**Prediction of Bike Rental Count**

*Sahayaraj Daniel*

Apr 24 2019

**Contents**

[**Chapter 1** 3](#_Toc6957611)

[**Introduction** 3](#_Toc6957612)

[**1.1** **Problem Statement** 3](#_Toc6957613)

[**1.2** **Data** 3](#_Toc6957614)

[**Chapter 2** 6](#_Toc6957615)

[**Methodology** 6](#_Toc6957616)

[**2.1 Pre-Processing** 6](#_Toc6957617)

[**2.1.1 Missing Value Analysis** 8](#_Toc6957618)

[**2.1.2 Outlier Analysis** 9](#_Toc6957619)

[**2.1.3 Feature Selection** 10](#_Toc6957620)

[**2.2 Modelling** 11](#_Toc6957621)

[**2.2.1 Model Selection** 11](#_Toc6957622)

[**2.2.2 Decision Tree** 11](#_Toc6957623)

[**2.2.3 Random Forest** 12](#_Toc6957624)

[**2.2.4 Linear Regression** 12](#_Toc6957625)

[**Chapter 3** 14](#_Toc6957626)

[**Conclusion** 14](#_Toc6957627)

[**3.1 Model Evaluation** 14](#_Toc6957628)

[**3.2 Mean Absolute Percentage Error:** 14](#_Toc6957629)

[**3.3** **Model Selection** 15](#_Toc6957630)

[**Appendix A – R Code** 16](#_Toc6957631)

[**Appendix B – Python Code** 21](#_Toc6957632)

# **Chapter 1**

# **Introduction**

## **Problem Statement**

The objective of this project is to predict the count of rented bikes based on environmental and seasonal settings. This prediction becomes crucial because knowing the number of bikes needed for a day prior can help the business arrange the number of free bikes as per the requirement. Seasonal and Environmental data can be widely taken from a variety of available sources for forecast sites. With that details, we are going to predict how they are affecting the counts of bike rented.

## **Data**

Our task is to build regression models which will predict the number of bikes that can possibly be rented depending on multiple environmental factors like temperature, humidity, windspeed, etc. Given below is a sample of the data set that we are using to predict the rented bikes count:

The details of data attributes in the dataset are as follows -

**instant**: Record index

**dteday**: Date

**season**: Season (1: spring, 2: summer, 3: fall, 4: winter)

**yr**: Year (0: 2011, 1:2012)

**mnth**: Month (1 to 12)

**holiday**: weather day is holiday or not (extracted from Holiday Schedule)

**weekday**: Day of the week

**workingday**: If day is neither weekend nor holiday is 1, otherwise is 0.

**weathersit**: (extracted 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

Here the dependant variables are the casual, registered and cnt, which are the bike counts on that day, whereas season, holiday, weathersit, temp, atemp, hum are the independent variables that will help us to predict the count of bikes rented.

**Some sample data:**

**Table1.1 Sample data (Column 1 to 8)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| instant | dteday | season | yr | mnth | holiday | weekday | workingday |
| 1 | 01-01-2011 | 1 | 0 | 1 | 0 | 6 | 0 |
| 2 | 02-01-2011 | 1 | 0 | 1 | 0 | 0 | 0 |
| 3 | 03-01-2011 | 1 | 0 | 1 | 0 | 1 | 1 |
| 4 | 04-01-2011 | 1 | 0 | 1 | 0 | 2 | 1 |
| 5 | 05-01-2011 | 1 | 0 | 1 | 0 | 3 | 1 |
| 6 | 06-01-2011 | 1 | 0 | 1 | 0 | 4 | 1 |
| 7 | 07-01-2011 | 1 | 0 | 1 | 0 | 5 | 1 |
| 8 | 08-01-2011 | 1 | 0 | 1 | 0 | 6 | 0 |
| 9 | 09-01-2011 | 1 | 0 | 1 | 0 | 0 | 0 |

**Table1.2 Sample data (Column 9 to 16)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 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 | 0.089565 | 88 | 1518 | ***1606*** |
| 2 | 0.196522 | 0.208839 | 0.498696 | 0.168726 | 148 | 1362 | ***1510*** |
| 2 | 0.165 | 0.162254 | 0.535833 | 0.266804 | 68 | 891 | ***959*** |
| 1 | 0.138333 | 0.116175 | 0.434167 | 0.36195 | 54 | 768 | ***822*** |

Our aim to predict the count of bike based on environment conditions. Therefore, the predictor variables we will be using for our project are:

