**BIKE RENTING**

Dudde Dharmatej

09 March 2020

**Contents:**

1. Introduction
   1. Problem statement………………………………………3
   2. Data……………………………………………………..3
2. Methodology
   1. Exploratory data analysis...……………………………..4
   2. Data preprocessing……………………………………...5
      1. Missing Value analysis……………………….….5
      2. Outlier analysis…………………………………..6
      3. Feature selection…………………………………7
   3. Modeling
      1. Linear regression…………………………………8
      2. Decision Trees……………………………………11
      3. Random Forest…………………………………...12
3. Conclusion

3.1 Model Evaluation………………………….............……13

* 1. Error Metric………………………………………..……14
  2. Model selection…………………………………………16

1. Visualizations…………………………………………………..16
2. Code…………………………………………………………….21
3. **Introduction:**
   1. **Problem Statement:**

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings.

* 1. **Data:**

The details of variable present 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)

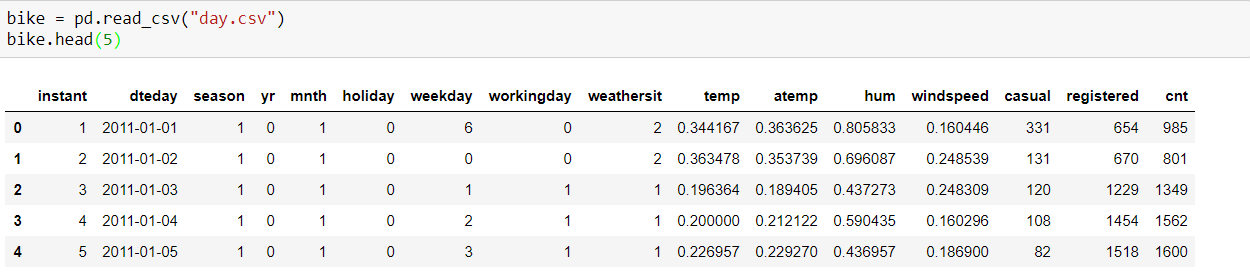
windspeed: Normalized wind speed. The values are divided to 67 (max)

casual: count of casual users

registered: count of registered users

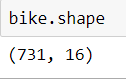
cnt: count of total rental bikes including both casual and registered

Sample Data:

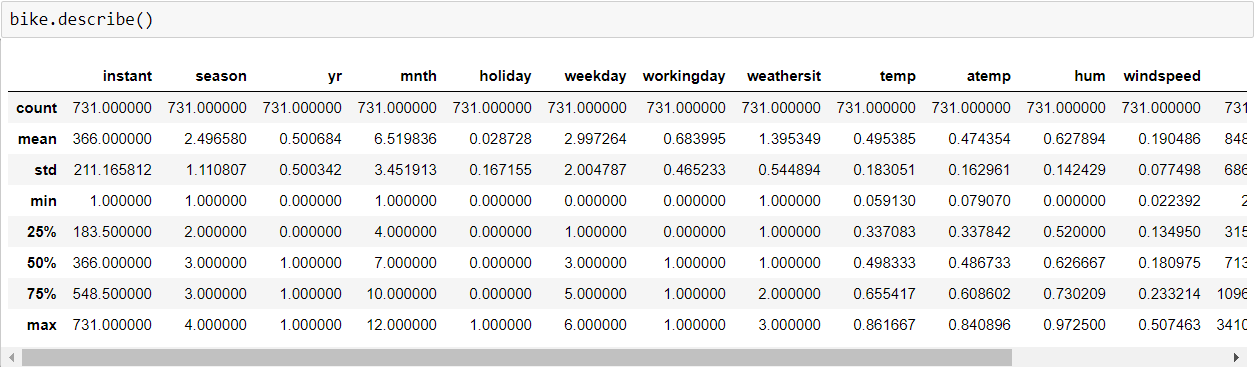
****

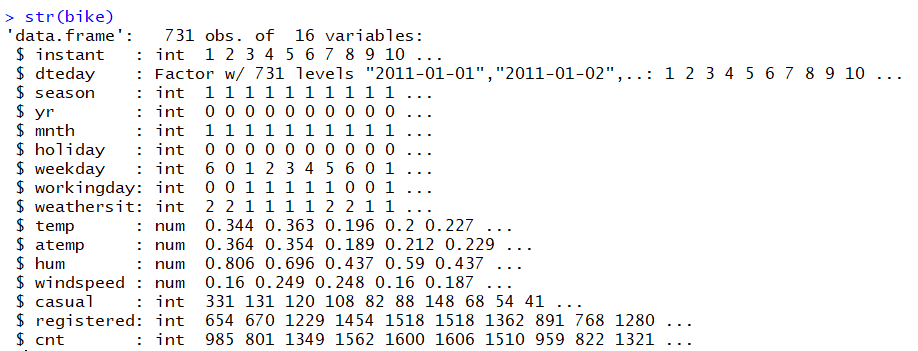
1. **Methodology:**
   1. **Exploratory data analysis:**

Given data have dimensions of 731 rows and 15 independent columns and 1 dependent column.



Structure of given data can be seen below





* 1. **Data Preprocessing:**

We can see variables like 'mnth', ‘holiday', 'weekday', 'weathersit' are categorical but taken as numeric values**.** Hence converted the required variables from Integer to categorical.

Deleted instant, dteday variables as they are nothing but an index. Omitted registered and casual variable as sum of registered and casual is the total count that is what we have to predict.

* + 1. **Missing value analysis:**

Missing value analysis is done to check is there any missing value present in given dataset. Missing values can be easily treated using various methods like mean, median method, knn method to impute missing value.

In R function(x){sum(is.na(x))} is the function used to check the sum of missing values.

