PREDICTING BIKE RENTAL COUNT

AUGUST 28, 2018

ASHISH SHARMA

Contents

1 Introduction 2
1.1 Problem Statement 2
1.2 Data 2
2 Methodology 4
2.1 Pre-Processing 4
2.1.1 Variable Identification 4
2.1.2 Univariate Analysis 4
2.1.3 Bi-variate analysis 5
2.1.4 Missing Values Analysis 7
2.1.5 Outlier Analysis 7
2.1.6 Feature Engineering 8
2.2 Modelling 10
2.2.1 Model Selection 10
2.2.2 Multiple Linear Regression 10
2.2.3 Random Forest Model 11
3 Conclusion 12
3.1 Model Evaluation 12
3.1.1 Root Mean Squared Error Loss(RMSE) 12
3.1.2 Root Mean Squared Logarithmic Error Loss(RMSLE) 12
3.2 Model Selection 12
Appendix A - Extra Figures 13
Appendix B - R Code 16
univariate analysis 16
Multivariate Analysis 17
Outlier analysis and treatment 19
Feature Engineering 19
Sampling 20
Model Building 20
Complete R File 22
Appendix c - Python Code 28
Univariate Analysis 28
Multivariate Analysis 29
Outliers treatment 30
Feature Engineering 30
Sampling of data 31
Model Building 31
Complete Python File 33

Chapter 1

1 Introduction

1.1 Problem Statement

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

1.2 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

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

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

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

2 Methodology

2.1 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.

2.1.1 Variable Identification

Here we will identify the predictor variables and target variable(s) and their data types.

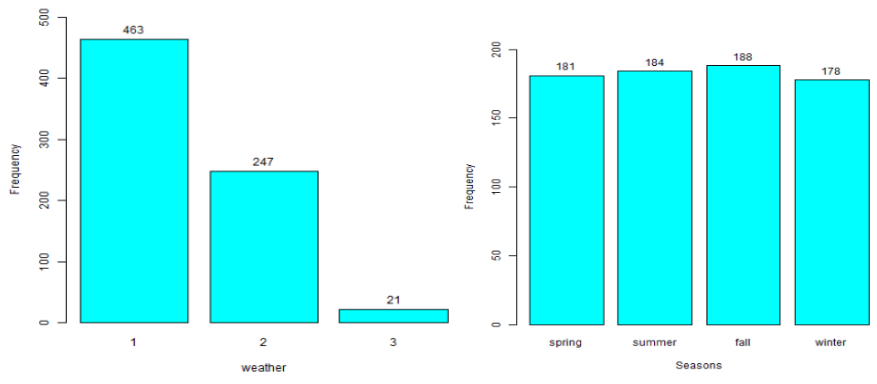
Fig: 2.1.1 Data Type Classification Table

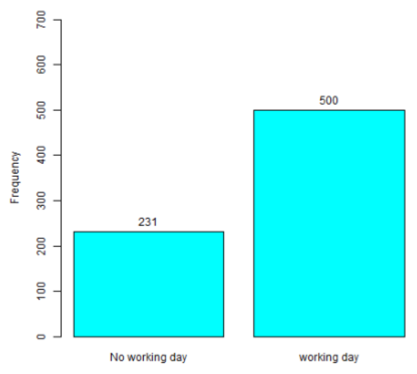
S.No.	Categorical predictor Variables	Continuous Predictor variables
1.	season	year
2.	holiday	Week day
3.	Working day	temperature
4.	weather	Feeling temperature
5.		humidity
6.		Wind speed
7.		Casual users
8.		Registered users
9.		month
Target Variable(s)		
1.	Bike Count (type: continuous)	