**Table 1.3 Predictor variable list**

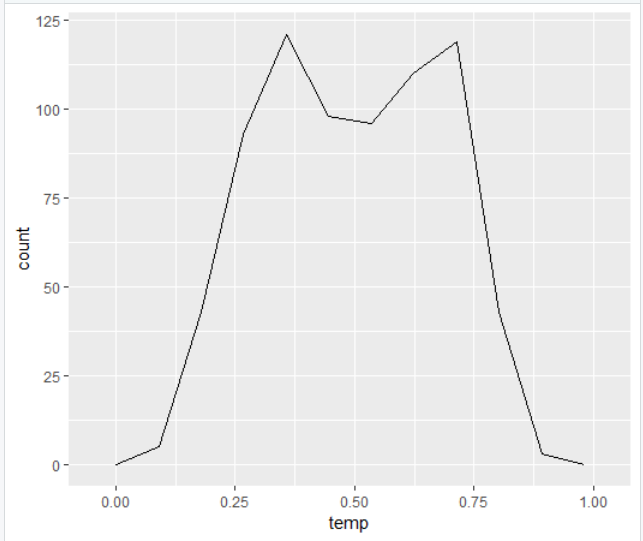
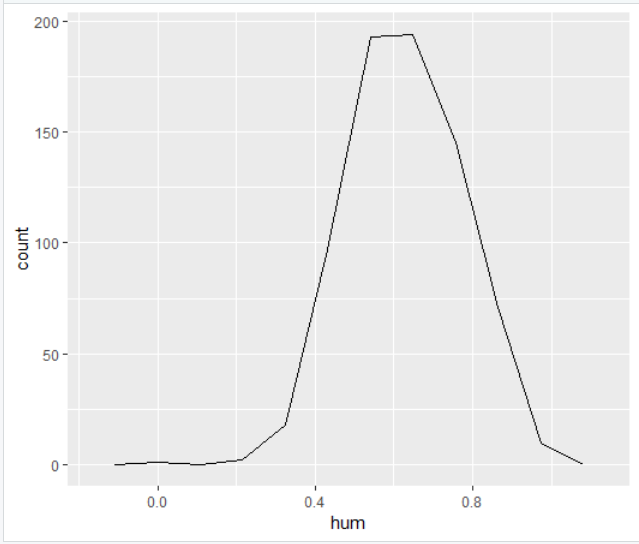
|  |  |
| --- | --- |
| S.No | Predictors |
| 1 | season |
| 2 | weathersit |
| 3 | temp |
| 4 | atemp |
| 5 | hum |
| 6 | windspeed |

# **Chapter 2**

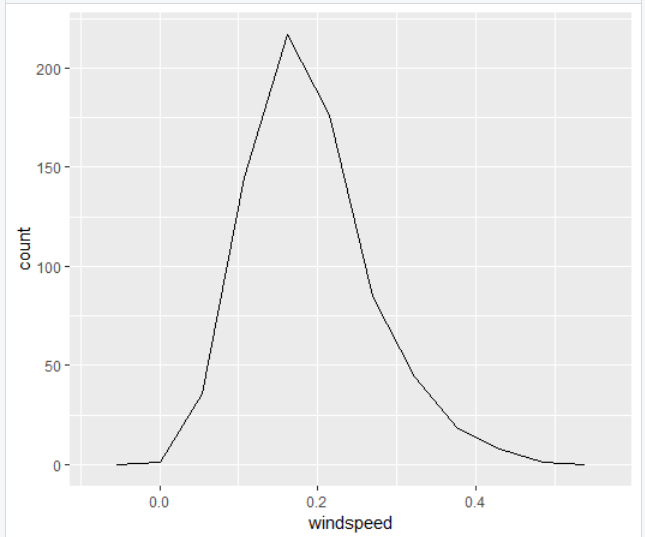
# **Methodology**

## **2.1 Pre-Processing**

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called as Exploratory Data Analysis. To start this process we will ﬁrst try and look at all the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions of the predictor variables. Also, the distribution of factor variables are shown in box plot.

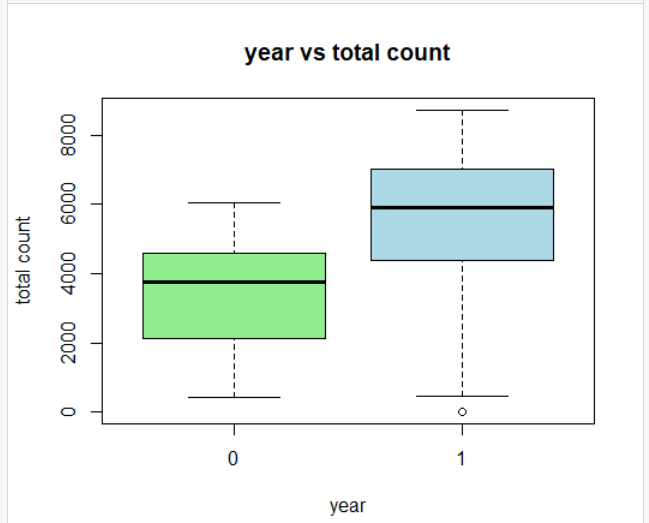
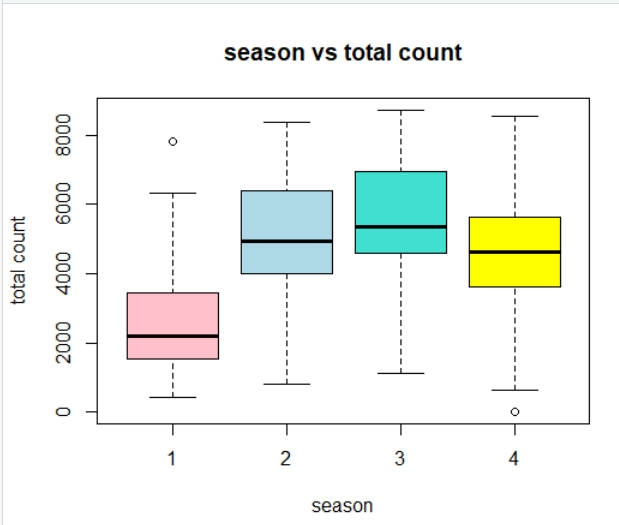
 

(a) (b)