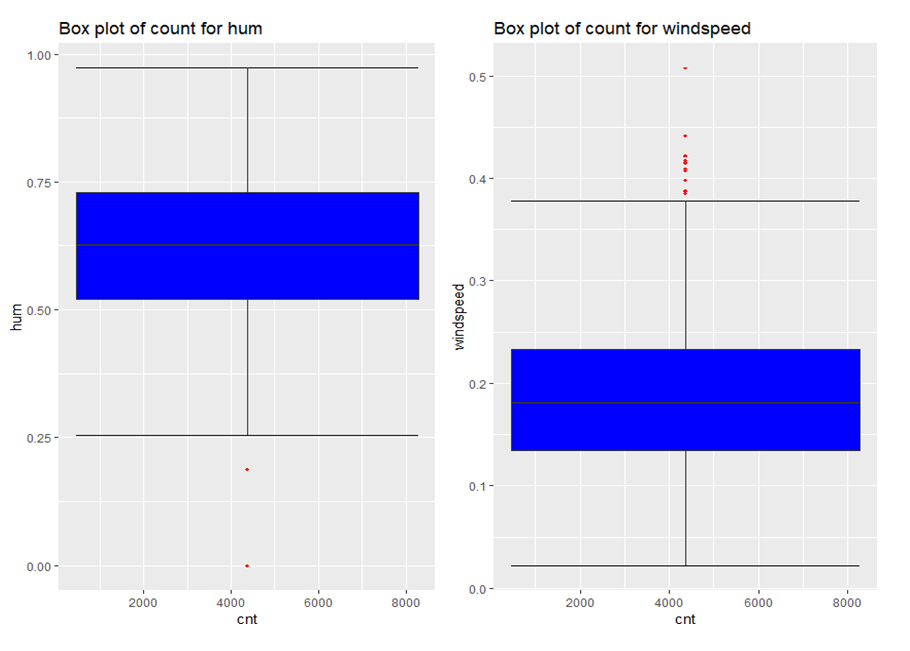
In python bike.isnull().sum() is used to detect any missing value

We found that there are no missing values in the given data



* + 1. **Outlier analysis:**

Outlier analysis is done to handle all inconsistent observations present in given dataset. Outlier analysis can only be done on continuous variable. Using box plot analysis we have found outliers in numeric variables like “hum” and “windspeed”



**Treating outliers:**

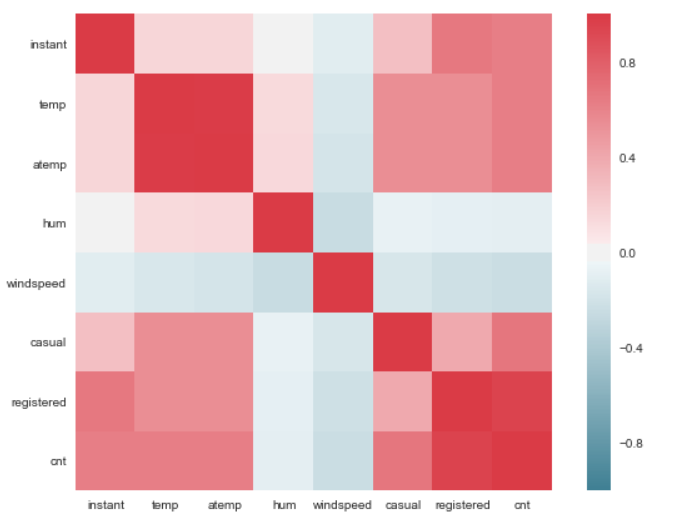
As there are fewer no of outliers compared to the number of observations we have decided to remove the outliers from given data.

Dimensions of the data after removing outliers is 655 rows and 12 variables

* + 1. **Feature selection:**

Feature selection analysis is done to select subsets of relevant features (variables, predictors) to be in model construction.

As our target variable is continuous its good to go for correlation check.



In above visualization we can see that only 2 variables are highly correlated with each other temp and atemp. Dark Red color represent highly correlated and light color represent very less correlated so we have a choice to remove either temp or atemp because these variables contains nearly equal information.

So I have removed atemp variable from dataset.

Feature scaling includes two functions normalization and standardization. It is done reduce unwanted variation either within or between variables and to bring all of the variables into proportion with one another.

In given dataset all numeric values are already present in normalized form.

**2.3 Modeling:**

In this case we have to predict the count of bike renting according to environmental and seasonal condition. So the target variable here is a continuous variable. For Continuous we can use various Regression models. Model having less error rate and more accuracy will be our final model.

**2.3.1: Linear regression:**

For linear regression model we have divided the categorical variables containing more than 2 classes into dummy variable. So that all categorical variable should be in binary classes form.

Further the data is divided into train and test with 80 % train data and 20 % test data using random sampling.

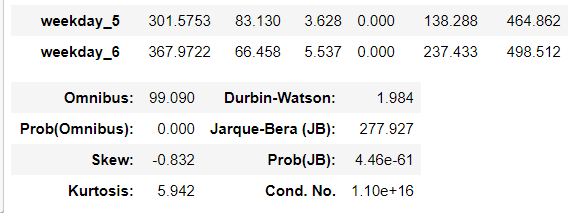
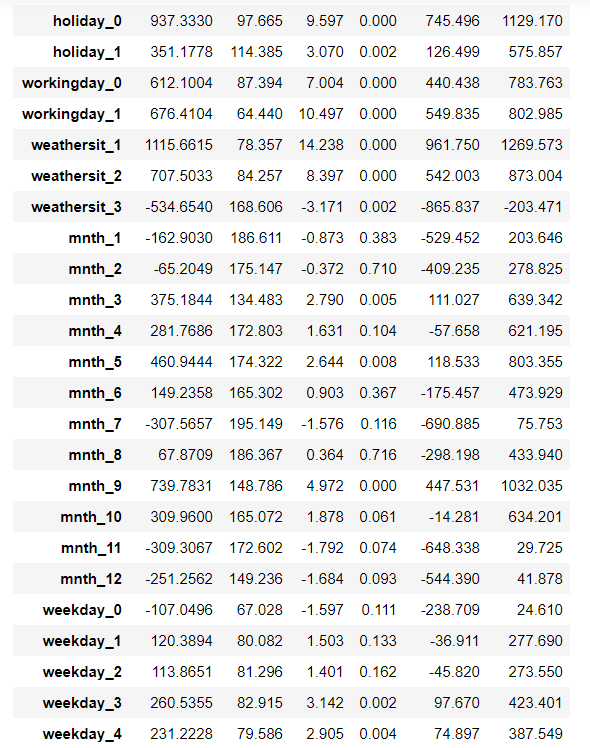
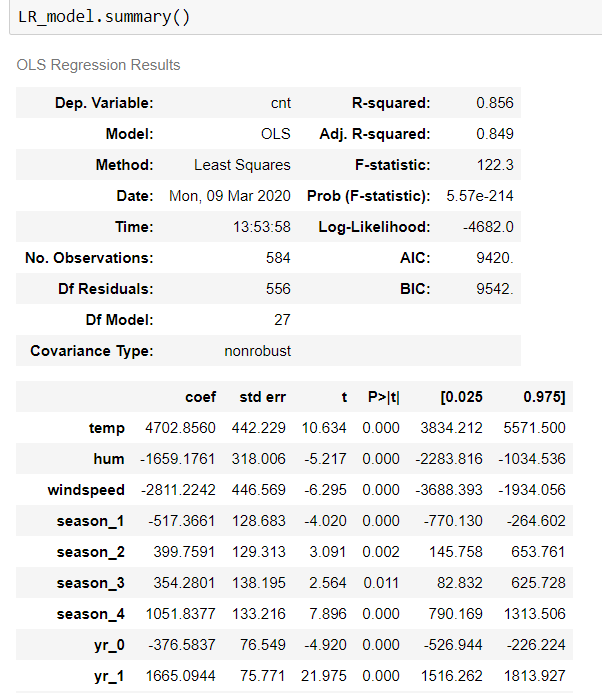
In R:

Lm model = lm(cnt ~., data = bike\_train\_lr)

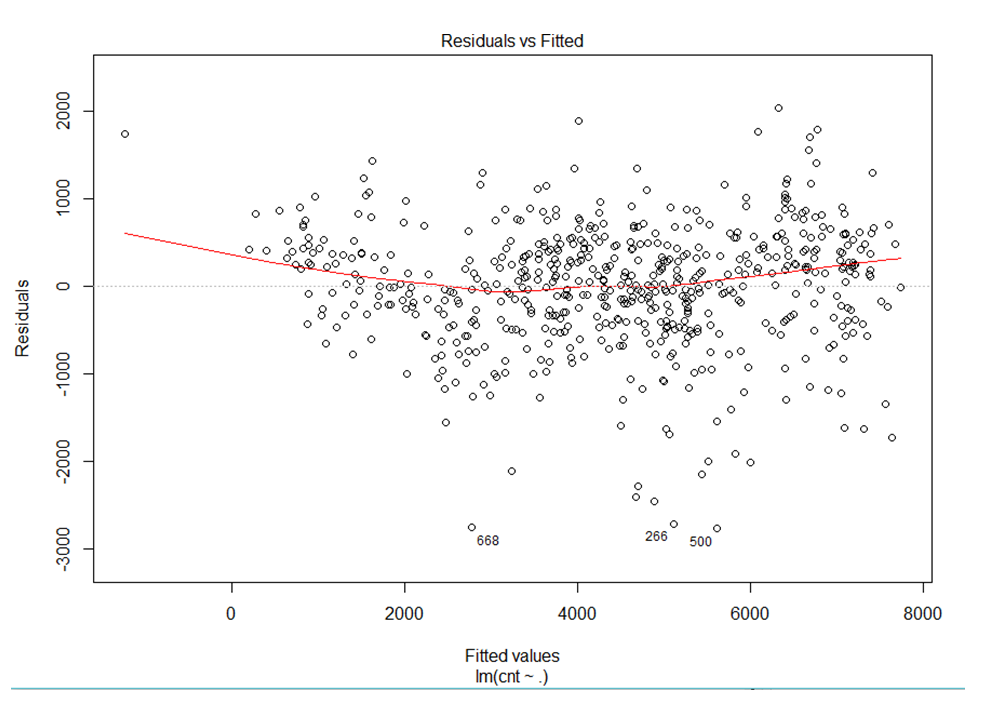
In Python:

LR\_model = sm.OLS(bike\_trainlr.iloc[:,35], bike\_trainlr.iloc[:,0:35]).fit()

Summary of logistic regression model:



**Regression graph:**



**2.3.2 Decision Tree:**

For Decision tree model we have divided the dataset into train and test part using random sampling. Where train contains 80% data of data set and test contains 20% data

In R:

lm\_model = lm(cnt ~., data = bike\_train\_lr)

In Python:

DT\_model = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:10],train.iloc[:,10])

**2.3.3 Random Forest:**

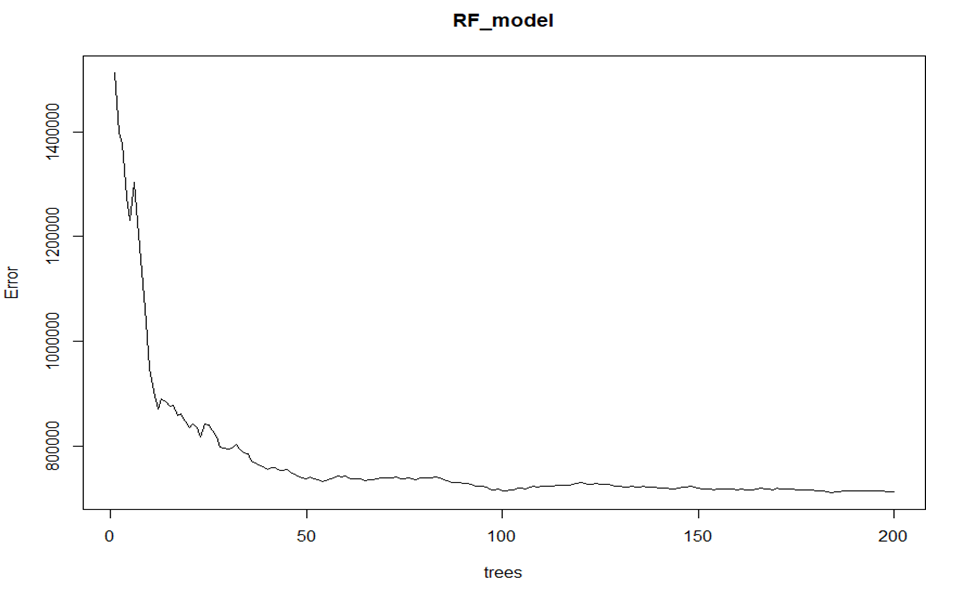
For this model we have divided the dataset into train and test part using random sampling where train contains 80% data of data set and test contains 20% data with 200 trees

In R:

RF\_model = randomForest(cnt.y ~ ., train, importance = TRUE, ntree = 200)

In Python:

RF\_model = RandomForestRegressor(n\_estimators = 200).fit(train.iloc[:,0:10], train.iloc[:,10])



The above graph represents the curve of error rate as the number of trees increases. After 200 trees the error rate reaches to be constant.

