

Predicting Bike Rental COunt



August 28, 2018

ashish sharma

Contents

[**1** **Introduction** 2](#_Toc523192869)

[**1.1** **Problem Statement** 2](#_Toc523192870)

[**1.2** **Data** 2](#_Toc523192871)

[**2** **Methodology** 4](#_Toc523192872)

[**2.1** **Pre-Processing** 4](#_Toc523192873)

[**2.1.1** **Variable Identification** 4](#_Toc523192874)

[**2.1.2** **Univariate Analysis** 4](#_Toc523192875)

[**2.1.3** **Bi-variate analysis** 5](#_Toc523192876)

[**2.1.4** **Missing Values Analysis** 7](#_Toc523192877)

[**2.1.5** **Outlier Analysis** 7](#_Toc523192878)

[**2.1.6** **Feature Engineering** 8](#_Toc523192879)

[**2.2** **Modelling** 10](#_Toc523192880)

[**2.2.1** **Model Selection** 10](#_Toc523192881)

[**2.2.2** **Multiple Linear Regression** 10](#_Toc523192882)

[**2.2.3** **Random Forest Model** 11](#_Toc523192883)

[**3** **Conclusion** 12](#_Toc523192884)

[**3.1** **Model Evaluation** 12](#_Toc523192885)

[**3.1.1** **Root Mean Squared Error Loss(RMSE)** 12](#_Toc523192886)

[**3.1.2** **Root Mean Squared Logarithmic Error Loss(RMSLE)** 12](#_Toc523192887)

[**3.2** **Model Selection** 12](#_Toc523192888)

[**Appendix A - Extra Figures** 13](#_Toc523192889)

[**Appendix B - R Code** 16](#_Toc523192890)

[**univariate analysis** 16](#_Toc523192891)

[**Multivariate Analysis** 17](#_Toc523192892)

[**Outlier analysis and treatment** 19](#_Toc523192893)

[**Feature Engineering** 19](#_Toc523192894)

[**Sampling** ……………………………………………………………………………………………………………………………………………………………….20](#_Toc523192895)

[**Model Building** 20](#_Toc523192896)

[**Complete R File** 22](#_Toc523192897)

[**Appendix c - Python Code** 28](#_Toc523192898)

[**Univariate Analysis** 28](#_Toc523192899)

[**Multivariate Analysis** 29](#_Toc523192900)

[**Outliers treatment** 30](#_Toc523192901)

[**Feature Engineering** 30](#_Toc523192902)

[**Sampling of data** 31](#_Toc523192903)

[**Model Building** 31](#_Toc523192904)

[**Complete Python File** 33](#_Toc523192905)

**Chapter 1**

# **Introduction**

* 1. **Problem Statement**

The aim of the project is to predict bike rental count daily based on the environmental and seasonal settings.

* 1. **Data**

The dataset contains daily count of rental bikes between years 2011 and 2012 with the corresponding weather and seasonal information. Our task is to build regression model which will forecast the bike rental count depending on various factors such as seasonal changes, weather changes, holidays, etc. Given below is a sample of the data set that we are using to predict the bike rental count.

Table 1.1: Bike Rental Sample Data (Columns 1:10)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **instant** | **dteday** | **season** | **yr** | **Mnth** | **holiday** | **weekday** | **workingday** | **weathersit** | **temp** |
| 1 | 01-01-2011 | 1 | 0 | 1 | 0 | 6 | 0 | 2 | 0.344167 |
| 2 | 02-01-2011 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 0.363478 |
| 3 | 03-01-2011 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0.196364 |
| 4 | 04-01-2011 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0.2 |
| 5 | 05-01-2011 | 1 | 0 | 1 | 0 | 3 | 1 | 1 | 0.226957 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **atemp** | **hum** | **windspeed** | **casual** | **registered** | **cnt** |
| 0.363625 | 0.805833 | 0.160446 | 331 | 654 | 985 |
| 0.353739 | 0.696087 | 0.248539 | 131 | 670 | 801 |
| 0.189405 | 0.437273 | 0.248309 | 120 | 1229 | 1349 |
| 0.212122 | 0.590435 | 0.160296 | 108 | 1454 | 1562 |

Table 1.2: Bike Rental Sample Data (Columns 11:16)

In the data set we have following 15 variables which will be used to forecast the bike rental count.

|  |  |
| --- | --- |
| **S.No.** | **Predictor** |
| 1 | instant |
| 2 | Date |
| 3 | season |
| 4 | year |
| 5 | month |
| 6 | holiday |
| 7 | Week day |
| 8 | Working day |
| 9 | weathersit |
| 10 | temperature |
| 11 | Feeling temperature |
| 12 | humidity |
| 13 | Wind speed |
| 14 | Casual users |
| 15 | Registered users |

**Chapter 2**

# **Methodology**

* 1. **Pre-Processing**

Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviours or trends, and is likely to contain many errors. Data pre-processing is a proven method of resolving such issues. Data pre-processing prepares raw data for further processing. Pre-processing involves various steps like exploratory data analysis i.e., visualizing the data through graphs and plots, data cleaning, data reduction, data transformation and so on.

### 

### **Variable Identification**

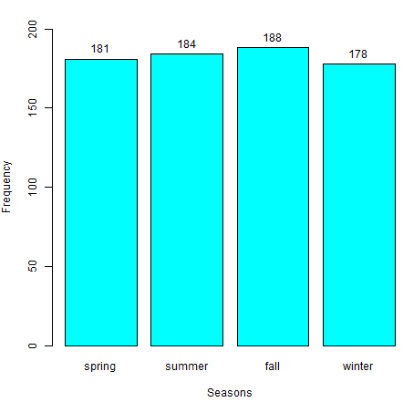
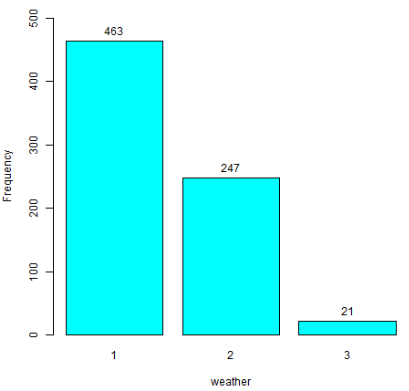
|  |  |  |
| --- | --- | --- |
| **S.No.** | **Categorical predictor Variables** | **Continuous Predictor variables** |
| 1. | season | year |
| 2. | holiday | Week day |
| 3. | Working day | temperature |
| 4. | weather | Feeling temperature |
| 5. |  | humidity |
| 6. |  | Wind speed |
| 7. |  | Casual users |
| 8. |  | Registered users |
| 9. |  | month |
| **Target Variable(s)** | | |
| 1. | Bike Count (type: continuous) | |

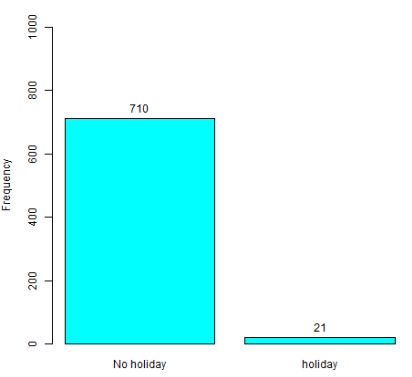
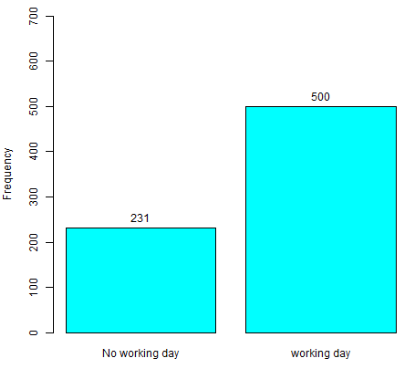
Here we will identify the predictor variables and target variable(s) and their data types.  
  
  
 Fig: 2.1.1 Data Type Classification Table

### **Univariate Analysis**

Here bar chart have been plotted for each factor variable respectively. Looking at the frequency distribution in fig.2.1.2, few observations can be drawn: Mostly Clear weather having few clouds is prevalent, all seasons are almost equally distributed, Working days and holidays show similar characteristics.

Figure 2.1.2 Frequency distributions of predictor variables

****

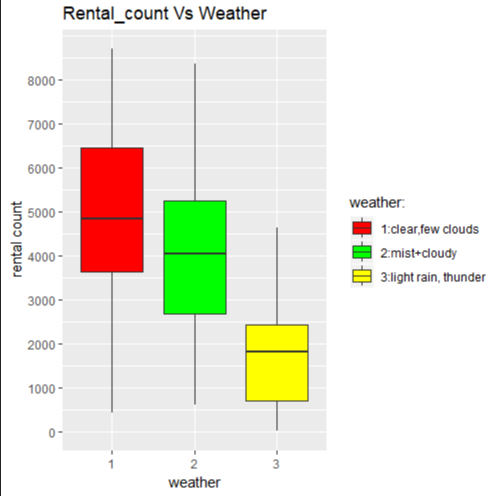
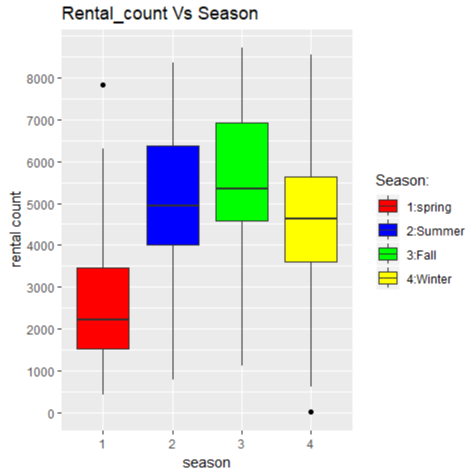