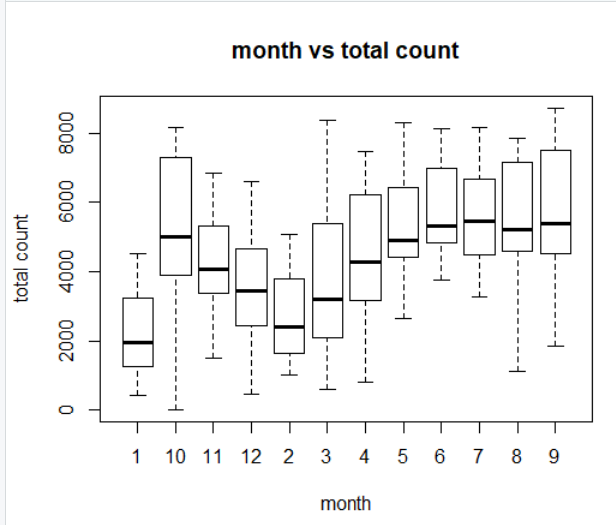


(c)

Fig 2.1 Density Function of continuous predictor variables

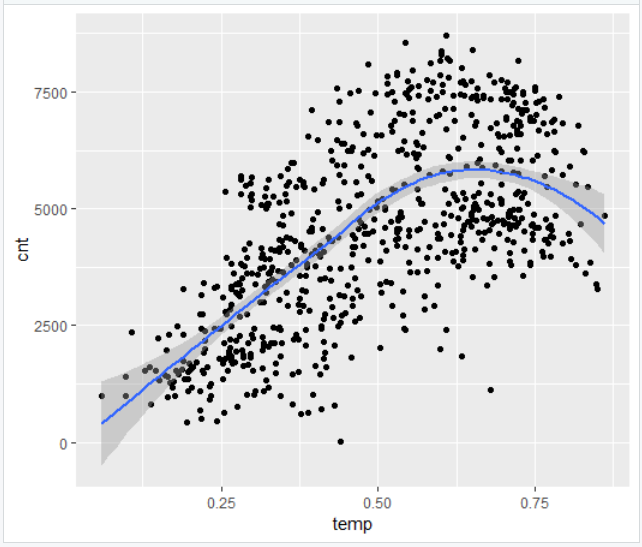
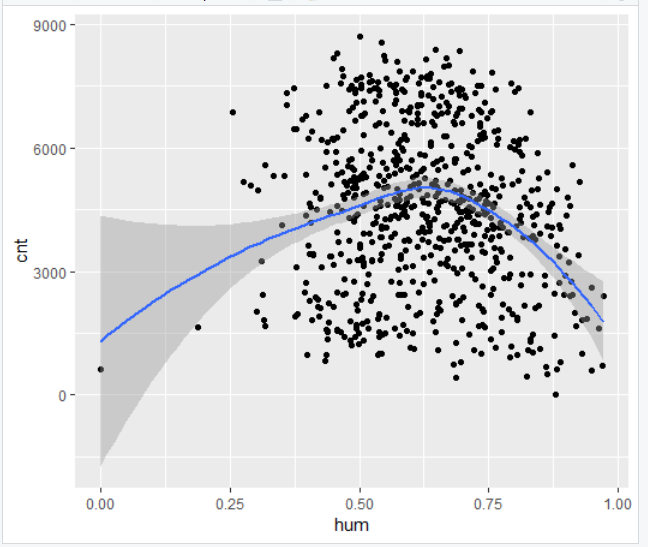
 

(a) (b)

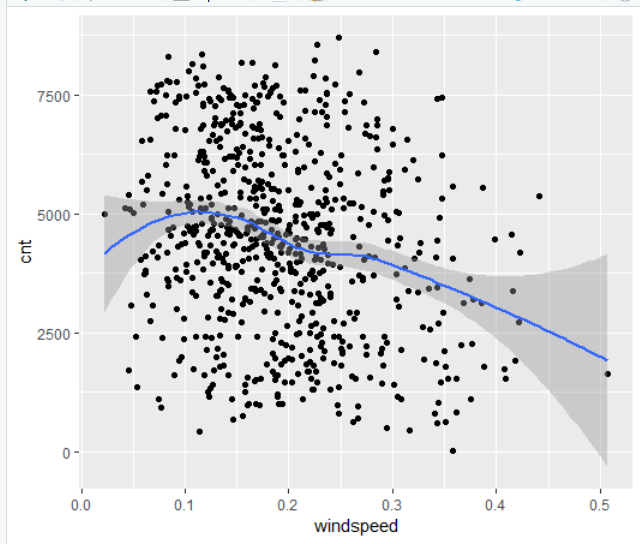


(c)

Fig 2.2 Box Plot of Count vs Year, Season and Month variables

(a) (b)



(c)

Fig 2.3 Variation of cnt variable for different predictor variable (Distribution)

### **2.1.1 Missing Value Analysis**

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned. We checked for missing values in all the variables of the data set and there is no missing value found in this data set. Hence there is no need to compute the missing values and all the data set can be taken for analysis.