1. **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 Renting, 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 error measure.

**3.2 Error Metric:**

**Mean Absolute Percentage Error (MAPE)**

MAPE 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 models and the reason to select MAPE is describes the error in percentage form which is easy to understand.

MAPE formula:

[https://www.statisticshowto.datasciencecentral.com/wp-content/uploads/2017/09/mape.jpeg](https://www.statisticshowto.datasciencecentral.com/wp-content/uploads/2017/09/mape.jpeg)

Where At is the actual value and Ft is the forecast value.

In R:

MAPE function

MAPE = function(y, yhat){

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

}

**MAPE values:**

MAPE for linear regression

MAPE(bike\_test\_lr[,768], predictions\_LR)

#108.3069

MAPE for decision tree regression

MAPE(test[,13], predictions\_DT)

#11.48524

MAPE for random forest regression

MAPE(test[,13], predictions\_RF)

#6.028095

In Python:

MAPE function

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

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

return mape

**MAPE values:**

MAPE for linear regression

MAPE(bike\_testlr.iloc[:,35], predictions\_LR)

#134.99458926507563

MAPE for decision tree regression

MAPE(test.iloc[:,10], predictions\_DT)

#27.98253318462642

MAPE for random forest regression

MAPE(test.iloc[:,10],RF\_Predictions)

#13.884538218183765

Where predictions\_LR are predicted values from linear regression model

predictions\_DT are predicted values from decision tree model predictions\_RF are predicted values from random forest model

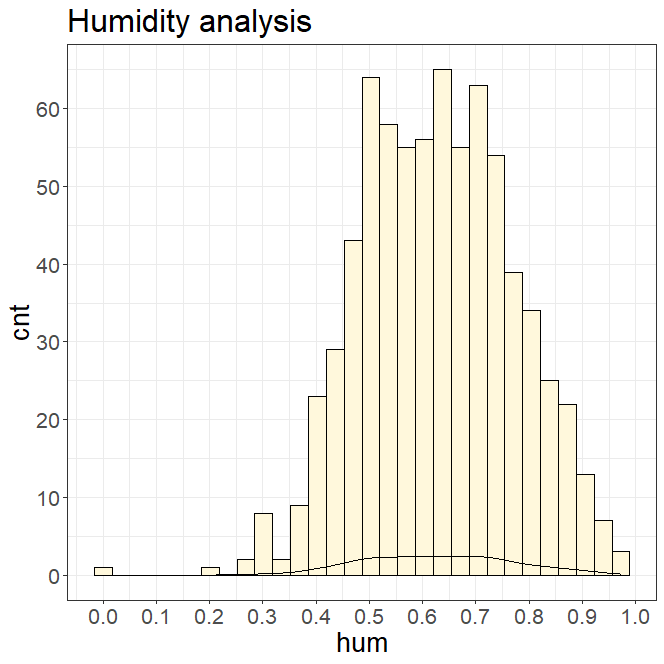
**3.3. Model selection:**

We can see that in both R and Python Random forest model performs best out of all the models. So random forest model is selected with 6% error in R and with 13% error in python.

Extracted predicted values of random forest model are saved with .csv file format.

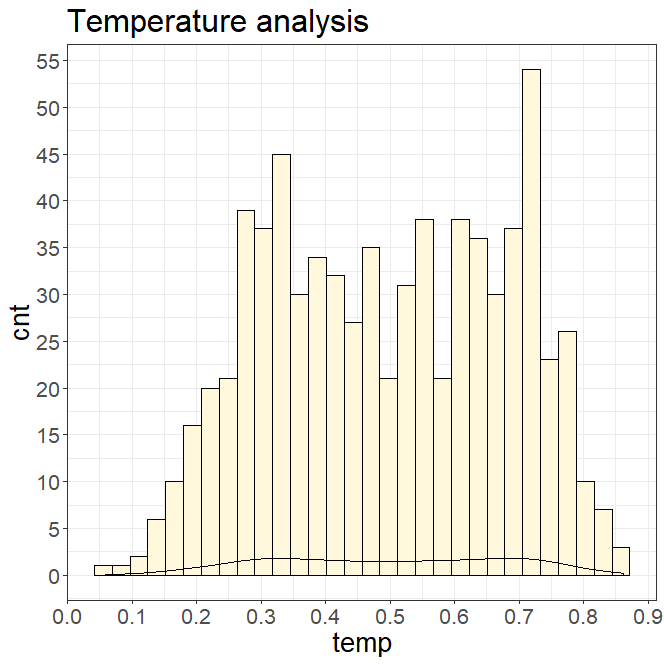
**4.Visualizations:**

**Humidity Vs Count:**



As per the above graph we can see it is normally distributed indicating that with extreme humidity weather conditions the number of bike rentals are very low and with mediocre humidity the bike rental number varies normally.

**Temperature Vs Count:**



As per the above graph we can see that it is similar to humidity that in extreme temperature conditions the bike rentals are less and vice versa.