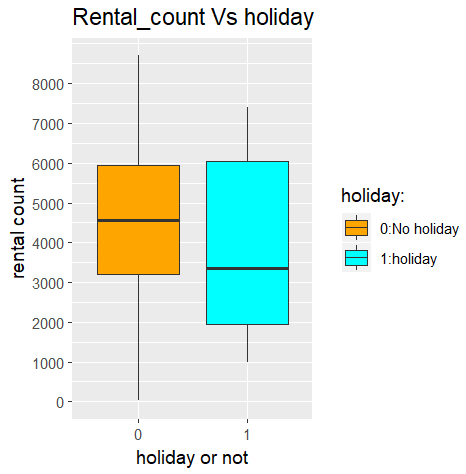
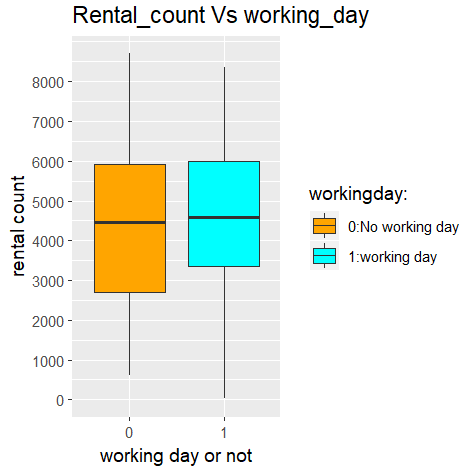
### **Bi-variate analysis**

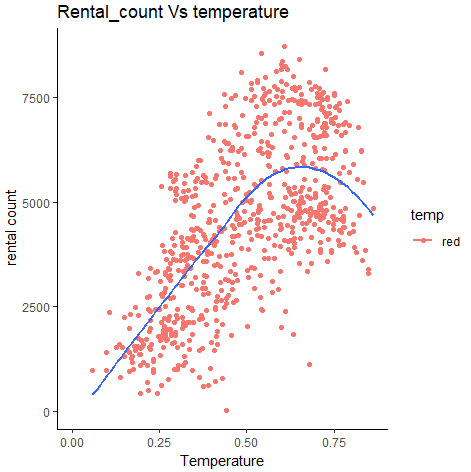
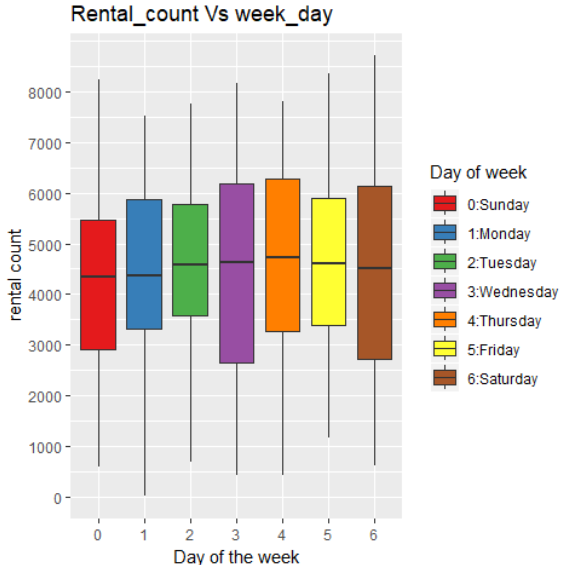
In figure 2.1.3, we look at the relationship of response variable “count” with other predictor variables and draw some notable inferences. In the plot showing relationship between weather and rental count, the average count is highest during clear weather and it decreases as the weather worsens. There is also seasonal trend associated with the numbers of bikes rented and count is comparatively higher during Fall and Summer seasons. Looking at temperature vs count plots, we can infer that more people will rent bikes when the temp is not cold and is warmer up to a certain point. Similarly in case of humidity, bike demand starts decreasing as the humidity reaches higher levels. In count vs holiday plot, the average rental count is almost similar, though the range of count in holidays is low due to small sample size for holidays. Similar is the case with count vs working days. Also, in the last plot showing relationship between wind speed and count, bike demand increases when it is slightly windy up to a certain point.

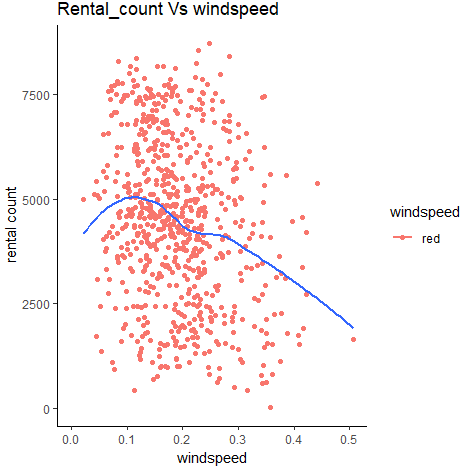
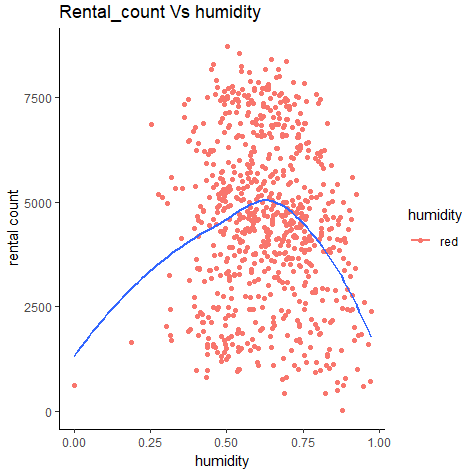
#### 

Figure 2.1.3 bivariate analysis





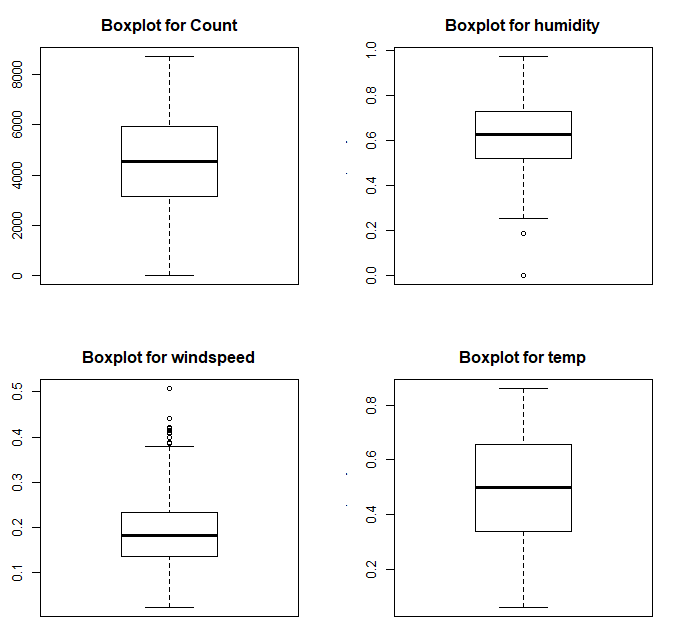
### **Missing Values Analysis**

Missing data in the training data set can reduce the power / fit of a model or can lead to a biased model because we have not analysed the behaviour and relationship with other variables correctly. Upon searching through the data, no missing value was found.

### **Outlier Analysis**

Looking at the boxplots in fig: 2.1.5, few outliers were found in variables windspeed and humidity, thereby removed from the data set.

Fig: 2.1.2.3



### **Feature Engineering**

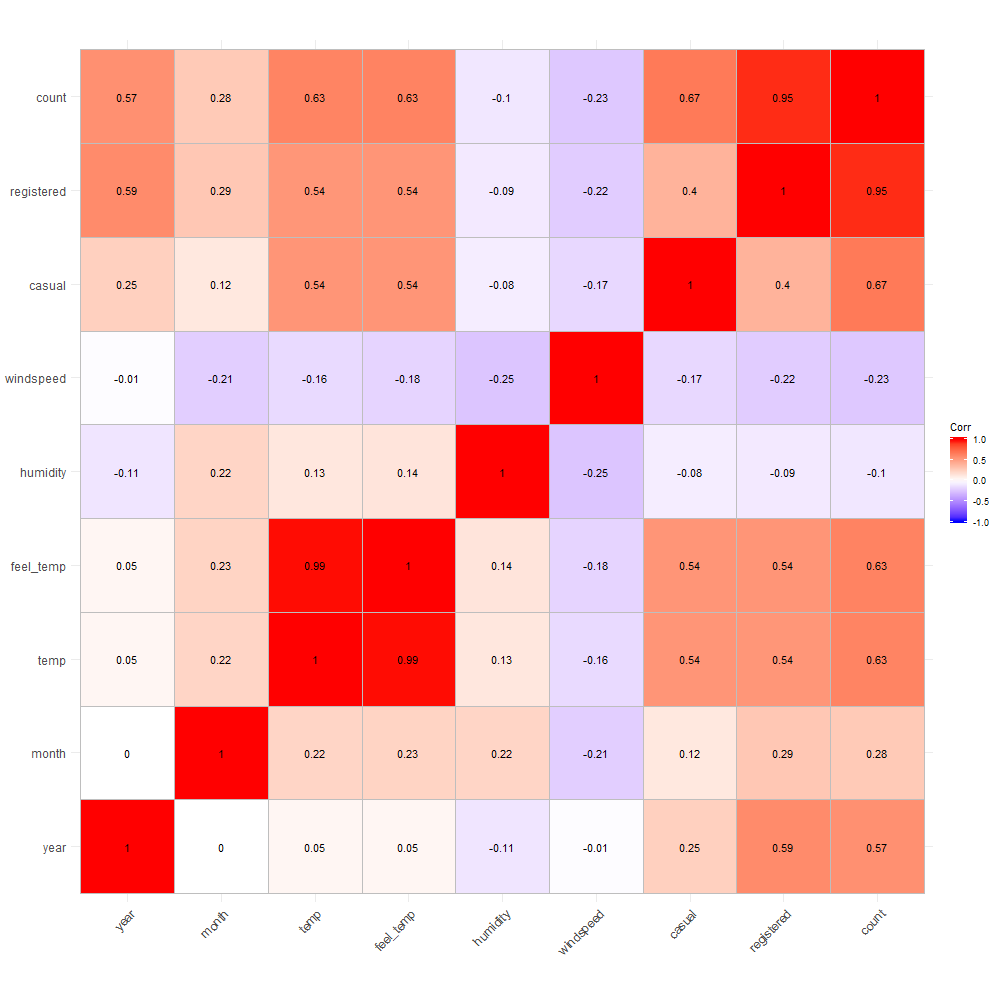
We will determine how strong the relationship is between the predictor variables because there might be a possibility that many variables in our analysis are not important at all to the regression problem. Then we’ll select a subset of relevant features (variables, predictors) for use in model construction.