> missing\_value = data.frame(apply(day\_data, 2, function(x){sum(is.na(x))}))

> missing\_value

apply.day\_data..2..function.x...

season 0

mnth 0

holiday 0

weekday 0

workingday 0

weathersit 0

temp 0

hum 0

windspeed 0

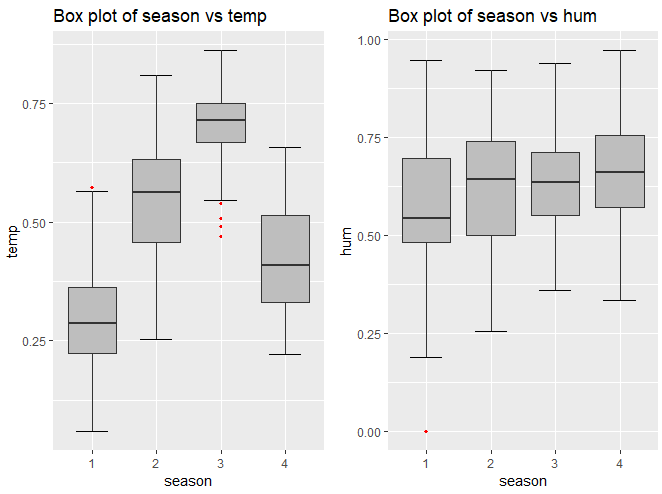
cnt 0

### **2.1.2 Outlier Analysis**

An outlier is an element of a data set that distinctly stands out from the rest of the data. In other words, outliers are those data points that lie outside the overall pattern of distribution as shown in figure below. The easiest way to detect outliers is to create a graph. Plots such as Box plots, Scatterplots and Histograms can help to detect outliers. In our case, we have used Box plots for various predictor variables. Since we have environmental parameters, we have plotted the box plot in such a way that how these parameters differ in given a season.

It can be seen that only for season 3, very low temperature values are recorded and windspeed has some outlier values. Since these are environmental factors are highly random, we are going to leave these values as it is and continue our analysis.

We can perform another analysis by removing the outlier values of windspeed factor and compare the results.



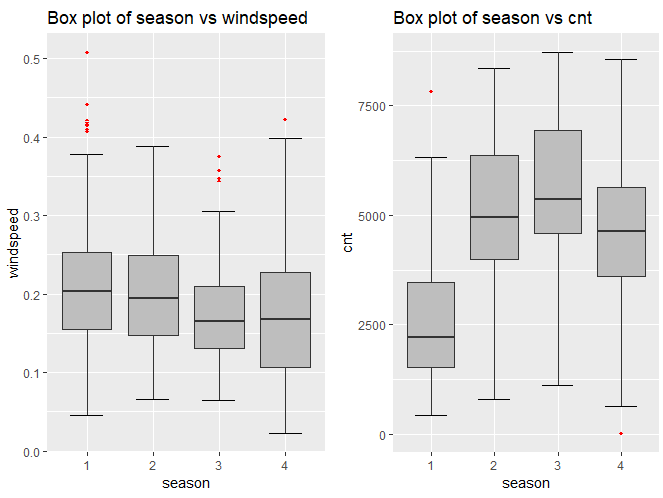


Fig 2.4 Outlier analysis – Box Plots showing outliers

### **2.1.3 Feature Selection**

Before performing any type of modelling we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of doing that and we have used a very simple method of finding the correlation between predictor variables through a correlation plot.