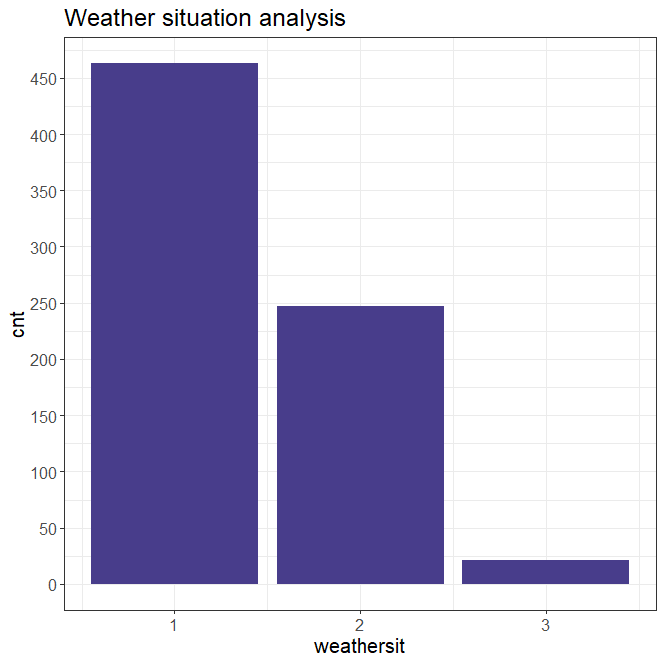
**Weather Vs count:**

1: Clear, Few clouds, Partly cloudy, Partly cloudy

2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

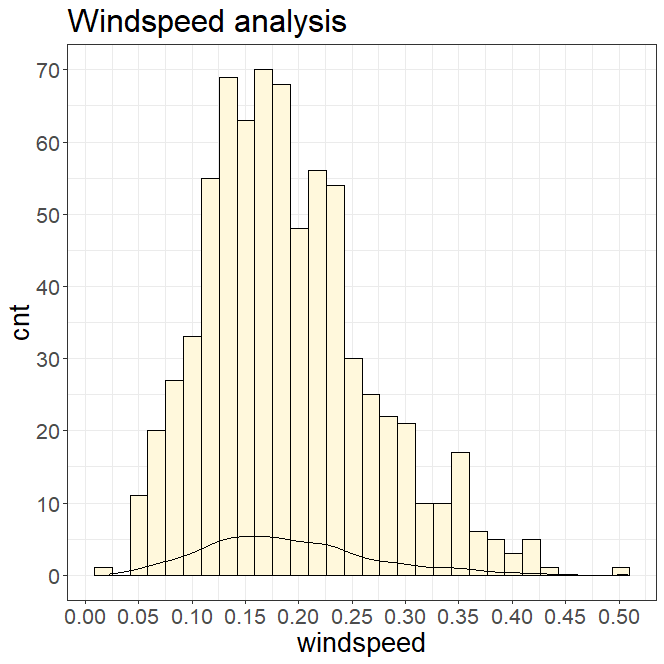
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered

clouds



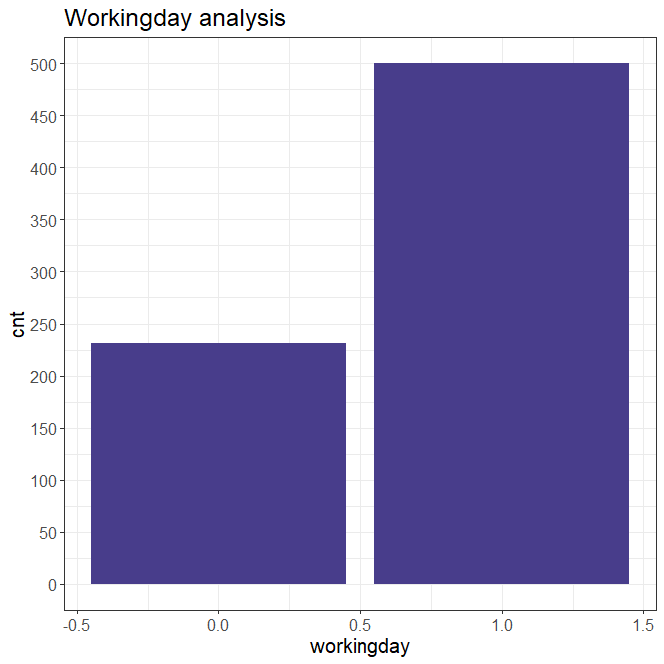
From the above graph we can see that clear weather conditions are favorable for higher bike rental count.

**Windspeed Vs Count:**

****

From the above grapth we can see that high windspeed conditions are unfavourable for bike renting which is alos common sense.

**Workingday Vs Count:**

****

From the above graph we can clearly see that working days have huge impact on count of bike renting one reason could be that people’s work commute methods is bikes and people might prefer other options on a holiday.

**5. Code:**

**R Code:**

#the objective of this case is to predict the bike rental count on daily based on the environmental and seasonal settings

#clearing all the existing objects

rm(list=ls())

#set working directory

setwd("C:/Users/Sunny/Desktop/edwisor/Bike rental")

getwd()

#install required packages

install.packages(c("ggplot2","corrgram","DMwR","caret","randomForest","C50","MASS","rpart","dplyr","plyr","reshape","ggplot2","data.table"))

library("ggplot2")

library("corrgram")

library("DMwR")

library("caret")

library("randomForest")

library("C50")

library("MASS")

library("rpart")

library("dplyr")

library("plyr")

library("reshape")

library("ggplot2")

library("data.table")

#reading the data file

bike = read.csv("day.csv", header = TRUE, sep = ",")

#sample data

head(bike)

#dimensions of data

dim(bike)

#Summarty of the data

summary(bike)

#structure of the data

str(bike)

#we can see variables like 'mnth',holiday','weekday','weathersit' are

#catagorical but taken as numeric values

#hence converting

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

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

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

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

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

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

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

str(bike)

#Nummeric vaiables like 'temp','atem','hum','windspeed' are already Normalized

#finding the missing values

missing\_val = data.frame(apply(bike,2,function(x){sum(is.na(x))}))

missing\_val

#no missing values found

#finding outliers

#selecting numeric data

numeric\_index = sapply(bike,is.numeric)

numeric\_data = bike[,numeric\_index]

cnames = colnames(numeric\_data)

dim(numeric\_data)

#loop to find outliers

for (i in 1:length(cnames))

{

assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = "cnt"), data = subset(bike))+