#### **Correlation Analysis**

In figure 2.1.3, we have created a correlation matrix to find out the correlation between the numeric variables. Notable observations are:

1. Temperature and feel temperature are highly correlated.
2. Temperature is positively correlated with count.
3. Wind speed and Humidity have low negative correlation with count.

Figure 2.1.3 correlation plot



#### **ANOVA Analysis**

We’ll use Analysis of variance test, also known ANOVA, for comparing means of different groups in a factor variable.

ANOVA test hypotheses:

* Null hypothesis: the means of the different groups are the same
* Alternative hypothesis: At least one sample mean is not equal to the others.

Table 2.1.3- ANOVA analysis

|  |  |  |
| --- | --- | --- |
| **Factor variable vs Target variable** | **P- value** | **Result** |
| Seasons vs count | <0.05 | Alternative hypothesis true |
| Workingday vs count | 0.132 | Null hypothesis true |
| Holiday vs count | 0.0549 | Null hypothesis true |
| Weather vs count | <.05 | Alternative hypothesis true |
| weekday vs count | 0.584 | Null hypothesis true |

Notables observations based on ANOVA analysis:   
Null hypothesis (having probability value more than .05) is true for workingday, holiday and weekday variables, meaning that the means of different groups present in each factor variables do not contribute to the variance in the factor variables. Boxplots of above factor variables also describe the same trend.

#### **One Hot Encoding (dummy variables creation)**

For regression analysis, we’ll be using dummy variables derived from categorical variables that is season and weather. Also, the number of dummy variables necessary to represent a single attribute variable will be equal to the number of levels (categories) in that variable minus one.

Reason: Regression analysis treats all independent (X) variables in the analysis as numerical. Often, however, we might want to include an attribute or nominal scale variable such as ‘Product Brand’ or ‘Type of Defect’ in our analysis. For example we have three types of defects, numbered ‘1’, ‘2’ and ‘3’. In this case, ‘3 minus 1’ doesn’t mean anything. We can’t subtract defect 1 from defect 3. The numbers here are used to indicate or identify the levels of ‘Defect Type’ and do not have intrinsic meaning of their own. Dummy variables are created in this situation to ‘trick’ the regression algorithm into correctly analysing attribute variables.

#### **Conclusion**

* Feel\_temp variable to be removed as temp and feel\_temp are highly correlated.
* Registered and Casual variables to be removed, as we are predicting the total count and registered & casual sum up to the total count.
* Holiday, workingday, weekday to be removed based on ANOVA analysis.
* Based on one hot encoding process, season variable will be converted into 4 new variables namely- season\_summer, season\_winter, season\_fall, season\_spring, having values 0 or 1 in each column. Similarly weather will also be converted into new columns based on its category levels. For season variable,3 out of 4 derived columns will be used and for weather,2 out of 3 weather types will be used.

Our data set would contain below variables:   
"year"   
"month"   
"temp"   
"humidity"   
"windspeed"   
"season\_summer"

"season\_fall"

"season\_winter"

"weather\_weather1"

"weather\_weather3"

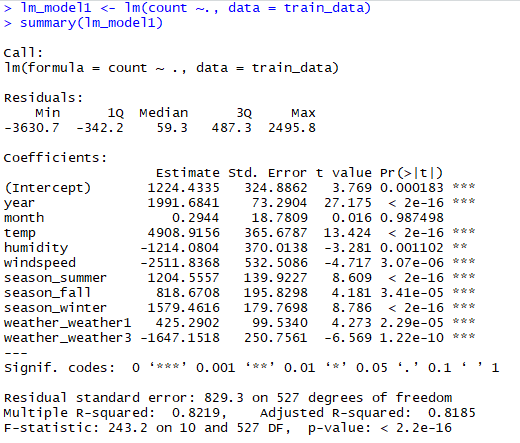
"count"

## **Modelling**

### **Model Selection**

Since we are predicting the bike count which is a continuous variable we’ll start by building a Multiple Linear Regression model. Before diving into building statistical models, data set is partitioned into two sets, training and testing. Training set will be used to train statistical models and estimate coefficients, while testing set will be used to validate the model. 80% of the complete data is partitioned into training set, sampled uniformly without replacement, and 20% is partitioned in to testing set.

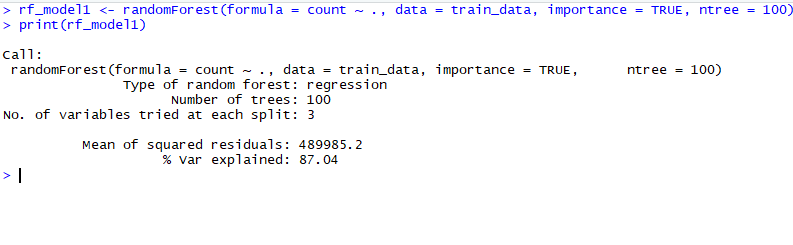
### **Multiple Linear Regression**



Looking at the Adjusted R-squared value, we can explain about 81.85% of the data using our multiple linear regression model. This seems sufficient, also looking at the p-value we can reject the null hypothesis that target variable does not depend on any of the predictor  
variables.

### **Random Forest Model**

Now we will try and use a different regression model to predict count target variable. We will use Random forest to predict the values of our target variable.



Looking at the % variance explained, we are able to explain about 87.04% of the data using random forest model.

**Chapter 3**

# **Conclusion**

## **Model Evaluation**

Now that we have two 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.

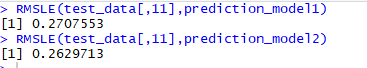
We will compare the models using predictive performance criteria.

### **Root Mean Squared Error Loss(RMSE)**

RMSE is one of the evaluation metrics used in regression problems. We will apply this measure to our models that we have generated in the previous section.



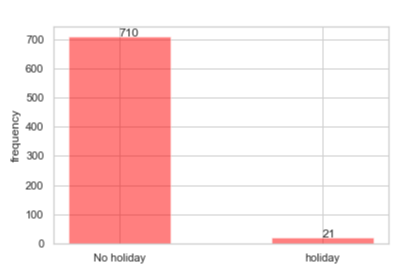
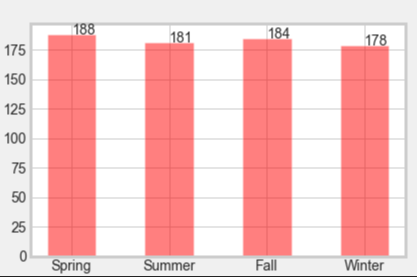
### **Root Mean Squared Logarithmic Error Loss(RMSLE)**

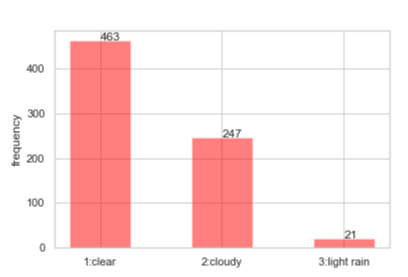
RMSLE can be obtained as follows  
  


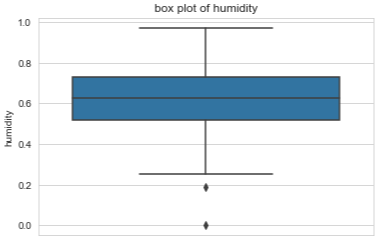
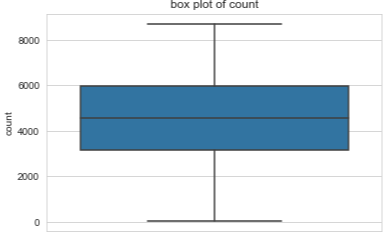
## **Model Selection**

Based on the above metric value we see that Random Forest model works better than Linear Regression model. Therefore we can select Random Forest for modelling data.