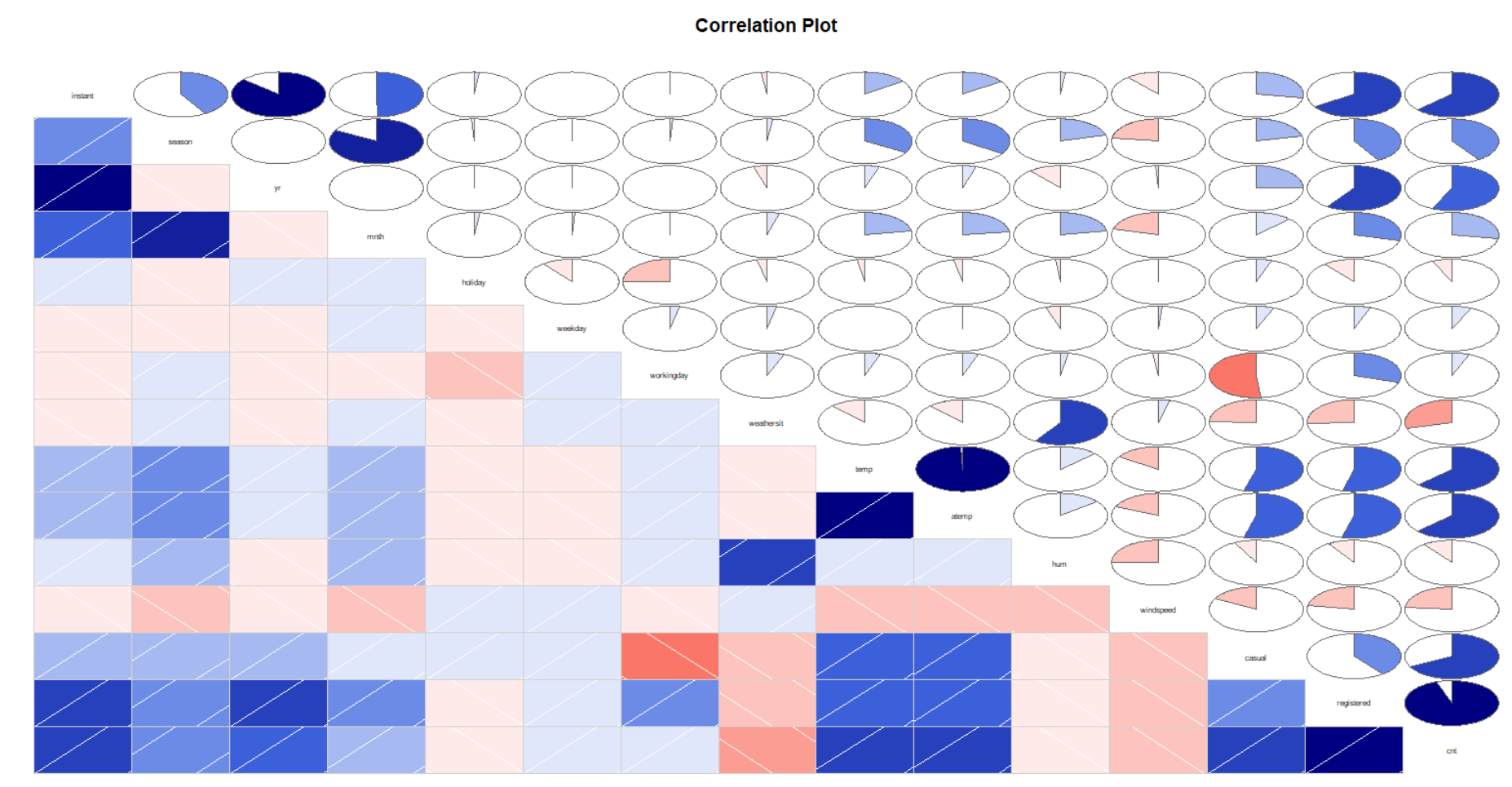


Fig 2.5 Correlation Plot of all the variables

In this plot, we can see that temp and atemp variable are highly correlated and atemp can be ignored in predicting the bike counts, as it will have the same effect as temp variable. Also between dependant variables casual, registered and cnt, casual and registered fields are highly correlated to cnt field and therefore we will be considering only cnt variable as dependant and other two fields are ignored.

## **2.2 Modelling**

### **2.2.1 Model Selection**

Selecting a model for data set purely depends on the nature of the dependant variable or the variable that must be predicted. The dependent variable can fall in either of the four categories:

1. Nominal
2. Ordinal
3. Interval
4. Ratio

If the dependent variable, in our case cnt (total count of bike rented), is Nominal the only predictive analysis that we can perform is Classiﬁcation, and if the dependent variable is Interval or Ratio the normal method is to do a Regression analysis, or classiﬁcation after binning; and if the dependent variable is Ordinal, then both classiﬁcation and regression can be done.

In our case, the variable to be predicted being continuous, we are going for Regression techniques. You always start your model building from the simplest to more complex. Here we are going to use three regression techniques, namely Decision tree Algorithm, Random Forest and Linear Regression techniques.

### **2.2.2 Decision Tree**

Decision tree is a type of supervised learning algorithm that can be used in both regression and classification problems. It works for both categorical and continuous input and output variables. Since our data is a continuous data, we have chosen this algorithm.

We are splitting the data into test and train data, where training data being 90% of the total data. We chose 90% training data as this percentage of data gave better results.

train\_index = sample(1:nrow(day\_data), 0.9 \* nrow(day\_data))

train = day\_data[train\_index,]

test = day\_data[-train\_index,]

##Regression algorithm - anova

regression\_result = rpart(cnt ~ ., data = train, method = "anova")

We are using anova method to predict the cnt variable and the decision tree used for regression is shown below.

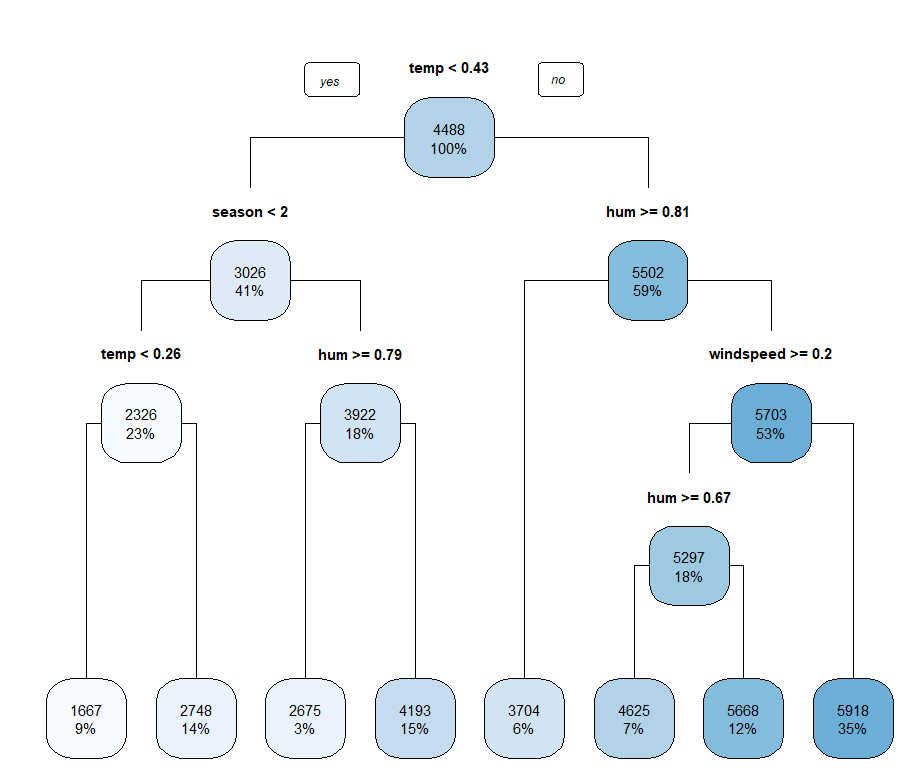


Fig 2.6 Decision tree built by Decision tree algorithm

### **2.2.3 Random Forest**

Random forest is a tree-based algorithm which involves building several trees (decision trees), then combining their output to improve generalization ability of the model. The method of combining trees is known as an ensemble method. For our modelling we have used 50 decision trees in our random forest algorithm

## Random Forest algorithm

RF\_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 50)

#Predict test data using random forest model

RF\_Predictions = predict(RF\_model, test[,-10])

### **2.2.4 Linear Regression**

Linear regression is one of the most commonly used predictive modelling techniques. The aim of linear regression is to find a mathematical equation for a continuous response variable Y as a function of one or more X variable(s). So that you can use this regression model to predict the Y when only the X is known.