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="cnt")+

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

}

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

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

#loop to remove all outliers from numeric data

numeric\_data\_no\_outliers = numeric\_data

for(i in cnames){

print(i)

val = numeric\_data\_no\_outliers[,i][numeric\_data\_no\_outliers[,i] %in% boxplot.stats(numeric\_data\_no\_outliers[,i])$out]

print(length(val))

numeric\_data\_no\_outliers = numeric\_data\_no\_outliers[which(!numeric\_data\_no\_outliers[,i] %in% val),]

}

dim(numeric\_data\_no\_outliers)

#merging and replacing data with no outlier numeric data

bike\_ready = merge(x=bike, y=numeric\_data\_no\_outliers, by="instant", all.y=TRUE)

dim(bike\_ready)

#Feature Selection and dimension reduction

# Correlation Plot

corrgram(bike[,numeric\_index], order = F,upper.panel=panel.pie, text.panel=panel.txt,

main = "Correlation Plot")

#we see that temp and atemp are highly correlated and hence removing atemp column and

#removing other redundent numeric coulumns after merging

bike = subset(bike\_ready,select = -c(atemp.x,atemp.y,temp.x,hum.x,windspeed.x,casual.x,registered.x,cnt.x))

dim(bike)

#Modeling

#Linear Regression

#converting multilevel categorical variable into binary dummy variable

str(bike)

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

bike\_lr=bike[,cnames]

dim(bike\_lr)

target=data.frame(bike$cnt.y)

dim(target)

names(target)[1]="cnt"

bike\_lr <- fastDummies::dummy\_cols(bike\_lr)

dim(bike\_lr)

bike\_lr= subset(bike\_lr,select = -c(dteday,season,mnth,weekday,weathersit))

dim(bike\_lr)

data = cbind(bike\_lr,bike)

View(data)

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

dim(data)

bike\_lr=cbind(data,target)

dim(bike\_lr)

#dividind data into test and train

train\_index = sample(1:nrow(bike\_lr), 0.8 \* nrow(bike\_lr))

bike\_train\_lr = bike\_lr[train\_index,]

bike\_test\_lr = bike\_lr[-train\_index,]

#building Linear regression model

lm\_model = lm(cnt ~., data = bike\_train\_lr)

summary(lm\_model)

#predicting using logistic regression model

predictions\_LR = predict(lm\_model,bike\_test\_lr[,-768])

bike = subset(bike,select = -c(instant,dteday))

dim(bike)

#Decision tree regression '

#dividind data into test and train

train\_index = sample(1:nrow(bike), 0.8 \* nrow(bike))

train = bike[train\_index,]

test = bike[-train\_index,]

#building decision tree model

DT\_model = rpart(cnt.y ~ ., data = train, method = "anova")

summary(DT\_model)

#predicting using decision tree model

predictions\_DT = predict(DT\_model, test[,-13])

#building Random Forest Model

RF\_model = randomForest(cnt.y ~ ., train, importance = TRUE, ntree = 200)

summary(RF\_model)

#predicting using random forest model

predictions\_RF = predict(RF\_model, test[,-13])

plot(RF\_model)

#evaluating MAPE value and finding the best model

MAPE = function(y, yhat){

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

}

MAPE(bike\_test\_lr[,768], predictions\_LR)

#108.3069

MAPE(test[,13], predictions\_DT)

#11.48524

MAPE(test[,13], predictions\_RF)

#6.028095

#Visualizations

library("scales")

#season vs cnt

ggplot(bike, aes\_string(x = bike$season)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("season") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Seasonal analysis") + theme(text=element\_text(size=15))

#mnth vs cnt

ggplot(bike, aes\_string(x = bike$mnth)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("mnth") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Monthly analysis") + theme(text=element\_text(size=15))

#holiday vs cnt

ggplot(bike, aes\_string(x = bike$holiday)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("holiday") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Holiday analysis") + theme(text=element\_text(size=15))

#weekday vs cnt

ggplot(bike, aes\_string(x = bike$weekday)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("weekday") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Weekly analysis") + theme(text=element\_text(size=15))

#workingday vs cnt

ggplot(bike, aes\_string(x = bike$workingday)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("workingday") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Workingday analysis") + theme(text=element\_text(size=15))

#weathersit vs cnt

ggplot(bike, aes\_string(x = bike$weathersit)) +

geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +

xlab("weathersit") + ylab('cnt') + scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

ggtitle("Weather situation analysis") + theme(text=element\_text(size=15))

#temp vs cnt

ggplot(bike, aes\_string(x = bike$temp)) +

geom\_histogram(fill="cornsilk", colour = "black") + geom\_density() +

scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

scale\_x\_continuous(breaks=pretty\_breaks(n=10))+

theme\_bw() + xlab("temp") + ylab("cnt") + ggtitle("Temperature analysis") +

theme(text=element\_text(size=20))

#hum vs cnt

ggplot(bike, aes\_string(x = bike$hum)) +

geom\_histogram(fill="cornsilk", colour = "black") + geom\_density() +

scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

scale\_x\_continuous(breaks=pretty\_breaks(n=10))+

theme\_bw() + xlab("hum") + ylab("cnt") + ggtitle("Humidity analysis") +

theme(text=element\_text(size=20))

#windspeed vs cnt

ggplot(bike, aes\_string(x = bike$windspeed)) +

geom\_histogram(fill="cornsilk", colour = "black") + geom\_density() +

scale\_y\_continuous(breaks=pretty\_breaks(n=10)) +

scale\_x\_continuous(breaks=pretty\_breaks(n=10))+

theme\_bw() + xlab("windspeed") + ylab("cnt") + ggtitle("Windspeed analysis") +

theme(text=element\_text(size=20))

**Python code:**

#importing required packages for initial data analysis

​

import os

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

​

#Set working directory

os.chdir("C:\\Users\\Sunny\\Desktop\\edwisor\\Bike rental")

os.getcwd()

'C:\\Users\\Sunny\\Desktop\\edwisor\\Bike rental'

#reading the data

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

​

#data sample

bike.head(5)

instant dteday season yr mnth holiday weekday workingday weathersit temp atemp hum windspeed casual registered cnt