# **Appendix A - Extra Figures**







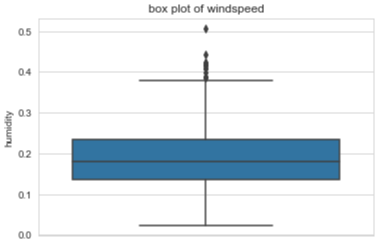
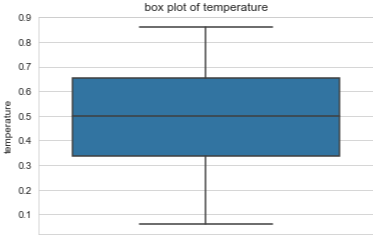
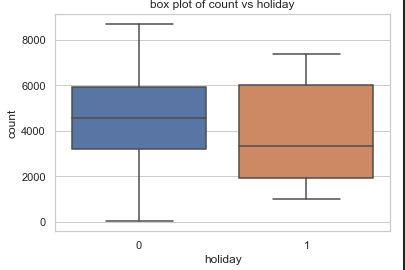
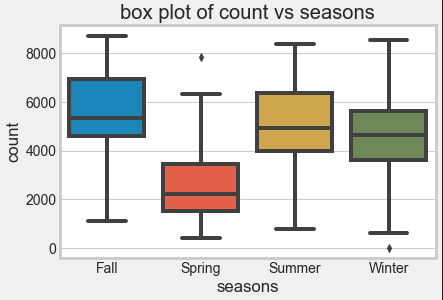
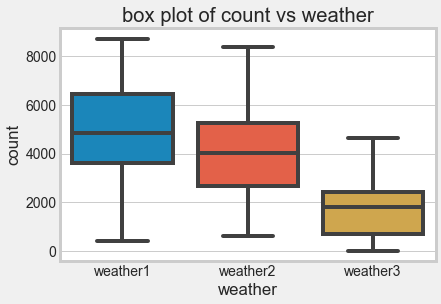
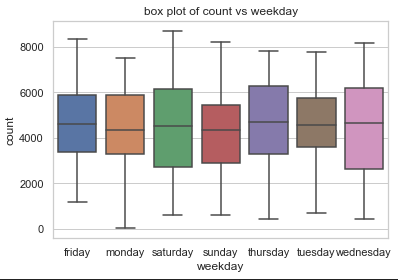
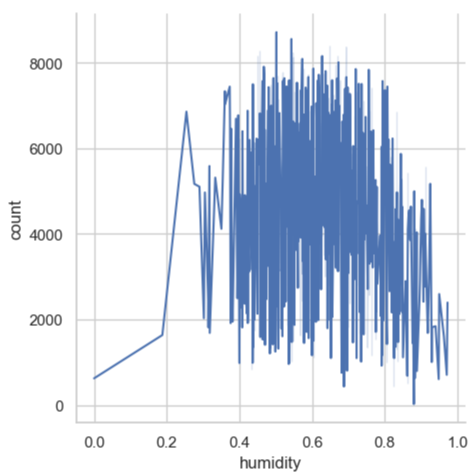
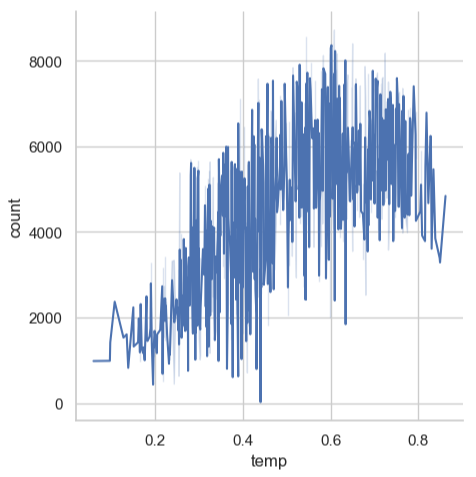
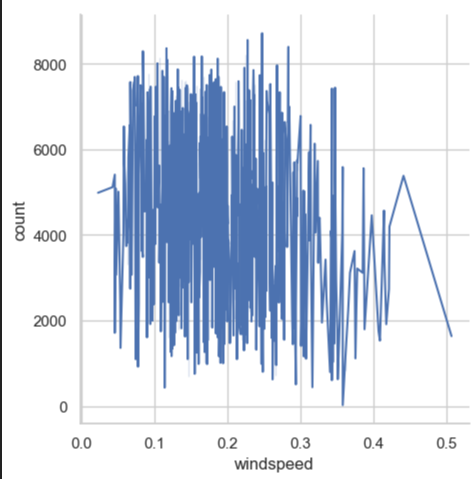
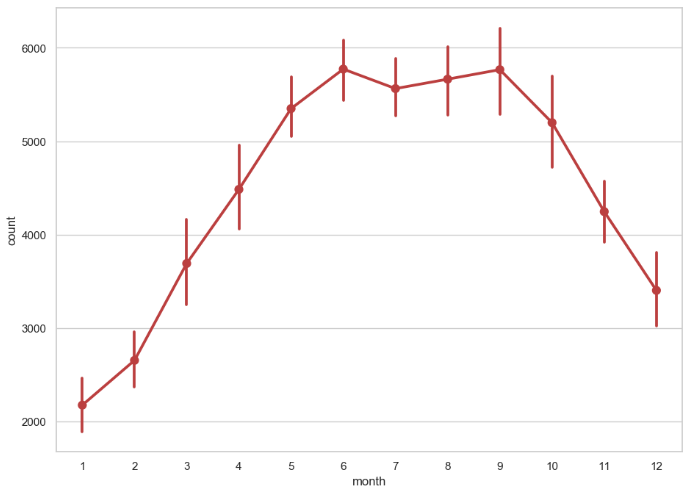


Figure: 3.1: frequency distribution of variables (python version)





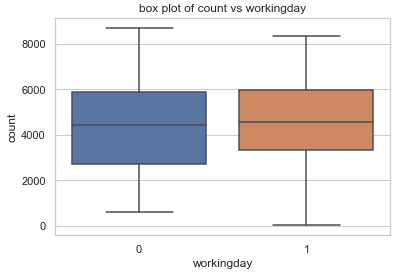
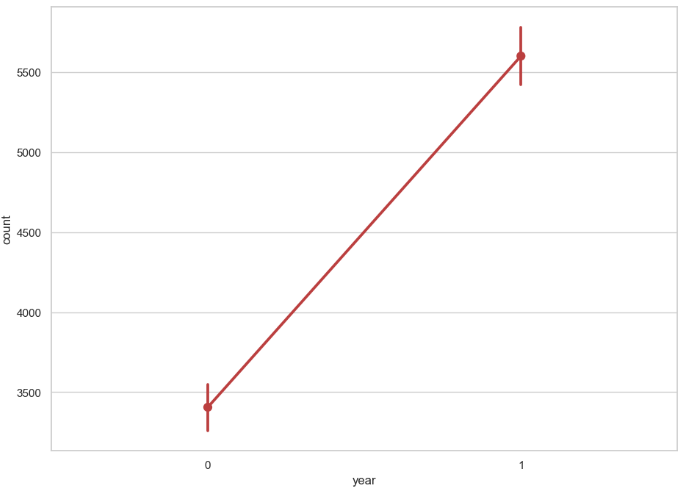
 

Figure: 3.2: Bi-variate Analysis (python version)

# **Appendix B - R Code**

### **univariate analysis**

#setting plot margins

par(mfrow=c(1,2),mar = rep(3,4))##Creates a multi-paneled plotting window & sets the margin size.

layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE))

#season barplot

png(file = "barplot\_seasons.png")

season\_plot <- barplot(table(data$season), names.arg= c("spring","summer","fall","winter"),col="cyan",ylim = c(0,200),

ylab = "Frequency",xlab = "Seasons")

text(season\_plot,table(data$season),labels = table(data$season),pos = 3)

dev.off()

#weather barplot

png(file = "barplot\_weather.png")

weather\_plot <- barplot(table(data$weather),col="cyan",ylim = c(0,500),

ylab = "Frequency",xlab = "weather")

text(weather\_plot,table(data$weather),labels = table(data$weather),pos = 3)

dev.off()

#holiday barplot

png(file = "barplot\_holiday.png")

holiday\_plot <- barplot(table(data$holiday),col="cyan",ylim = c(0,1000),

ylab = "Frequency",names.arg = c("No holiday","holiday"))

text(holiday\_plot,table(data$holiday),labels = table(data$holiday),pos = 3)

dev.off()

# workingday barplot

png(file = "barplot\_workingday.png")

workingday\_plot <- barplot(table(data$workingday),col="cyan",ylim = c(0,700),

ylab = "Frequency",names.arg = c("No working day","working day"))

text(workingday\_plot,table(data$workingday),labels = table(data$workingday),pos = 3)

dev.off()

#Boxplot for Count

png(file = "boxplot\_count.png")

boxplot(data$count, data = data,

xlab = "count",

ylab = "frequency", main = "Boxplot for Count")

dev.off()

#Boxplot for temp

png(file = "boxplot\_temp.png")

boxplot(data$temp, data = data,

xlab = "temperature",

ylab = "frequency", main = "Boxplot for temp")

dev.off()

#Boxplot for humidity

png(file = "boxplot\_humidity.png")

boxplot(data$humidity, data = data,

xlab = "humidity",

ylab = "frequency", main = "Boxplot for humidity")

dev.off()

#Boxplot for windspeed

png(file = "boxplot\_windspeed.png")

boxplot(data$windspeed, data = data,

xlab = "windspeed",

ylab = "frequency", main = "Boxplot for windspeed")

dev.off()

### **Multivariate Analysis**

#boxplot of rental count & season

png(file = "rental count vs season.png",res = 100)

ggplot(data, aes(x = season, y = count,fill = season)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("season") +

ylab("rental count") +

ggtitle("Rental\_count Vs Season") +

scale\_fill\_manual(values=c("red", "blue", "green","yellow"),

name="Season:",

labels=c("spring", "Summer", "Fall", "Winter"))

dev.off()

#boxplot of rental count & weather

png(file = "rental count vs weather.png",width = 500,height = 500,res = 100)

ggplot(data, aes(x = weather, y = count,fill = weather)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("weather") +

ylab("rental count") +

ggtitle("Rental\_count Vs Weather") +

scale\_fill\_manual(values=c("red", "green", "yellow","blue"),

name="weather:",

labels=c("1:clear,few clouds", "2:mist+cloudy",

"3:light rain, thunder", "4:heavy rain,thunderstorm"))

dev.off()

#boxplot of rental count & holiday

png(file = "rental count vs holiday.png",res = 125)

ggplot(data, aes(x = holiday, y = count,fill = holiday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("holiday or not") +

ylab("rental count") +

ggtitle("Rental\_count Vs holiday") +

scale\_fill\_manual(values=c("orange", "cyan"),

name="holiday:",

labels=c("0:No holiday","1:holiday"))

dev.off()

