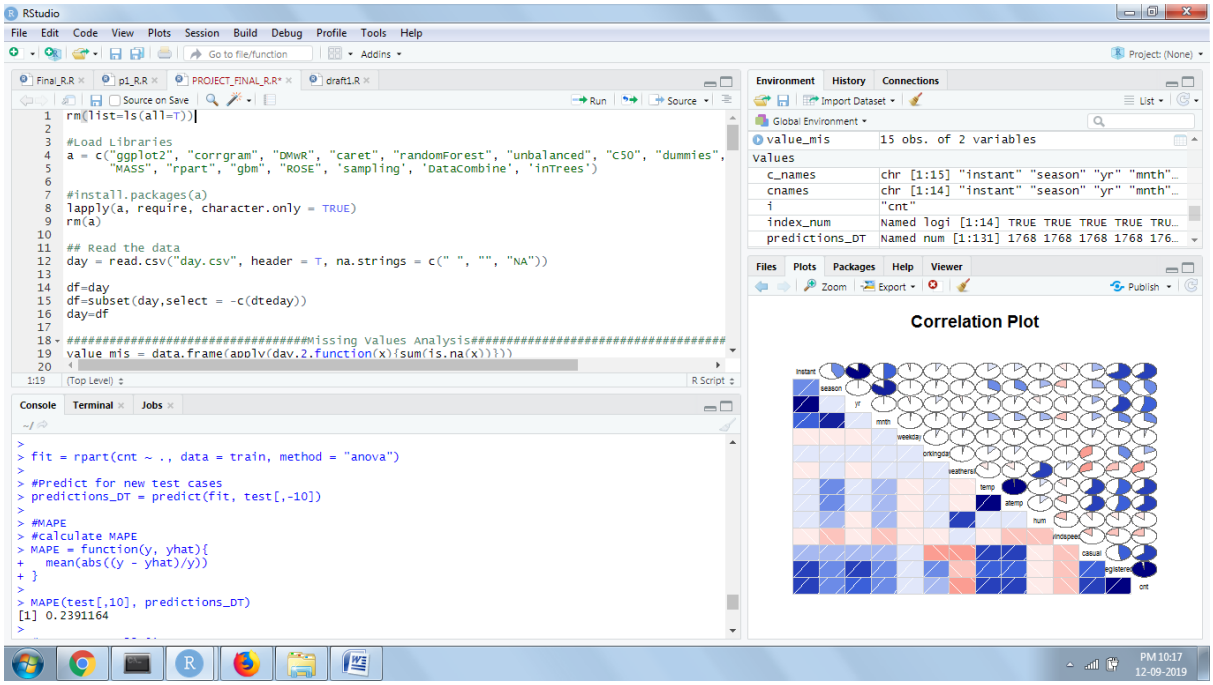


Fig A.3:MAPE is calculated as 0.2391 using decision tree

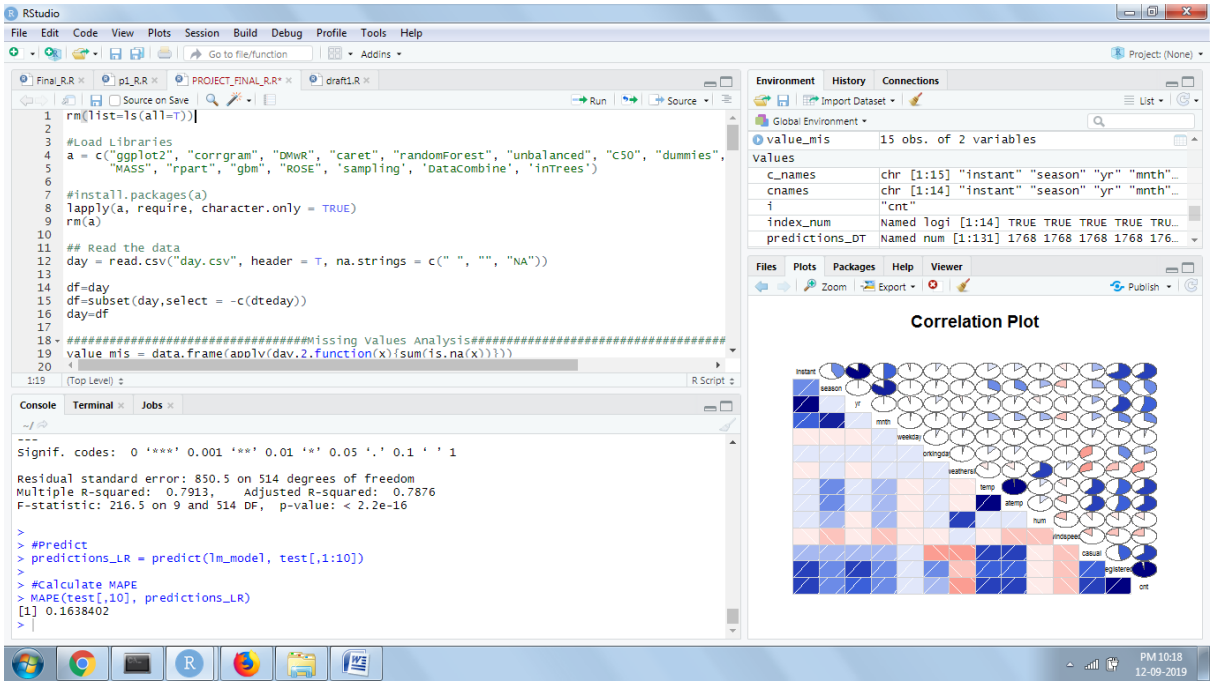


Fig A.4:MAPE is calculated as 0.1638 using linear regression

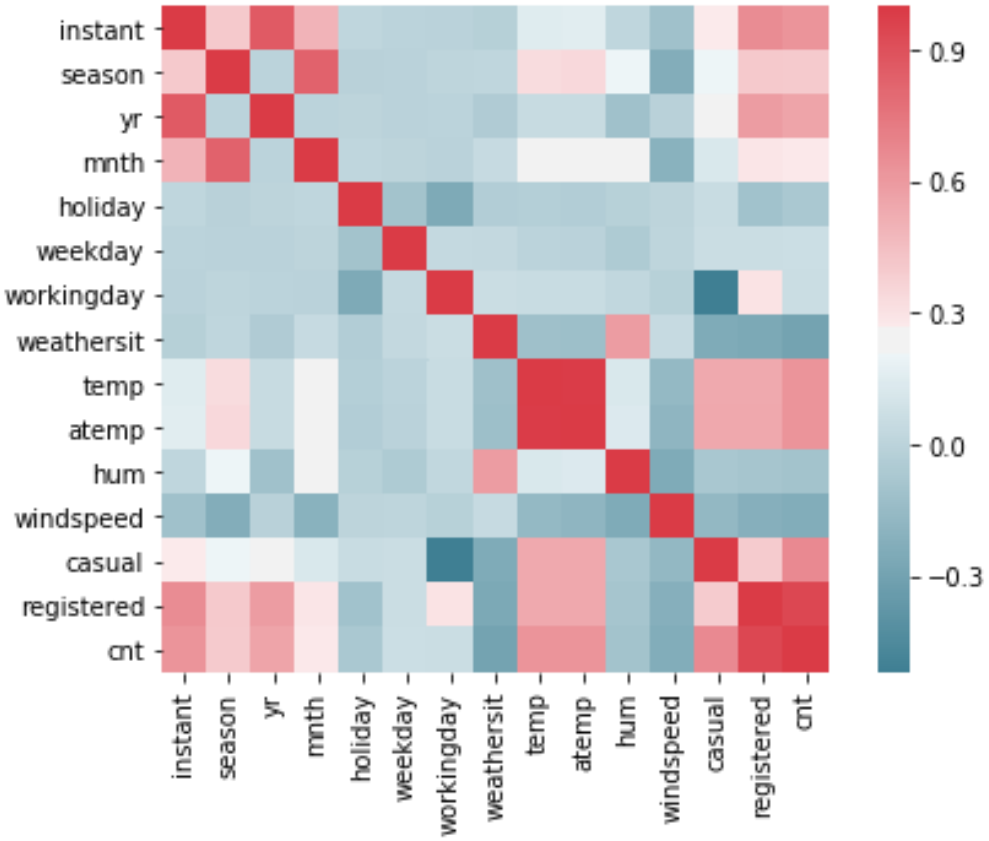


Fig A.5: Heat map of correlation

APPENDIX B

R CODE

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

#Load Libraries
a = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies",
"e1071", "Information",
     "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees')

#install.packages(a)
lapply(a, require, character.only = TRUE)
rm(a)