0 1 2011-01-01 1 0 1 0 6 0 2 0.344167 0.363625 0.805833 0.160446 331 654 985

1 2 2011-01-02 1 0 1 0 0 0 2 0.363478 0.353739 0.696087 0.248539 131 670 801

2 3 2011-01-03 1 0 1 0 1 1 1 0.196364 0.189405 0.437273 0.248309 120 1229 1349

3 4 2011-01-04 1 0 1 0 2 1 1 0.200000 0.212122 0.590435 0.160296 108 1454 1562

4 5 2011-01-05 1 0 1 0 3 1 1 0.226957 0.229270 0.436957 0.186900 82 1518 1600

#dimensions of the data

bike.shape

(731, 16)

#description of data

bike.describe()

instant season yr mnth holiday weekday workingday weathersit temp atemp hum windspeed casual registered cnt

count 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000 731.000000

mean 366.000000 2.496580 0.500684 6.519836 0.028728 2.997264 0.683995 1.395349 0.495385 0.474354 0.627894 0.190486 848.176471 3656.172367 4504.348837

std 211.165812 1.110807 0.500342 3.451913 0.167155 2.004787 0.465233 0.544894 0.183051 0.162961 0.142429 0.077498 686.622488 1560.256377 1937.211452

min 1.000000 1.000000 0.000000 1.000000 0.000000 0.000000 0.000000 1.000000 0.059130 0.079070 0.000000 0.022392 2.000000 20.000000 22.000000

25% 183.500000 2.000000 0.000000 4.000000 0.000000 1.000000 0.000000 1.000000 0.337083 0.337842 0.520000 0.134950 315.500000 2497.000000 3152.000000

50% 366.000000 3.000000 1.000000 7.000000 0.000000 3.000000 1.000000 1.000000 0.498333 0.486733 0.626667 0.180975 713.000000 3662.000000 4548.000000

75% 548.500000 3.000000 1.000000 10.000000 0.000000 5.000000 1.000000 2.000000 0.655417 0.608602 0.730209 0.233214 1096.000000 4776.500000 5956.000000

max 731.000000 4.000000 1.000000 12.000000 1.000000 6.000000 1.000000 3.000000 0.861667 0.840896 0.972500 0.507463 3410.000000 6946.000000 8714.000000

#data types of individual variables

bike.dtypes

instant int64

dteday object

season int64

yr int64

mnth int64

holiday int64

weekday int64

workingday int64

weathersit int64

temp float64

atemp float64

hum float64

windspeed float64

casual int64

registered int64

cnt int64

dtype: object

#counting no.of observations of each variable

bike.count()

#all coulumns have the same no.of rows indicting there are no missing values

​

instant 731

dteday 731

season 731

yr 731

mnth 731

holiday 731

weekday 731

workingday 731

weathersit 731

temp 731

atemp 731

hum 731

windspeed 731

casual 731

registered 731

cnt 731

dtype: int64

#also Finding the missing values in a more efficiengt way.

print(bike.isnull().sum())

#no missing values

instant 0

dteday 0

season 0

yr 0

mnth 0

holiday 0

weekday 0

workingday 0

weathersit 0

temp 0

atemp 0

hum 0

windspeed 0

casual 0

registered 0

cnt 0

dtype: int64

#converting required data types

bike['season']= bike['season'].astype('category')

bike['yr']=bike['yr'].astype('category')

bike['mnth']=bike['mnth'].astype('category')

bike['holiday']=bike['holiday'].astype('category')

bike['workingday']=bike['workingday'].astype('category')

bike['weekday']=bike['weekday'].astype('category')

bike['weathersit']=bike['weathersit'].astype('category')

bike.dtypes

instant int64

dteday object

season category

yr category

mnth category

holiday category

weekday category

workingday category

weathersit category

temp float64

atemp float64

hum float64

windspeed float64

casual int64

registered int64

cnt int64

dtype: object

#outlier analysis

#boxplot for temp variable

plt.boxplot(bike['temp'], showfliers=True)

​

#sns.set(style="whitegrid")

#sns.boxplot(x=bike['temp'],orient ='h')

sns.plt.show()

#boxplot for atemp

#boxplot for atemp

plt.boxplot(bike['atemp'], showfliers=True)

​

sns.plt.show()

#boxplot for humidity

plt.boxplot(bike['hum'], showfliers=True)

​

sns.plt.show()

#boxplot for windspeed

plt.boxplot(bike['windspeed'], showfliers=True)

​

sns.plt.show()

#we dont see any outliers present in the normalized numeric variables

#lets find correlation between variables

​

bike\_corr = bike

​

f, ax = plt.subplots(figsize=(10,7))

​

#Generate correlation matrix

corr = bike\_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)

sns.plt.show()

​

​

#we see that temp and atemp are highly correlated

#hence dropping atemp variable from data

bike = bike.drop(['atemp'], axis=1)

​

bike.shape

(731, 15)

#also dropping other unnecessary variables ,like instant, dteday, casual and registered

#casual + registered = cnt

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

​

bike.shape

bike.describe()

temp hum windspeed cnt

count 731.000000 731.000000 731.000000 731.000000

mean 0.495385 0.627894 0.190486 4504.348837

std 0.183051 0.142429 0.077498 1937.211452

min 0.059130 0.000000 0.022392 22.000000

25% 0.337083 0.520000 0.134950 3152.000000

50% 0.498333 0.626667 0.180975 4548.000000

75% 0.655417 0.730209 0.233214 5956.000000

max 0.861667 0.972500 0.507463 8714.000000

bike.dtypes

season category

yr category

mnth category

holiday category

weekday category

workingday category

weathersit category

temp float64

hum float64

windspeed float64

cnt int64

dtype: object

#importing required packages for model building

​

from scipy.stats import chi2\_contingency

from random import randrange, uniform

import datetime as dt

from sklearn.cross\_validation import train\_test\_split

from sklearn.tree import DecisionTreeRegressor

import statsmodels.api as sm

from sklearn.ensemble import RandomForestRegressor

​

​

#model building

#linear regression

​

bike\_lr=bike.copy()

bike.shape

(731, 11)

bike\_lr.shape

(731, 11)