#run regression model

lm\_model = lm(cnt ~., data = train)

#Summary of the model

summary(lm\_model)

Call:

lm(formula = cnt ~ ., data = train)

Residuals:

Min 1Q Median 3Q Max

-4245.8 -965.8 -165.9 1053.1 4031.3

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 3299.47 360.91 9.142 < 2e-16 \*\*\*

season 474.26 87.24 5.436 7.72e-08 \*\*\*

mnth -25.54 27.15 -0.941 0.347214

holiday -487.63 317.75 -1.535 0.125364

weekday 56.59 26.79 2.112 0.035044 \*

workingday 79.95 118.91 0.672 0.501595

weathersit -478.94 125.69 -3.811 0.000152 \*\*\*

temp 5603.88 310.74 18.034 < 2e-16 \*\*\*

hum -2331.34 500.05 -4.662 3.80e-06 \*\*\*

windspeed -3552.63 733.82 -4.841 1.62e-06 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1337 on 647 degrees of freedom

Multiple R-squared: 0.5329, Adjusted R-squared: 0.5264

F-statistic: 82.03 on 9 and 647 DF, p-value: < 2.2e-16

The summary statistics above tells us several things.

One of them is the model’s p-Value (in last line) and the p-Value of individual predictor variables (extreme right column under ‘Coefficients’).

The p-Values are very important.

Because, we can consider a linear model to be statistically significant only when both these p-Values are less than the pre-determined statistical significance level of 0.05.

This can visually interpreted by the significance stars at the end of the row against each X variable.

The more the stars beside the variable’s p-Value, the more significant the variable.

# **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 Bike rental 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.

There are so many error metrics available for regression analysis such as *Mean Absolute error (MAE), Mean Squared Error (MSE), mean absolute percentage error (MAPE) and Root Mean Squared Error (RMSE).* MSE and RMSE are widely used for time series analysis and therefore we are going to use MAPE to evaluate our model.

## **3.2 Mean Absolute Percentage Error:**

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. It usually expresses accuracy as a percentage, and is defined by the formula:

M =

where At is the actual value and Ft is the forecast value. The difference between At and Ft is divided by the actual value At again. The absolute value in this calculation is summed for every forecasted point in time and divided by the number of fitted points n. Multiplying by 100% makes it a percentage error.

#calculate MAPE

mape = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

That implying Accuracy is obtained by subtracting the MAPE from 100, through which our models are evaluated.

[1] "Accuracy of Decision tree with Outliers : "

[1] 67.39725

[1] "Accuracy of Decision tree without Outliers : "

[1] 62.7713

[1] "Accuracy of Linear Regression Model with Outliers : "

[1] 69.01409

[1] "Accuracy of Linear Regression Model without Outliers : "

[1] 62.80723

[1] "Accuracy of Random Forest with Outliers : "

[1] 65.75589

[1] "Accuracy of Random Forest without Outliers : "

[1] 64.6088

## **Model Selection**

It can be seen that there is no much difference between the accuracy of the three models evaluated and any one of the models can be used to predict the count of bikes rented. In case of python code, Random Forest provides more accuracy compared to other two models.

# **Appendix A – R Code**

#remove existing variables:

rm(list = ls())

library(ggplot2) #visualizations

library(caret)

library(DMwR)#knnImputation

library(corrgram)#for correlation calculaltion

library(rpart)#decision tree alg

library(rpart.plot)

library(randomForest)#for random forest algorithm

#set working directory

setwd("E:/EDW/Projects/Bike Rental")

## Read the data

day\_data = read.csv("day.csv", header = T)

#names(day\_data)

#[1] "instant" "dteday" "season" "yr" "mnth" "holiday" "weekday"

#[8] "workingday" "weathersit" "temp" "atemp" "hum" "windspeed" "casual"

#[15] "registered" "cnt"

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

#1. Missing Value Identification

missing\_value = data.frame(apply(day\_data, 2, function(x){sum(is.na(x))}))

#2. Outlier analysis

numeric\_index = sapply(day\_data,is.numeric) #selecting only numeric

numeric\_data = day\_data[,numeric\_index]

cnames = colnames(numeric\_data)

for (i in 1:length(cnames))

{

assign(paste0("boxplot",i), ggplot(aes\_string(y = (cnames[i]), x = "season"), data = subset(day\_data))+

stat\_boxplot(geom = "errorbar", width = 0.5) +

geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

outlier.size=1, notch=FALSE) +

theme(legend.position="bottom")+

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

ggtitle(paste("Box plot of season vs",cnames[i])))

}

# Plotting box plots together for dependant variables

#gridExtra::grid.arrange(boxplot2, boxplot4, ncol=2)