## Read the data
day = read.csv("day.csv", header = T, na.strings = c(" ", "", "NA"))
str(day)
df=day
df=subset(day,select = -c(dteday))
day=df

###################################Missing Values
Analysis#################################################
value_mis = data.frame(apply(day,2,function(x){sum(is.na(x))}))
value_mis$Columns = row.names(value_mis)
names(value_mis)[1] =  "Missing_percentage"
value_mis$Missing_percentage = (value_mis$Missing_percentage/nrow(day)) * 100
value_mis = value_mis[order(-value_mis$Missing_percentage),]
row.names(value_mis) = NULL

############################################Outlier
Analysis#############################################
# ## BoxPlots - Distribution and Outlier Check
index_num = sapply(day,is.numeric) #selecting only numeric

num_data = day[,index_num]
```

```
c_names = colnames(num_data)

for (i in 1:length(c_names))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (c_names[i]), x = "cnt"), data = subset(day))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=c_names[i],x="cnt")+
           ggtitle(paste("Box plot of cnt for",c_names[i])))
}
#
# ## Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
gridExtra::grid.arrange(gn10,gn11,gn12,ncol=3)
gridExtra::grid.arrange(gn13,gn14,gn15,ncol=3)

# # #loop to remove from all variables
for(i in c_names){
  print(i)
  val = day[,i][day[,i] %in% boxplot.stats(day[,i])$out]
  #print(length(val))
  day = day[which(!day[,i] %in% val),]
}

# #Replace all outliers with NA and impute

for(i in c_names){
  value = day[,i][day[,i] %in% boxplot.stats(day[,i])$out]
  #   #print(length(val))
  day[,i][day[,i] %in% value] = NA
}
#
# IMpute the NA from predictors windspeed and casual
```

```
day$holiday[is.na(day$holiday)] = mean(day$holiday,na.rm = T)
day$hum[is.na(day$hum)] = mean(day$hum,na.rm = T)
day$windspeed[is.na(day$windspeed)] = mean(day$windspeed,na.rm = T)
day$casual[is.na(day$casual)] = mean(day$casual,na.rm = T)
day1=day
day1=subset(day,select = -c(holiday))

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

df1 =day1
day1=df1
index_num = sapply(day1,is.numeric) #selecting only numeric

numeric_data = day1[,index_num]

cnames = colnames(numeric_data) #column names of only numeric variables

#  get the correlation plot
corrgram(day1[,index_num], order = F ,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

# Dimension Reduction

day_sub = subset(day,select = -c(instant,atemp,holiday,registered,casual))
day1=day_sub

#Divide the data into train and test
#set.seed(123)
train_index = sample(1:nrow(day1), 0.8 * nrow(day1))
train_data = day1[train_index,]
test_data = day1[-train_index,]

# Perform analysis of variance
fit = rpart(cnt ~ ., data = train_data, method = "anova")

#Predict for new test cases
```

```
predictions_DT = predict(fit, test_data[,-10])

#MAPE
#calculate MAPE

MAPE = function(y, y_pred){
  mean(abs((y - y_pred)/y))
}
MAPE(test_data[,10], predictions_DT)
# MAPE = function(y, yhat){
#   mean(abs((y - yhat)/y))
# }
# MAPE(test_data[,10], predictions_DT)

#ERROR RATE: 23.91
#ACCURACY:76.09

#Linear Regression
#check  for multicollearity of independent variables
library(usdm)
vif(df1[,-10])

vifcor(df1[,-10], th = 0.9)

#run regression model
reg_model = lm(cnt ~., data = train_data)

#Summary of the model
summary(reg_model)

#Predict
predictions_LR = predict(reg_model, test_data[,1:10])

#Calculate MAPE
MAPE(test_data[,10], predictions_LR)

# #Error Rate: 16.38
```

#acuracy: 83.62

PYHTON CODE

```
#Load all the required libraries
import os
import pandas as pd
import numpy as np
import sklearn
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
from sklearn.model_selection import train_test_split
import sklearn
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
#Load the  data into python environment
day = pd.read_csv("day.csv",encoding="latin1")
mising_value = pd.DataFrame(day.isnull().sum())
mising_value = mising_value.reset_index()
#Rename variable
mising_value = mising_value.rename(columns = {'index': 'Variable', 0: 'Missing_percentage'})
#Calculate percentage of missing value : NO MISSING VALUES
mising_value['Missing_percentage'] = (mising_value['Missing_percentage']/len(day))*100
# create a copy of the variable
df = day.copy()
%matplotlib inline
# Use poxplot to find the outliers
boxplot = df.boxplot(grid=True, rot=90, fontsize=10)
plt.boxplot(day['windspeed'])
#save only numeric values from the dataset
col_names=["instant","season","yr","mnth","holiday","weekday","workingday","weathersit
","temp","atemp","hum","windspeed","casual","registered","cnt"]
```