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

cat\_names

['season', 'yr', 'holiday', 'workingday', 'weathersit', 'mnth', 'weekday']

#dummify

​

for i in cat\_names:

temp = pd.get\_dummies(bike\_lr[i], prefix = i)

bike\_lr = bike\_lr.join(temp)

bike\_lr.shape

(731, 43)

fields\_to\_drop = ['yr', 'season','holiday','workingday', 'weathersit', 'weekday', 'mnth','cnt']

bike\_lr = bike\_lr.drop(fields\_to\_drop, axis=1)

bike\_lr.shape

(731, 35)

bike\_lr=bike\_lr.join(bike['cnt'])

​

bike\_lr.shape

(731, 36)

#predicting values using logisting regression model

#splitting data in to train and test

bike\_trainlr, bike\_testlr = train\_test\_split(bike\_lr, test\_size=0.2)

​

#building logistic regression

LR\_model = sm.OLS(bike\_trainlr.iloc[:,35], bike\_trainlr.iloc[:,0:35]).fit()

​

#predicting values using logisting regression model

predictions\_LR = LR\_model.predict(bike\_testlr.iloc[:,0:35])

​

LR\_model.summary()

OLS Regression Results

Dep. Variable: cnt R-squared: 0.856

Model: OLS Adj. R-squared: 0.849

Method: Least Squares F-statistic: 122.3

Date: Mon, 09 Mar 2020 Prob (F-statistic): 5.57e-214

Time: 13:53:58 Log-Likelihood: -4682.0

No. Observations: 584 AIC: 9420.

Df Residuals: 556 BIC: 9542.

Df Model: 27

Covariance Type: nonrobust

coef std err t P>|t| [0.025 0.975]

temp 4702.8560 442.229 10.634 0.000 3834.212 5571.500

hum -1659.1761 318.006 -5.217 0.000 -2283.816 -1034.536

windspeed -2811.2242 446.569 -6.295 0.000 -3688.393 -1934.056

season\_1 -517.3661 128.683 -4.020 0.000 -770.130 -264.602

season\_2 399.7591 129.313 3.091 0.002 145.758 653.761

season\_3 354.2801 138.195 2.564 0.011 82.832 625.728

season\_4 1051.8377 133.216 7.896 0.000 790.169 1313.506

yr\_0 -376.5837 76.549 -4.920 0.000 -526.944 -226.224

yr\_1 1665.0944 75.771 21.975 0.000 1516.262 1813.927

holiday\_0 937.3330 97.665 9.597 0.000 745.496 1129.170

holiday\_1 351.1778 114.385 3.070 0.002 126.499 575.857

workingday\_0 612.1004 87.394 7.004 0.000 440.438 783.763

workingday\_1 676.4104 64.440 10.497 0.000 549.835 802.985

weathersit\_1 1115.6615 78.357 14.238 0.000 961.750 1269.573

weathersit\_2 707.5033 84.257 8.397 0.000 542.003 873.004

weathersit\_3 -534.6540 168.606 -3.171 0.002 -865.837 -203.471

mnth\_1 -162.9030 186.611 -0.873 0.383 -529.452 203.646

mnth\_2 -65.2049 175.147 -0.372 0.710 -409.235 278.825

mnth\_3 375.1844 134.483 2.790 0.005 111.027 639.342

mnth\_4 281.7686 172.803 1.631 0.104 -57.658 621.195

mnth\_5 460.9444 174.322 2.644 0.008 118.533 803.355

mnth\_6 149.2358 165.302 0.903 0.367 -175.457 473.929

mnth\_7 -307.5657 195.149 -1.576 0.116 -690.885 75.753

mnth\_8 67.8709 186.367 0.364 0.716 -298.198 433.940

mnth\_9 739.7831 148.786 4.972 0.000 447.531 1032.035

mnth\_10 309.9600 165.072 1.878 0.061 -14.281 634.201

mnth\_11 -309.3067 172.602 -1.792 0.074 -648.338 29.725

mnth\_12 -251.2562 149.236 -1.684 0.093 -544.390 41.878

weekday\_0 -107.0496 67.028 -1.597 0.111 -238.709 24.610

weekday\_1 120.3894 80.082 1.503 0.133 -36.911 277.690

weekday\_2 113.8651 81.296 1.401 0.162 -45.820 273.550

weekday\_3 260.5355 82.915 3.142 0.002 97.670 423.401

weekday\_4 231.2228 79.586 2.905 0.004 74.897 387.549

weekday\_5 301.5753 83.130 3.628 0.000 138.288 464.862

weekday\_6 367.9722 66.458 5.537 0.000 237.433 498.512

Omnibus: 99.090 Durbin-Watson: 1.984

Prob(Omnibus): 0.000 Jarque-Bera (JB): 277.927

Skew: -0.832 Prob(JB): 4.46e-61

Kurtosis: 5.942 Cond. No. 1.10e+16

#dividing data into train and test

train, test = train\_test\_split(bike, test\_size=0.2)

​

​

#building Decision tree model

DT\_model = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:10], train.iloc[:,10])

​

#predicting values using decision tree model

predictions\_DT = DT\_model.predict(test.iloc[:,0:10])

​

​

#building random forest

RF\_model = RandomForestRegressor(n\_estimators = 200).fit(train.iloc[:,0:10], train.iloc[:,10])

​

#predicting values using random forest model

RF\_Predictions = RF\_model.predict(test.iloc[:,0:10])

​

#MAPE for linear regression

MAPE(bike\_testlr.iloc[:,35], predictions\_LR)

#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 linear regression

MAPE(bike\_testlr.iloc[:,35], predictions\_LR)

​134.99458926507563

#MAPE for decision tree regression

MAPE(test.iloc[:,10], predictions\_DT)

​27.98253318462642

#MAPE for random forest regression

MAPE(test.iloc[:,10],RF\_Predictions)

13.884538218183765

#writing predictited values to csv file

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

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

​

result.to\_csv("bike\_rental\_RF\_python.csv",index=False)