#gridExtra::grid.arrange(boxplot5, boxplot8,ncol=2)

df = day\_data

day\_data\_wo = day\_data

#day\_data = df

# #Replace all outliers(windspeed and temp) with NA and impute

val\_w = day\_data\_wo$windspeed[day\_data\_wo$windspeed %in% boxplot.stats(day\_data\_wo$windspeed)$out]

day\_data\_wo$windspeed[day\_data\_wo$windspeed %in% val\_w] = NA

#Imputation

day\_data\_wo = knnImputation(day\_data\_wo, k = 2)

##3. Feature Selection - Correlation Plot

corrgram(day\_data[,numeric\_index], order = F,

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

##atemp and temp are highly correlated and atemp can be removed

##casual and registered are correlated to cnt, hence cnt can be considered dependant variable

day\_data = subset(day\_data, select= -c(instant, dteday, yr, atemp, casual, registered))

##4.Feature Scaling is already done by normalising temp, hum, windspeed variables

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

#1. Decision tree algorith for regression

#Divide the data into train and test

train\_index = sample(1:nrow(day\_data), 0.9 \* nrow(day\_data))

train = day\_data[train\_index,]

test = day\_data[-train\_index,]

##Regression algorithm - anova

regression\_result = rpart(cnt ~ ., data = train, method = "anova")

#prediction of test values

predict\_DT = predict(regression\_result, test[,-10])

rpart.plot(regression\_result, type = 1)

#MAPE

#calculate MAPE

mape = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

accuracy\_DT = 100 - (mape(test[,10], predict\_DT))

#alternate method to find error metrics for regression.

regr.eval(test[,10], predict\_DT, stats = c('mae', 'rmse', 'mape', 'mse'))

##2. Random Forest algorithm

RF\_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 50)

#Predict test data using random forest model

RF\_Predictions = predict(RF\_model, test[,-10])

accuracy\_RF = 100 - (mape(test[,10], RF\_Predictions))

##3.Linear regression

#check multicollearity

library(usdm)

vif(day\_data[,-11])

vifcor(day\_data[,-11], th = 0.9)

#run regression model

lm\_model = lm(cnt ~., data = train)

#Summary of the model

summary(lm\_model)

#Predict

predictions\_LR = predict(lm\_model, test[,1:9])

accuracy\_LR = 100 - (mape(test[,10], predictions\_LR))

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

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

#1. Decision tree algorith for regression - removing outliers

#Divide the data into train and test

day\_data\_wo = subset(day\_data\_wo, select= -c(instant, dteday, yr, atemp, casual, registered))

train\_index\_wo = sample(1:nrow(day\_data\_wo), 0.9 \* nrow(day\_data\_wo))

train\_wo = day\_data\_wo[train\_index\_wo,]

test\_wo = day\_data\_wo[-train\_index\_wo,]

##Regression algorithm - anova

regression\_result\_wo = rpart(cnt ~ ., data = train\_wo, method = "anova")

#prediction of test values

predict\_DT\_wo = predict(regression\_result\_wo, test\_wo[,-10])

rpart.plot(regression\_result\_wo, type = 1)

#MAPE

#calculate MAPE

mape = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

accuracy\_DT\_wo = 100 - (mape(test\_wo[,10], predict\_DT\_wo))

#alternate method to find error metrics for regression.

regr.eval(test\_wo[,10], predict\_DT\_wo, stats = c('mae', 'rmse', 'mape', 'mse'))

##2. Random Forest algorithm

RF\_model\_wo = randomForest(cnt ~ ., train\_wo, importance = TRUE, ntree = 50)

#Predict test data using random forest model

RF\_Predictions\_wo = predict(RF\_model\_wo, test\_wo[,-10])

accuracy\_RF\_wo = 100 - (mape(test\_wo[,10], RF\_Predictions\_wo))

##3.Linear regression

#check multicollearity

library(usdm)

vif(day\_data\_wo[,-11])

vifcor(day\_data\_wo[,-11], th = 0.9)

#run regression model

lm\_model = lm(cnt ~., data = train\_wo)

#Summary of the model

summary(lm\_model)

#Predict

predictions\_LR\_wo = predict(lm\_model, test\_wo[,1:9])

accuracy\_LR\_wo = 100 - (mape(test\_wo[,10], predictions\_LR\_wo))

print("Accuracy of Decision tree with Outliers : ")

accuracy\_DT

print("Accuracy of Decision tree without Outliers : ")

accuracy\_DT\_wo

print("Accuracy of Linear Regression Model with Outliers : ")

accuracy\_LR

print("Accuracy of Linear Regression Model without Outliers : ")

accuracy\_LR\_wo

print("Accuracy of Random Forest with Outliers : ")

accuracy\_RF

print("Accuracy of Random Forest without Outliers : ")

accuracy\_RF\_wo

# **Appendix B – Python Code**

#Load libraries\n",

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

#Set working directory

os.chdir("E:\EDW\Projects\Bike Rental")

#os.getcwd()

#'E:\\EDW\\Projects\\Bike Rental'

#Load data

day\_data = pd.read\_csv("day.csv")

df=day\_data

day\_data\_wo = day\_data

#1. Missing Value Identification

missing\_val = pd.DataFrame(day\_data.isnull().sum())

#missing\_val

missing\_val = missing\_val.reset\_index() #Reset index

missing\_val = missing\_val.rename(columns = {'index': 'Variables', 0: 'Missing\_percentage'})#Rename variable

missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(day\_data))\*100 #Calculate Missing Percentage