#boxplot of rental count & working day

png(file = "rental count vs workingday.png",res = 100)

ggplot(data, aes(x = workingday, y = count,fill = workingday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("working day or not") +

ylab("rental count") +

ggtitle("Rental\_count Vs working\_day") +

scale\_fill\_manual(values=c("orange", "cyan"),

name="workingday:",

labels=c("0:No working day","1:working day"))

dev.off()

#boxplot of rental count & weekday

png(file = "rental count vs weekday.png",res = 100)

ggplot(data, aes(x = weekday, y = count,fill = weekday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("Day of the week") +

ylab("rental count") +

ggtitle("Rental\_count Vs week\_day") +

scale\_fill\_brewer(palette="Set1",name="Day of week",

labels=c("0:Sunday","1:Monday","2:Tuesday","3:Wednesday",

"4:Thursday","5:Friday","6:Saturday"))

dev.off()

#line plot of rental count & month

png(file = "rental count vs month.png",res = 100)

ggplot(data, aes(x = month, y = count, color = month)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$month)), ylim=c(0,max(data$count))) +

xlab("month") +

ylab("rental count") +

ggtitle("Rental\_count Vs month") +

theme\_classic()

dev.off()

# line plot of rental v.s. year

png(file = "rental count vs year.png",res = 100)

ggplot(data, aes(x = year, y = count, color = year)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$year)), ylim=c(0,max(data$count))) +

xlab("year") +

ylab("rental count") +

ggtitle("Rental\_count Vs year") +

theme\_classic()

dev.off()

# line plot of rental v.s. temperature

png(file = "rental count vs temp.png",res = 100)

ggplot(data, aes(x = temp, y = count, color = temp)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$temp)), ylim=c(0,max(data$count))) +

xlab("Temperature") +

ylab("rental count") +

ggtitle("Rental\_count Vs temperature") +

theme\_classic()

dev.off()

# line plot of rental v.s. humidity

png(file = "rental count vs humidity.png",res = 100)

ggplot(data, aes(x = humidity, y = count,color = humidity)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$humidity)), ylim=c(0,max(data$count))) +

xlab("humidity") +

ylab("rental count") +

ggtitle("Rental\_count Vs humidity") +

theme\_classic()

dev.off()

# line plot of rental v.s. wind speed

png(file = "rental count vs windspeed.png",res = 100)

ggplot(data, aes(x = windspeed, y = count,color = windspeed)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$windspeed)), ylim=c(0,max(data$count))) +

xlab("windspeed") +

ylab("rental count") +

ggtitle("Rental\_count Vs windspeed") +

theme\_classic()

dev.off()

### **Outlier analysis and treatment**

#save numeric names

cnames <- c("count", "temp", "humidity", "windspeed")

for (i in cnames) {

q25 <- quantile(data[,i],probs = 0.25)

q75 <- quantile(data[,i],probs = 0.75)

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

data <- data[!data[,i] < min,]

data <- data[!data[,i] > max,]

}

### **Feature Engineering**

#correlation Plot

numeric\_index <- sapply(data,is.numeric) #selecting only numeric

numeric\_data <- data[,numeric\_index]

#plot1

png(file = "correlation\_matrix\_1.png",width = 1000,height = 1000)

corrgram(numeric\_data, order = F,

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

dev.off()

#Plot2

png(file = "correlation\_matrix\_2.png",width = 1000,height = 1000)

ggcorrplot(cor(numeric\_data),method = "square",type = "full", lab = TRUE)

dev.off()

#Anova analysis

#count wrt seasons

anova\_one\_way1 <- aov(count~season,data[order(data$season),])

summary(anova\_one\_way1)

#count wrt workingday

anova\_one\_way2 <- aov(count~workingday,data[order(data$workingday),])

summary(anova\_one\_way2) #p > .05

#count wrt holiday

anova\_one\_way3 <- aov(count~holiday,data[order(data$holiday),])

summary(anova\_one\_way3) #p > .05

#count wrt weather

anova\_one\_way4 <- aov(count~weather,data[order(data$weather),])

summary(anova\_one\_way4)

#count wrt weekday

anova\_one\_way5 <- aov(count~weekday,data[order(data$weekday),])

summary(anova\_one\_way5) #p > .05

#------------------------------------------------------------------------------

#conclusion based on anova,correlation plot

#remove weekday,holiday,workingday,atemp,casual,registered

drop <- c("holiday","weekday","feel\_temp","workingday","casual","registered")

data <- data[ , !names(data) %in% drop]

rm(drop)

#dummy variable(one hot) encoding---------------------

#install.packages("fastDummies")

library("fastDummies")

data <- dummy\_cols(data,remove\_first\_dummy = TRUE)

data <- data[ , !names(data) %in% c("season","weather")]

data <- data[,c(1:5,7:11,6)]

### **Sampling**

set.seed(200)

sample\_index <- sample(nrow(data), nrow(data)\*0.20,replace = FALSE)

test\_data <- data[sample\_index,]

train\_data <- data[-sample\_index,]

### **Model Building**

#multiple linear regression

# Build the model

lm\_model1 <- lm(count ~., data = train\_data)

summary(lm\_model1)

#predict

prediction\_model1 <- predict(lm\_model1,test\_data[1:10])

#modelevaluation

#1.Root Mean Squared Error Loss

RMSE = function(yhat,y\_tru ){

sqrt(mean((yhat - y\_tru)^2))

}

RMSE(prediction\_model1,test\_data[,11])

#2.Root Mean Squared Logarithmic Error Loss

RMSLE <- function(y\_true, y\_pred) {

sqrt(mean((log1p(y\_true)-log1p(y\_pred))^2))

}

RMSLE(test\_data[,11],prediction\_model1)

#random forest algorithm-------------------------------------

library(randomForest)

rf\_model1 <- randomForest(formula = count ~ ., data = train\_data, importance = TRUE, ntree = 100)

print(rf\_model1)

#predict

prediction\_model2 <- predict(rf\_model1,test\_data[1:10])

#model evaluation

RMSE(prediction\_model2,test\_data[,11])

RMSLE(test\_data[,11],prediction\_model2)

submit = data.frame(test\_data, predicted\_count = round(prediction\_model2))

write.csv(submit,file = "submit\_r.csv",row.names = FALSE)

#---------------------------------------------------------------------------------------------------

### 

### **Complete R File**

#remove all objects stored

rm(list = ls())

#set working directory

setwd("D:/edwisor\_project/R\_files")

#Load libraries

x = c("ggplot2","ggcorrplot", "corrgram", "MASS", "rpart", "gbm")

#install.packages(x)

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

rm(x)

#load data

data\_original <- read.csv("day.csv",stringsAsFactors = FALSE)

#use copy of coriginal data

data <- data\_original

#rename variables,if required

names(data)[names(data)=="dteday"] <- "date"

names(data)[names(data)=="yr"] <- "year"

names(data)[names(data)=="mnth"] <- "month"

names(data)[names(data)=="hum"] <- "humidity"

names(data)[names(data)=="cnt"] <- "count"

names(data)[names(data)=="atemp"] <- "feel\_temp"

names(data)[names(data)=="weathersit"] <- "weather"

#convert to required data types

data$season=factor(data$season,levels = c(1,2,3,4), labels = c("spring","summer","fall","winter"))

data$weather=factor(data$weather,levels = c(1,2,3,4), labels = c("weather1","weather2","weather3","weather4"))

data$weekday=factor(data$weekday,levels = c(0,1,2,3,4,5,6), labels = c("sunday","monday","tuesday","wednesday","thursday","friday","saturday"))

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

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

#------------------------------------------DATA PRE-PROCESSING----------------------------------------------

#removing column instant(index numbers) as it has no corelation to any other variable.

#day,season,workingday,holiday are already derived in the table therfore date attribute not required.

data <- data[,c(-1,-2)]

#check for missing values ,if any

#can be tested using is.null() also.

if(sum(is.na(data))== 0) {

print("No missing values found")

} else {

print("missing value(s) existing")

}

#check for duplicates

if(sum(duplicated(data))== 0) {

print("No duplicates found")

} else {

print("duplicate data existing")

}

#####1.univariate analysis

#setting plot margins

par(mfrow=c(1,2),mar = rep(3,4))##Creates a multi-paneled plotting window & sets the margin size.

layout(matrix(c(1,2,3,4), 2, 2, byrow = TRUE))

#season barplot

png(file = "barplot\_seasons.png")

season\_plot <- barplot(table(data$season), names.arg= c("spring","summer","fall","winter"),col="cyan",ylim = c(0,200),

ylab = "Frequency",xlab = "Seasons")

text(season\_plot,table(data$season),labels = table(data$season),pos = 3)

dev.off()

#weather barplot

png(file = "barplot\_weather.png")

weather\_plot <- barplot(table(data$weather),col="cyan",ylim = c(0,500),

ylab = "Frequency",xlab = "weather")

text(weather\_plot,table(data$weather),labels = table(data$weather),pos = 3)

dev.off()

#holiday barplot

png(file = "barplot\_holiday.png")

holiday\_plot <- barplot(table(data$holiday),col="cyan",ylim = c(0,1000),

ylab = "Frequency",names.arg = c("No holiday","holiday"))

text(holiday\_plot,table(data$holiday),labels = table(data$holiday),pos = 3)

dev.off()

# workingday barplot

png(file = "barplot\_workingday.png")

workingday\_plot <- barplot(table(data$workingday),col="cyan",ylim = c(0,700),

ylab = "Frequency",names.arg = c("No working day","working day"))

text(workingday\_plot,table(data$workingday),labels = table(data$workingday),pos = 3)

dev.off()

#Boxplot for Count

png(file = "boxplot\_count.png")

boxplot(data$count, data = data,

xlab = "count",

ylab = "frequency", main = "Boxplot for Count")

dev.off()