```
# Calculate interquartile range to determine outliers
for i in col_names:
    #print(i)
   q75, q25 = np.percentile(day.loc[:,i], [75 ,25])
   iqr = q75 - q25
   minimum = q25 - (iqr*1.5)# calculate lower fence
   maximum = q75 + (iqr*1.5)# calculate upper fence
   print(minimum)
   print(maximum)

   day = day.drop(day[day.loc[:,i] < minimum].index)
   day = day.drop(day[day.loc[:,i] > maximum].index)
#Detect and replace with NA
# #Extract quartiles
q75_1, q25_1 = np.percentile(day['hum'], [75 ,25])
q75_2, q25_2 = np.percentile(day['windspeed'], [75 ,25])
q75_3, q25_3 = np.percentile(day['casual'], [75 ,25])
# #Calculate IQR
iqr_1 = q75_1 - q25_1
iqr_2 = q75_2 - q25_2
iqr_3 = q75_3 - q25_3

# #Calculate inner and outer fence
minimum_1 = q25_1 - (iqr_1*1.5)
maximum_1 = q75_1 + (iqr_1*1.5)

minimum_2 = q25_2 - (iqr_2*1.5)
maximum_2 = q75_2 + (iqr_2*1.5)

minimum_3 = q25_3 - (iqr_3*1.5)
maximum_3 = q75_3 + (iqr_3*1.5)

# #Replace with NA
day.loc[day['hum'] < minimum,:'hum'] = np.nan
day.loc[day['windspeed'] < minimum,:'windspeed'] = np.nan
day.loc[day['casual'] > maximum,:'casual'] = np.nan
```

```
# #Calculate missing value
missing_val = pd.DataFrame(day.isnull().sum())

#Impute the missing valuse by mean
day['hum'] = day['hum'].fillna(day['hum'].mean())
day['windspeed'] = day['windspeed'].fillna(day['windspeed'].mean())
day['casual'] = day['casual'].fillna(day['casual'].mean())
##Correlation analysis
#Correlation plot
df_corr = day.loc[:,col_names]
#Set the width and hieght of the plot
fun, ax = plt.subplots(figsize=(8, 6))

#Generate correlation matrix
cor_mat = df_corr.corr()

#Plot using seaborn library
sns.heatmap(cor_mat, mask=np.zeros_like(cor_mat, dtype=np.bool),
cmap=sns.diverging_palette(200, 20, as_cmap=True),
        square=True, ax=ax)
#Feature Scaling

df = day.copy()
day = df.copy()
df = day.drop(['instant','casual','registered','dteday','holiday','temp'], axis=1)
col_names=["season","yr","mnth","holiday","weekday","workingday","weathersit","atemp
","cnt"]
Import Libraries for decision tree
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
#train, test = train_test_split(df, test_size=0.2)
training,testing=train_test_split(df, test_size=0.2)

#Decision tree for regression
DTree = DecisionTreeRegressor(max_depth=2).fit(training.iloc[:,0:9], training.iloc[:,9])
```

```
#Apply model on test data
p_DTree = DTree.predict(testing.iloc[:,0:9])

#Calculate MAPE
def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
   return mape
MAPE(testing.iloc[:,9], p_DTree)
import statsmodels.api as sm
#Train the model using the training sets
model = sm.OLS(training.iloc[:,9], training.iloc[:,0:9]).fit()
#Print out the statistics
model.summary()
final = sm.OLS(training.iloc[:,9], training.iloc[:,0:9]).fit()

# Print out the statistics
final.summary()
# make the predictions by the model
#predictions_LR = final.predict(testing.iloc[:,0:9])
linear_reg = final.predict(testing.iloc[:,0:9])

#Calculate MAPE
def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
   return mape
#Calculate MAPE
MAPE(testing.iloc[:,9], linear_reg)
```