#Plot boxplot to visualize Outliers\n",

%matplotlib inline

plt.boxplot(day\_data['temp'])

plt.boxplot(day\_data['atemp'])

plt.boxplot(day\_data['windspeed'])

plt.boxplot(day\_data['hum'])

plt.boxplot(day\_data['cnt'])

cnames = ["temp", "atemp", "hum", "windspeed", "casual", "registered", "cnt"]

#Detect and delete outliers from data

for i in cnames:

print(i)

q75, q25 = np.percentile(day\_data\_wo.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

day\_data\_wo = day\_data\_wo.drop(day\_data\_wo[day\_data\_wo.loc[:,i] < min].index)

day\_data\_wo = day\_data\_wo.drop(day\_data\_wo[day\_data\_wo.loc[:,i] > max].index)

##Correlation analysis

#Correlation plot

df\_corr = day\_data.loc[:,cnames]

#Set the width and hieght of the plot

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

#Generate correlation matrix

corr = df\_corr.corr()

#Plot using seaborn library\n",

sns.heatmap(corr, mask=np.zeros\_like(corr, dtype=np.bool), cmap=sns.diverging\_palette(220, 10, as\_cmap=True),square=True, ax=ax)

#Chisquare test of independence

#Save categorical variables

from scipy.stats import chi2\_contingency

cat\_names = ["season", "yr", "mnth", "holiday", "weekday", "workingday", "weathersit"]

#loop for chi square values

for i in cat\_names:

print(i)

chi2, p, dof, ex = chi2\_contingency(pd.crosstab(day\_data['cnt'], day\_data[i]))

print(p)

if p > 0.05:

print("Variable not useful in predicting")

day\_data = day\_data.drop(["instant", "dteday","workingday","weathersit", "atemp", "casual", "registered"], axis=1)

##Machine Learning Algorithms

import sklearn

from sklearn.model\_selection import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

#Divide data into train and test

train, test = train\_test\_split(day\_data, test\_size=0.1)

#Decision tree for regression

fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:8], train.iloc[:,8])

#Apply model on test data

predictions\_DT = fit\_DT.predict(test.iloc[:,0:8])

#Calculate MAPE

def MAPE(y\_true, y\_pred):

mape = np.mean(np.abs((y\_true - y\_pred) / y\_true))\*100

return mape

mape\_dt = MAPE(test.iloc[:,8], predictions\_DT)

print("error percentage for data - Decision tree", mape\_dt)

##Random Forest

from sklearn.ensemble import RandomForestRegressor

#Random Forest for regression

fit\_RF = RandomForestRegressor(bootstrap=True, max\_depth=None, random\_state=0 ,n\_estimators=50).fit(train.iloc[:,0:8], train.iloc[:,8])

#Apply model on test data

predictions\_RF = fit\_RF.predict(test.iloc[:,0:8])

mape\_rf = MAPE(test.iloc[:,8], predictions\_RF)

print("error percentage for data - Random Forest", mape\_rf)

##Linear Regression

#Import libraries for LR

import statsmodels.api as sm

# Train the model using the training sets

linear\_model = sm.OLS(train.iloc[:,8], train.iloc[:,0:8]).fit()

# Print out the statistics

linear\_model.summary()

# make the predictions by the model

predictions\_LR = linear\_model.predict(test.iloc[:,0:8])

#Calculate MAPE

mape\_lr = MAPE(test.iloc[:,8], predictions\_LR)

print("error percentage for data - Linear Regression ", mape\_lr)

#for outliers removed data

#Divide data into train and test

day\_data\_wo = df

day\_data\_wo = day\_data\_wo.drop(["instant", "dteday","workingday","weathersit", "atemp", "casual", "registered"], axis=1)

train\_wo, test\_wo = train\_test\_split(day\_data\_wo, test\_size=0.1)

#Decision tree for regression

fit\_DT\_wo = DecisionTreeRegressor(max\_depth=2).fit(train\_wo.iloc[:,0:8], train\_wo.iloc[:,8])

#Apply model on test data

predictions\_DT\_wo = fit\_DT\_wo.predict(test\_wo.iloc[:,0:8])

mape\_dt\_wo = MAPE(test\_wo.iloc[:,8], predictions\_DT\_wo)

print("error percentage for data without outlier - Decision tree", mape\_dt\_wo)

#Random Forest for regression

fit\_RF\_wo = RandomForestRegressor(bootstrap=True, max\_depth=None, random\_state=0 ,n\_estimators=50).fit(train\_wo.iloc[:,0:8], train\_wo.iloc[:,8])

#Apply model on test data

predictions\_RF\_wo = fit\_RF\_wo.predict(test\_wo.iloc[:,0:8])

mape\_rf\_wo = MAPE(test.iloc[:,8], predictions\_RF\_wo)

print("error percentage for data without outlier - Random Forest", mape\_rf\_wo)

# Train the model using the training sets

linear\_model\_wo = sm.OLS(train\_wo.iloc[:,8], train\_wo.iloc[:,0:8]).fit()

# make the predictions by the model

predictions\_LR\_wo = linear\_model\_wo.predict(test\_wo.iloc[:,0:8])

#Calculate MAPE

mape\_lr\_wo = MAPE(test.iloc[:,8], predictions\_LR)

print("error percentage for data without outliers - Linear Regression ", mape\_lr\_wo)

**R file:**

****

**Python File:**