#Boxplot for temp

png(file = "boxplot\_temp.png")

boxplot(data$temp, data = data,

xlab = "temperature",

ylab = "frequency", main = "Boxplot for temp")

dev.off()

#Boxplot for humidity

png(file = "boxplot\_humidity.png")

boxplot(data$humidity, data = data,

xlab = "humidity",

ylab = "frequency", main = "Boxplot for humidity")

dev.off()

#Boxplot for windspeed

png(file = "boxplot\_windspeed.png")

boxplot(data$windspeed, data = data,

xlab = "windspeed",

ylab = "frequency", main = "Boxplot for windspeed")

dev.off()

####2.Multivariate Analysis

#boxplot of rental count & season

png(file = "rental count vs season.png",res = 100)

ggplot(data, aes(x = season, y = count,fill = season)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("season") +

ylab("rental count") +

ggtitle("Rental\_count Vs Season") +

scale\_fill\_manual(values=c("red", "blue", "green","yellow"),

name="Season:",

labels=c("spring", "Summer", "Fall", "Winter"))

dev.off()

#boxplot of rental count & weather

png(file = "rental count vs weather.png",width = 500,height = 500,res = 100)

ggplot(data, aes(x = weather, y = count,fill = weather)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("weather") +

ylab("rental count") +

ggtitle("Rental\_count Vs Weather") +

scale\_fill\_manual(values=c("red", "green", "yellow","blue"),

name="weather:",

labels=c("1:clear,few clouds", "2:mist+cloudy",

"3:light rain, thunder", "4:heavy rain,thunderstorm"))

dev.off()

#boxplot of rental count & holiday

png(file = "rental count vs holiday.png",res = 125)

ggplot(data, aes(x = holiday, y = count,fill = holiday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("holiday or not") +

ylab("rental count") +

ggtitle("Rental\_count Vs holiday") +

scale\_fill\_manual(values=c("orange", "cyan"),

name="holiday:",

labels=c("0:No holiday","1:holiday"))

dev.off()

#boxplot of rental count & working day

png(file = "rental count vs workingday.png",res = 100)

ggplot(data, aes(x = workingday, y = count,fill = workingday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("working day or not") +

ylab("rental count") +

ggtitle("Rental\_count Vs working\_day") +

scale\_fill\_manual(values=c("orange", "cyan"),

name="workingday:",

labels=c("0:No working day","1:working day"))

dev.off()

#boxplot of rental count & weekday

png(file = "rental count vs weekday.png",res = 100)

ggplot(data, aes(x = weekday, y = count,fill = weekday)) +

geom\_boxplot(outlier.colour = "black",na.rm = TRUE) +

coord\_cartesian(ylim=c(0,max(data$count))) +

scale\_y\_continuous(breaks=seq(0, max(data$count),1000)) +

xlab("Day of the week") +

ylab("rental count") +

ggtitle("Rental\_count Vs week\_day") +

scale\_fill\_brewer(palette="Set1",name="Day of week",

labels=c("0:Sunday","1:Monday","2:Tuesday","3:Wednesday",

"4:Thursday","5:Friday","6:Saturday"))

dev.off()

#line plot of rental count & month

png(file = "rental count vs month.png",res = 100)

ggplot(data, aes(x = month, y = count, color = month)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$month)), ylim=c(0,max(data$count))) +

xlab("month") +

ylab("rental count") +

ggtitle("Rental\_count Vs month") +

theme\_classic()

dev.off()

# line plot of rental v.s. year

png(file = "rental count vs year.png",res = 100)

ggplot(data, aes(x = year, y = count, color = year)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$year)), ylim=c(0,max(data$count))) +

xlab("year") +

ylab("rental count") +

ggtitle("Rental\_count Vs year") +

theme\_classic()

dev.off()

# line plot of rental v.s. temperature

png(file = "rental count vs temp.png",res = 100)

ggplot(data, aes(x = temp, y = count, color = temp)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$temp)), ylim=c(0,max(data$count))) +

xlab("Temperature") +

ylab("rental count") +

ggtitle("Rental\_count Vs temperature") +

theme\_classic()

dev.off()

# line plot of rental v.s. humidity

png(file = "rental count vs humidity.png",res = 100)

ggplot(data, aes(x = humidity, y = count,color = humidity)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$humidity)), ylim=c(0,max(data$count))) +

xlab("humidity") +

ylab("rental count") +

ggtitle("Rental\_count Vs humidity") +

theme\_classic()

dev.off()

# line plot of rental v.s. wind speed

png(file = "rental count vs windspeed.png",res = 100)

ggplot(data, aes(x = windspeed, y = count,color = windspeed)) +

geom\_point(aes(color="red")) +

geom\_smooth(fill = NA) +

coord\_cartesian(xlim = c(0,max(data$windspeed)), ylim=c(0,max(data$count))) +

xlab("windspeed") +

ylab("rental count") +

ggtitle("Rental\_count Vs windspeed") +

theme\_classic()

dev.off()

#####3. outlier analysis and treatment

#save numeric names

cnames <- c("count", "temp", "humidity", "windspeed")

for (i in cnames) {

q25 <- quantile(data[,i],probs = 0.25)

q75 <- quantile(data[,i],probs = 0.75)

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

data <- data[!data[,i] < min,]

data <- data[!data[,i] > max,]

}

#####4.Feature Engineering

#correlation Plot

numeric\_index <- sapply(data,is.numeric) #selecting only numeric

numeric\_data <- data[,numeric\_index]

#plot1

png(file = "correlation\_matrix\_1.png",width = 1000,height = 1000)

corrgram(numeric\_data, order = F,

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

dev.off()

#Plot2

png(file = "correlation\_matrix\_2.png",width = 1000,height = 1000)

ggcorrplot(cor(numeric\_data),method = "square",type = "full", lab = TRUE)

dev.off()

#Anova analysis

#count wrt seasons

anova\_one\_way1 <- aov(count~season,data[order(data$season),])

summary(anova\_one\_way1)

#count wrt workingday

anova\_one\_way2 <- aov(count~workingday,data[order(data$workingday),])

summary(anova\_one\_way2) #p > .05

#count wrt holiday

anova\_one\_way3 <- aov(count~holiday,data[order(data$holiday),])

summary(anova\_one\_way3) #p > .05

#count wrt weather

anova\_one\_way4 <- aov(count~weather,data[order(data$weather),])

summary(anova\_one\_way4)

#count wrt weekday

anova\_one\_way5 <- aov(count~weekday,data[order(data$weekday),])

summary(anova\_one\_way5) #p > .05

#------------------------------------------------------------------------------

#conclusion based on anova,correlation plot

#remove weekday,holiday,workingday,atemp,casual,registered

drop <- c("holiday","weekday","feel\_temp","workingday","casual","registered")

data <- data[ , !names(data) %in% drop]

rm(drop)

#dummy variable(one hot) encoding---------------------

#install.packages("fastDummies")

library("fastDummies")

data <- dummy\_cols(data,remove\_first\_dummy = TRUE)

data <- data[ , !names(data) %in% c("season","weather")]

data <- data[,c(1:5,7:11,6)]

#--------------------------------------------------------------------------------------

#sampling

set.seed(200)

sample\_index <- sample(nrow(data), nrow(data)\*0.20,replace = FALSE)

test\_data <- data[sample\_index,]

train\_data <- data[-sample\_index,]

#--------------------------------------MODEL BUILDING-------------------------------------------------------

#multiple linear regression

# Build the model

lm\_model1 <- lm(count ~., data = train\_data)

summary(lm\_model1)

#predict

prediction\_model1 <- predict(lm\_model1,test\_data[1:10])

#modelevaluation

#1.Root Mean Squared Error Loss

RMSE = function(yhat,y\_tru ){

sqrt(mean((yhat - y\_tru)^2))

}

RMSE(prediction\_model1,test\_data[,11])

#2.Root Mean Squared Logarithmic Error Loss

RMSLE <- function(y\_true, y\_pred) {

sqrt(mean((log1p(y\_true)-log1p(y\_pred))^2))

}

RMSLE(test\_data[,11],prediction\_model1)

#random forest algorithm-------------------------------------

library(randomForest)

rf\_model1 <- randomForest(formula = count ~ ., data = train\_data, importance = TRUE, ntree = 100)

print(rf\_model1)

#predict

prediction\_model2 <- predict(rf\_model1,test\_data[1:10])

#model evaluation

RMSE(prediction\_model2,test\_data[,11])

RMSLE(test\_data[,11],prediction\_model2)

submit = data.frame(test\_data, predicted\_count = round(prediction\_model2))

write.csv(submit,file = "submit\_r.csv",row.names = FALSE)

#---------------------------------------------------------------------------------------------------

# **Appendix c - Python Code**

### **Univariate Analysis**

#season barplot

season\_table = pd.DataFrame({"category":df.season.value\_counts().index, "frequency":df.season.value\_counts().values}).sort\_values('category').reset\_index(drop = True)

labels = ("Spring","Summer","Fall","Winter")

plt.bar(x = season\_table.category, height = season\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(season\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(season\_table.category, season\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('seasons.png')

#plt.close()

#weather barplot

weather\_table = pd.DataFrame({"category":df.weather.value\_counts().index, "frequency":df.weather.value\_counts().values})

labels = ("1:clear","2:cloudy","3:light rain","4:heavy rain")

plt.bar(x = weather\_table.category, height = weather\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(weather\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(weather\_table.category, weather\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('weather.png')

#plt.close()

#holiday barplot

holiday\_table = pd.DataFrame({"category":df.holiday.value\_counts().index, "frequency":df.holiday.value\_counts().values})

labels = ("No holiday","holiday")

plt.bar(x = holiday\_table.category, height = holiday\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(holiday\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(holiday\_table.category, holiday\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('holiday.png')

#plt.close()

#workingday barplot

workingday\_table = pd.DataFrame({"category":df.workingday.value\_counts().index, "frequency":df.workingday.value\_counts().values})

labels = ("working day ","no working day")

plt.bar(x = workingday\_table.category, height = workingday\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(workingday\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(workingday\_table.category, workingday\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('workingday.png',)

#plt.close()

#count boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="count",

data=df).set(ylabel = 'count',title="box plot of count")

plt.savefig('boxplot\_count.png',bbox\_inches='tight')

#temperature boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="temp",

data=df).set(ylabel = 'temperature',title="box plot of temperature")

plt.savefig('boxplot\_temp.png',bbox\_inches='tight')

#humidity boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="humidity",

data=df).set(ylabel = 'humidity',title="box plot of humidity")

plt.savefig('boxplot\_humidity.png',bbox\_inches='tight')

#windspeed boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="windspeed",

data=df).set(ylabel = 'humidity',title="box plot of windspeed")

plt.savefig('boxplot\_windspeed.png',bbox\_inches='tight')

#count distribution plot

sns.distplot(df['count'], hist=True, kde=False,

bins = np.arange(0,10000, 500),color = 'darkred',

hist\_kws={'edgecolor':'black'},

kde\_kws={'linewidth': 2})

### **Multivariate Analysis**

#boxplot of rental count & season

sns.set\_style('whitegrid')

sns.boxplot(x="season", y="count",

data=df).set(xlabel = 'seasons', ylabel = 'count',title="box plot of count vs seasons")

plt.savefig('count\_vs\_season.png',bbox\_inches='tight')

#boxplot of rental count & weather

sns.boxplot(x = "weather", y="count",

data=df).set(xlabel = 'weather', ylabel = 'count',title="box plot of count vs weather")

plt.savefig('count\_vs\_weather.png',bbox\_inches='tight')

#boxplot of rental count & holiday

sns.boxplot(x="holiday", y="count",

data=df).set(xlabel = 'holiday', ylabel = 'count',title="box plot of count vs holiday")

plt.savefig('count\_vs\_holiday.png',bbox\_inches='tight')

#boxplot of rental count & working day

sns.boxplot(x="workingday", y="count",

data=df).set(xlabel = 'workingday', ylabel = 'count',title="box plot of count vs workingday")

plt.savefig('count\_vs\_workingday.png',bbox\_inches='tight')

#boxplot of rental count & weekday

sns.boxplot(x="weekday", y="count",

data=df).set(xlabel = 'weekday', ylabel = 'count',title="box plot of count vs weekday")

plt.savefig('count\_vs\_weekday.png',bbox\_inches='tight')

#Lineplot of rental count & year

fig, ax = plt.subplots()

fig.set\_size\_inches(11, 8)

sns.set(style="whitegrid")

sns.pointplot(x="year", y="count", data=df,color="#bb3f3f")

plt.savefig("count\_vs\_year.png",bbox\_inches='tight',dpi=100)

#Lineplot of rental count & month

fig, ax = plt.subplots()

fig.set\_size\_inches(11, 8)

sns.set(style="whitegrid")

sns.pointplot(x="month", y="count", data=df,color="#bb3f3f")

plt.savefig("count\_vs\_month.png",bbox\_inches='tight',dpi=100)

# scatter plot of rental v.s. temperature

sns.set(style="whitegrid")

sns.relplot(x="temp", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_temp.png",bbox\_inches='tight',dpi=100)

# line plot of rental v.s. humidity

sns.set(style="whitegrid")

sns.relplot(x="humidity", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_humidity.png",bbox\_inches='tight',dpi=100)

# line plot of rental v.s. wind speed

sns.set(style="whitegrid")

sns.relplot(x="windspeed", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_windspeed.png",bbox\_inches='tight',dpi=100)

### **Outliers treatment**

#save numeric names

cnames = ("count", "temp", "humidity", "windspeed")

for i in cnames:

print(i)

q75, q25 = np.percentile(df.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

df = df.drop(df[df.loc[:,i] < min].index)

df = df.drop(df[df.loc[:,i] > max].index)

### **Feature Engineering**

numeric\_data = df.select\_dtypes(include=['float64','int64'])

corr\_data = numeric\_data.corr()

fig,ax= plt.subplots()

fig.set\_size\_inches(20,10)

sns.heatmap(corr\_data,vmin = -1 ,vmax = 1, square=True,annot=True)

ANOVA analysis for categorical variables

stats.f\_oneway(df['count'][df['season'] == "Spring"],

df['count'][df['season'] == 'Summer'],

df['count'][df['season'] == 'Fall'],

df['count'][df['season'] == 'Winter'])

stats.f\_oneway(df['count'][df['weather'] == 'weather1'],

df['count'][df['weather'] == 'weather2'],

df['count'][df['weather'] == 'weather3'])

stats.f\_oneway(df['count'][df['workingday'] == 0],

df['count'][df['workingday'] == 1])

stats.f\_oneway(df['count'][df['holiday'] == 0],

df['count'][df['holiday'] == 1])

stats.f\_oneway(df['count'][df['weekday'] == 'sunday'],

df['count'][df['weekday'] == 'monday'],

df['count'][df['weekday'] == 'tuesday'],

df['count'][df['weekday'] == 'wednesday'],

df['count'][df['weekday'] == 'thursday'],

df['count'][df['weekday'] == 'friday'],

df['count'][df['weekday'] == 'saturday'])

#conclusion based on anova,correlation plot

#remove weekday,holiday,workingday,feel\_temp,casual,registered

df = df.drop(columns=["feel\_temp","casual","registered","weekday","holiday","workingday"], axis = 1);

# One-hot encode the data using pandas get\_dummies

df = pd.get\_dummies(df,drop\_first = True)

#reorder columns

cols = df.columns.tolist()

cols = cols[:5] + cols[6:] + cols[5:6]

df = df[cols]

### **Sampling of data**

# Using Skicit-learn to split data into training and testing sets

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

# Split the data into training and testing sets

train, test = train\_test\_split(df,test\_size = 0.20, random\_state = 42)

# Saving feature names for later use

feature\_list = list(df.drop('count', axis = 1).columns)

### **Model Building**

Linear Regression

import statsmodels.api as sm

model\_lr = sm.OLS(train.iloc[:,10], train.iloc[:,0:10]).fit()

model\_lr.summary()

predictions\_lr = model\_lr.predict(test.iloc[:,0:10])

#define error metrics

##1.Root Mean Squared Error Loss

def rmse(predictions, targets):

return np.sqrt(((predictions - targets) \*\* 2).mean())

#2.Root Mean Squared Logarithmic Error Loss

def rmsle(y\_pred, y\_test) :

assert len(y\_test) == len(y\_pred)

return np.sqrt(np.mean((np.log1p(y\_pred) - np.log1p(y\_test))\*\*2))

print("RMSE is:",rmse(predictions\_lr,test["count"]))

print("RMSLE is:",rmsle(predictions\_lr,test["count"]))

Random Forest Model

#separate in to Features and Targets and Convert Data to Arrays

# Labels are the values we want to predict.

train\_labels = np.array(train['count'])

test\_labels = np.array(test['count'])

# Remove the labels from the features(data)

train\_features= train.drop('count', axis = 1)

test\_features= test.drop('count', axis = 1)

# Convert to numpy array

train\_features = np.array(train\_features)

test\_features = np.array(test\_features)

print('Training Features Shape:', train\_features.shape)

print('Training Labels Shape:', train\_labels.shape)

print('Testing Features Shape:', test\_features.shape)

print('Testing Labels Shape:', test\_labels.shape)

from sklearn.ensemble import RandomForestRegressor

# Instantiate model with 1000 decision trees

rf = RandomForestRegressor(n\_estimators = 100, random\_state = 42)

# Train the model on training data

rf.fit(train\_features, train\_labels);

# Use the forest's predict method on the test data

predictions\_rf = rf.predict(test\_features)

#error metrics

print("RMSE is:",rmse(predictions\_rf,test\_labels))

print("RMSLE is:",rmsle(predictions\_rf,test\_labels))

##write to csv file

data = pd.DataFrame(test\_features)

data["count"] = test\_labels

data["predicted\_count"] = predictions\_rf

data.to\_csv("submit\_py.csv",index = False)

### 

### **Complete Python File**

#set working directory

import os

os.chdir("D:\edwisor\_project\python\_files")

%matplotlib inline

#Load packages

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

#import scipy as sp

import seaborn as sns

import scipy.stats as stats

#Load data

df\_original = pd.read\_csv("day.csv")

#all the processing will be done on copy of data

df = df\_original.copy()

#rename variables,if required

df.rename(columns={'dteday': 'date', 'yr': 'year','mnth':'month','weathersit':'weather','atemp':'feel\_temp',

'hum':'humidity','cnt':'count'},

inplace=True)

#map the categorical variables

df["season"] = df.season.map({1: "Spring", 2 : "Summer", 3 : "Fall", 4 :"Winter" })

df["weather"] = df.weather.map({1: "weather1", 2 : "weather2", 3 : "weather3", 4 :"weather4" })

df["weekday"] = df.weekday.map({0:"sunday", 1: "monday", 2 : "tuesday", 3 : "wednesday", 4 :"thursday",

5:"friday",6:"saturday"})

#convert data types

conversion\_list = ['season','weather','workingday','holiday','weekday']

df[conversion\_list] = df[conversion\_list].apply(lambda x: x.astype('category'),axis= 0)

DATA PRE-PROCESSING

#removing column instant(index numbers) as it has no corelation to any other variable.

#day,season,workingday,holiday are already derived in the table therfore date attribute not required.

df = df.drop(columns=["date","instant"])

#check for missing values ,if any

print(df.isnull().sum())

#check for duplicates

print(df.duplicated(subset=None, keep='first').sum())

UNIVARIATE ANALYSIS

#season barplot

season\_table = pd.DataFrame({"category":df.season.value\_counts().index, "frequency":df.season.value\_counts().values}).sort\_values('category').reset\_index(drop = True)

labels = ("Spring","Summer","Fall","Winter")

plt.bar(x = season\_table.category, height = season\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(season\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(season\_table.category, season\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('seasons.png')

#plt.close()

#weather barplot

weather\_table = pd.DataFrame({"category":df.weather.value\_counts().index, "frequency":df.weather.value\_counts().values})

labels = ("1:clear","2:cloudy","3:light rain","4:heavy rain")

plt.bar(x = weather\_table.category, height = weather\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(weather\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(weather\_table.category, weather\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('weather.png')

#plt.close()

#holiday barplot

holiday\_table = pd.DataFrame({"category":df.holiday.value\_counts().index, "frequency":df.holiday.value\_counts().values})

labels = ("No holiday","holiday")

plt.bar(x = holiday\_table.category, height = holiday\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(holiday\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(holiday\_table.category, holiday\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('holiday.png')

#plt.close()

#workingday barplot

workingday\_table = pd.DataFrame({"category":df.workingday.value\_counts().index, "frequency":df.workingday.value\_counts().values})

labels = ("working day ","no working day")

plt.bar(x = workingday\_table.category, height = workingday\_table.frequency, align='center',color = 'red', alpha=.5,width = 0.5)

plt.xticks(workingday\_table.category, labels)

plt.ylabel('frequency')

for a,b in zip(workingday\_table.category, workingday\_table.frequency):

plt.text(a, b, str(b))

#plt.show()

plt.savefig('workingday.png',)

#plt.close()

#count boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="count",

data=df).set(ylabel = 'count',title="box plot of count")

plt.savefig('boxplot\_count.png',bbox\_inches='tight')

#temperature boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="temp",

data=df).set(ylabel = 'temperature',title="box plot of temperature")

plt.savefig('boxplot\_temp.png',bbox\_inches='tight')

#humidity boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="humidity",

data=df).set(ylabel = 'humidity',title="box plot of humidity")

plt.savefig('boxplot\_humidity.png',bbox\_inches='tight')

#windspeed boxplot

sns.set\_style('whitegrid')

sns.boxplot(y="windspeed",

data=df).set(ylabel = 'humidity',title="box plot of windspeed")

plt.savefig('boxplot\_windspeed.png',bbox\_inches='tight')

#count distribution plot

sns.distplot(df['count'], hist=True, kde=False,

bins = np.arange(0,10000, 500),color = 'darkred',

hist\_kws={'edgecolor':'black'},

kde\_kws={'linewidth': 2})

MULTIVARIATE ANALYSIS

#boxplot of rental count & season

sns.set\_style('whitegrid')

sns.boxplot(x="season", y="count",

data=df).set(xlabel = 'seasons', ylabel = 'count',title="box plot of count vs seasons")

plt.savefig('count\_vs\_season.png',bbox\_inches='tight')

#boxplot of rental count & weather

sns.boxplot(x = "weather", y="count",

data=df).set(xlabel = 'weather', ylabel = 'count',title="box plot of count vs weather")

plt.savefig('count\_vs\_weather.png',bbox\_inches='tight')

#boxplot of rental count & holiday

sns.boxplot(x="holiday", y="count",

data=df).set(xlabel = 'holiday', ylabel = 'count',title="box plot of count vs holiday")

plt.savefig('count\_vs\_holiday.png',bbox\_inches='tight')

#boxplot of rental count & working day

sns.boxplot(x="workingday", y="count",

data=df).set(xlabel = 'workingday', ylabel = 'count',title="box plot of count vs workingday")

plt.savefig('count\_vs\_workingday.png',bbox\_inches='tight')

#boxplot of rental count & weekday

sns.boxplot(x="weekday", y="count",

data=df).set(xlabel = 'weekday', ylabel = 'count',title="box plot of count vs weekday")

plt.savefig('count\_vs\_weekday.png',bbox\_inches='tight')

#Lineplot of rental count & year

fig, ax = plt.subplots()

fig.set\_size\_inches(11, 8)

sns.set(style="whitegrid")

sns.pointplot(x="year", y="count", data=df,color="#bb3f3f")

plt.savefig("count\_vs\_year.png",bbox\_inches='tight',dpi=100)

#Lineplot of rental count & month

fig, ax = plt.subplots()

fig.set\_size\_inches(11, 8)

sns.set(style="whitegrid")

sns.pointplot(x="month", y="count", data=df,color="#bb3f3f")

plt.savefig("count\_vs\_month.png",bbox\_inches='tight',dpi=100)

# scatter plot of rental v.s. temperature

sns.set(style="whitegrid")

sns.relplot(x="temp", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_temp.png",bbox\_inches='tight',dpi=100)

# line plot of rental v.s. humidity

sns.set(style="whitegrid")

sns.relplot(x="humidity", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_humidity.png",bbox\_inches='tight',dpi=100)

# line plot of rental v.s. wind speed

sns.set(style="whitegrid")

sns.relplot(x="windspeed", y="count", data=df,kind = "line");

plt.savefig("count\_vs\_windspeed.png",bbox\_inches='tight',dpi=100)

Outliers treatment

#save numeric names

cnames = ("count", "temp", "humidity", "windspeed")

for i in cnames:

print(i)

q75, q25 = np.percentile(df.loc[:,i], [75 ,25])

iqr = q75 - q25

min = q25 - (iqr\*1.5)

max = q75 + (iqr\*1.5)

print(min)

print(max)

df = df.drop(df[df.loc[:,i] < min].index)

df = df.drop(df[df.loc[:,i] > max].index)

FEATURE ENGINEERING

numeric\_data = df.select\_dtypes(include=['float64','int64'])

corr\_data = numeric\_data.corr()

fig,ax= plt.subplots()

fig.set\_size\_inches(20,10)

sns.heatmap(corr\_data,vmin = -1 ,vmax = 1, square=True,annot=True)

ANOVA analysis for categorical variables

stats.f\_oneway(df['count'][df['season'] == "Spring"],

df['count'][df['season'] == 'Summer'],

df['count'][df['season'] == 'Fall'],

df['count'][df['season'] == 'Winter'])

stats.f\_oneway(df['count'][df['weather'] == 'weather1'],

df['count'][df['weather'] == 'weather2'],

df['count'][df['weather'] == 'weather3'])

stats.f\_oneway(df['count'][df['workingday'] == 0],

df['count'][df['workingday'] == 1])

stats.f\_oneway(df['count'][df['holiday'] == 0],

df['count'][df['holiday'] == 1])

stats.f\_oneway(df['count'][df['weekday'] == 'sunday'],

df['count'][df['weekday'] == 'monday'],

df['count'][df['weekday'] == 'tuesday'],

df['count'][df['weekday'] == 'wednesday'],

df['count'][df['weekday'] == 'thursday'],

df['count'][df['weekday'] == 'friday'],

df['count'][df['weekday'] == 'saturday'])

#conclusion based on anova,correlation plot

#remove weekday,holiday,workingday,feel\_temp,casual,registered

df = df.drop(columns=["feel\_temp","casual","registered","weekday","holiday","workingday"], axis = 1);

# One-hot encode the data using pandas get\_dummies

df = pd.get\_dummies(df,drop\_first = True)

#reorder columns

cols = df.columns.tolist()

cols = cols[:5] + cols[6:] + cols[5:6]

df = df[cols]

Sampling of data

# Using Skicit-learn to split data into training and testing sets

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

# Split the data into training and testing sets

train, test = train\_test\_split(df,test\_size = 0.20, random\_state = 42)

# Saving feature names for later use

feature\_list = list(df.drop('count', axis = 1).columns)

Model Building

Linear Regression

import statsmodels.api as sm

model\_lr = sm.OLS(train.iloc[:,10], train.iloc[:,0:10]).fit()

model\_lr.summary()

predictions\_lr = model\_lr.predict(test.iloc[:,0:10])

#define error metrics

##1.Root Mean Squared Error Loss

def rmse(predictions, targets):

return np.sqrt(((predictions - targets) \*\* 2).mean())

#2.Root Mean Squared Logarithmic Error Loss

def rmsle(y\_pred, y\_test) :

assert len(y\_test) == len(y\_pred)

return np.sqrt(np.mean((np.log1p(y\_pred) - np.log1p(y\_test))\*\*2))

print("RMSE is:",rmse(predictions\_lr,test["count"]))

print("RMSLE is:",rmsle(predictions\_lr,test["count"]))

Random Forest Model

#separate in to Features and Targets and Convert Data to Arrays

# Labels are the values we want to predict.

train\_labels = np.array(train['count'])

test\_labels = np.array(test['count'])

# Remove the labels from the features(data)

train\_features= train.drop('count', axis = 1)

test\_features= test.drop('count', axis = 1)

# Convert to numpy array

train\_features = np.array(train\_features)

test\_features = np.array(test\_features)

print('Training Features Shape:', train\_features.shape)

print('Training Labels Shape:', train\_labels.shape)

print('Testing Features Shape:', test\_features.shape)

print('Testing Labels Shape:', test\_labels.shape)

from sklearn.ensemble import RandomForestRegressor

# Instantiate model with 1000 decision trees

rf = RandomForestRegressor(n\_estimators = 100, random\_state = 42)

# Train the model on training data

rf.fit(train\_features, train\_labels);

# Use the forest's predict method on the test data

predictions\_rf = rf.predict(test\_features)

#error metrics

print("RMSE is:",rmse(predictions\_rf,test\_labels))

print("RMSLE is:",rmsle(predictions\_rf,test\_labels))

##write to csv file

data = pd.DataFrame(test\_features)

data["count"] = test\_labels

data["predicted\_count"] = predictions\_rf

data.to\_csv("submit\_py.csv",index = False)