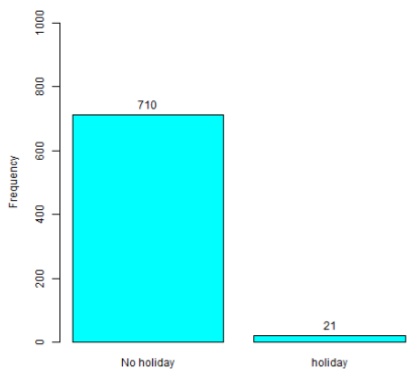
2.1.2 Univariate Analysis

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



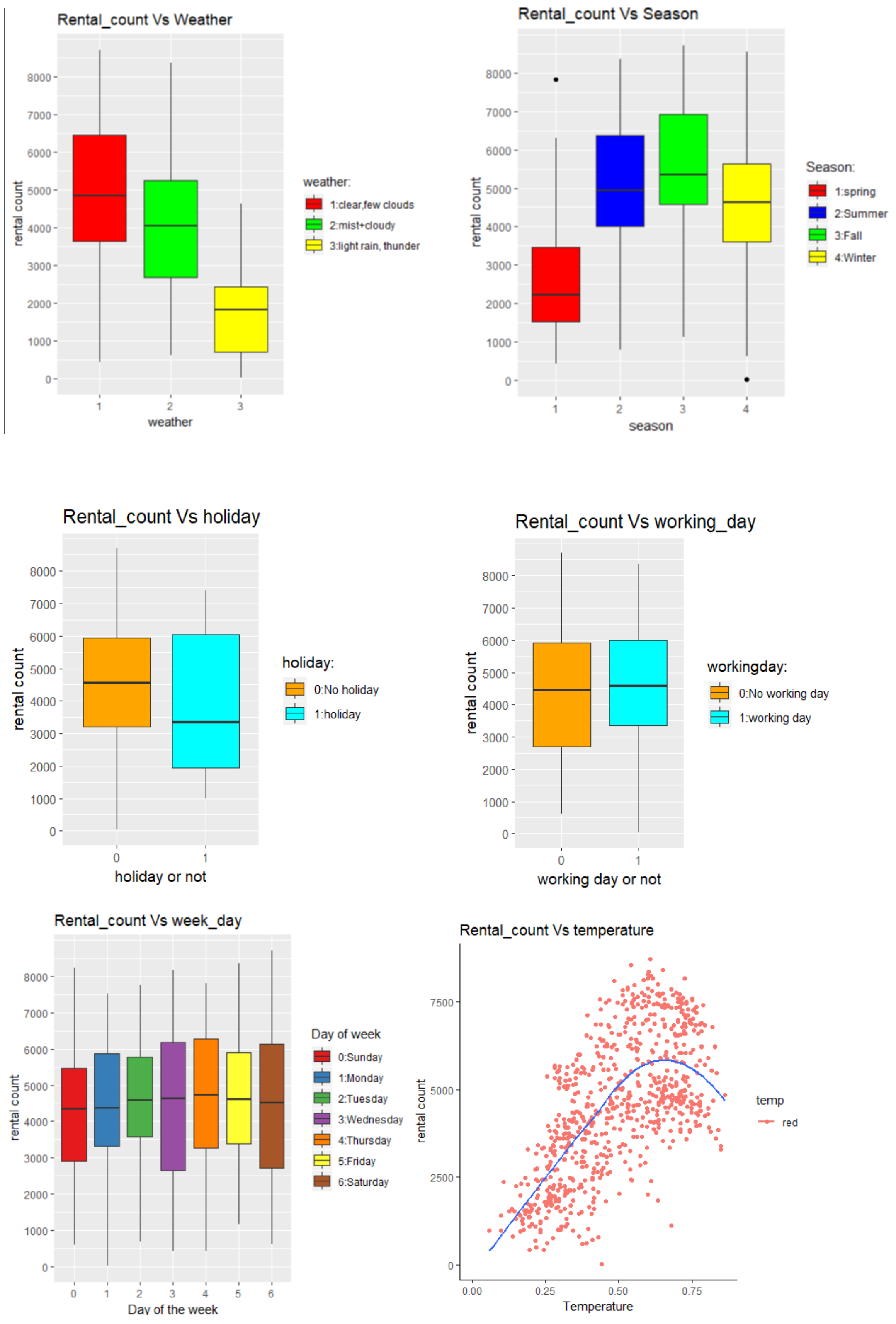


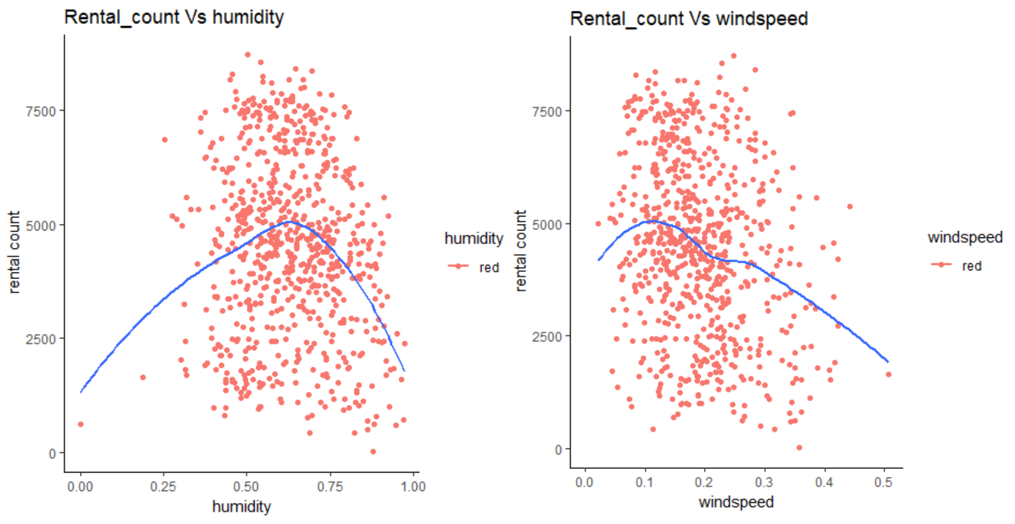


2.1.3 Bi-variate analysis

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





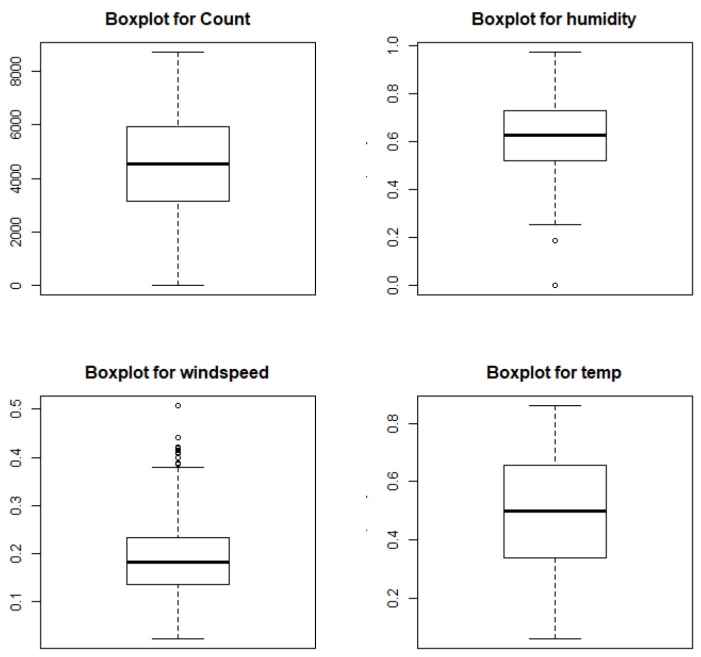
2.1.4 Missing Values Analysis

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.

2.1.5 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



2.1.6 Feature Engineering

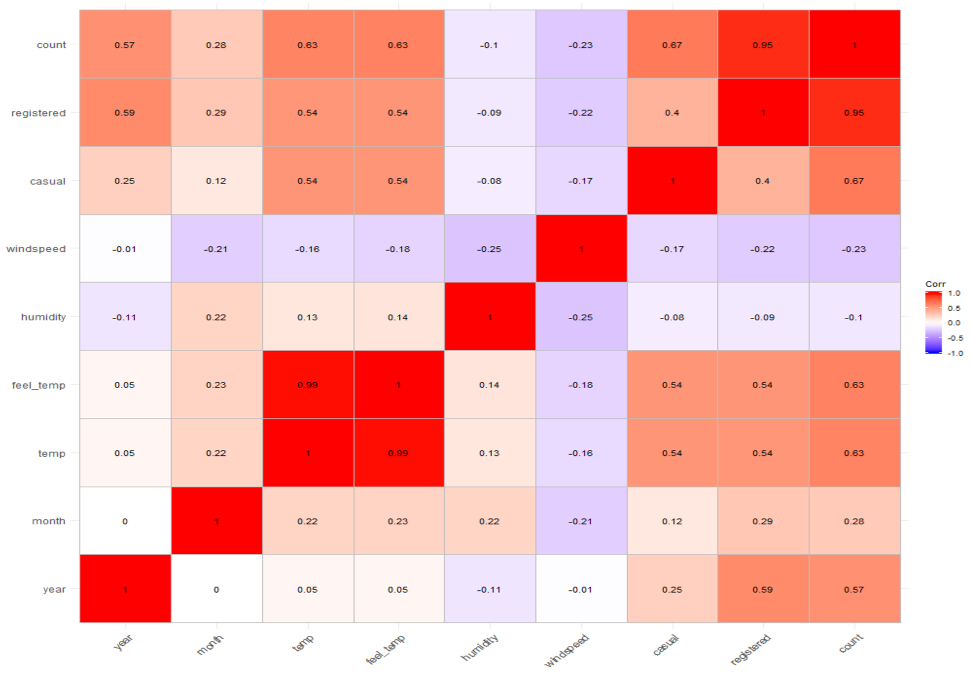
We will determine how strong the relationship is between the predictor variables because there might be a possibility that many variables in our analysis are not important at all to the regression problem. Then we'll select a subset of relevant features (variables, predictors) for use in model construction.

2.1.6.1 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:

a. Temperature and feel temperature are highly correlated.
b. Temperature is positively correlated with count.
c. Wind speed and Humidity have low negative correlation with count.

Figure 2.1.3 correlation plot



2.1.6.2 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.

2.1.6.3 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.

2.1.6.4 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"

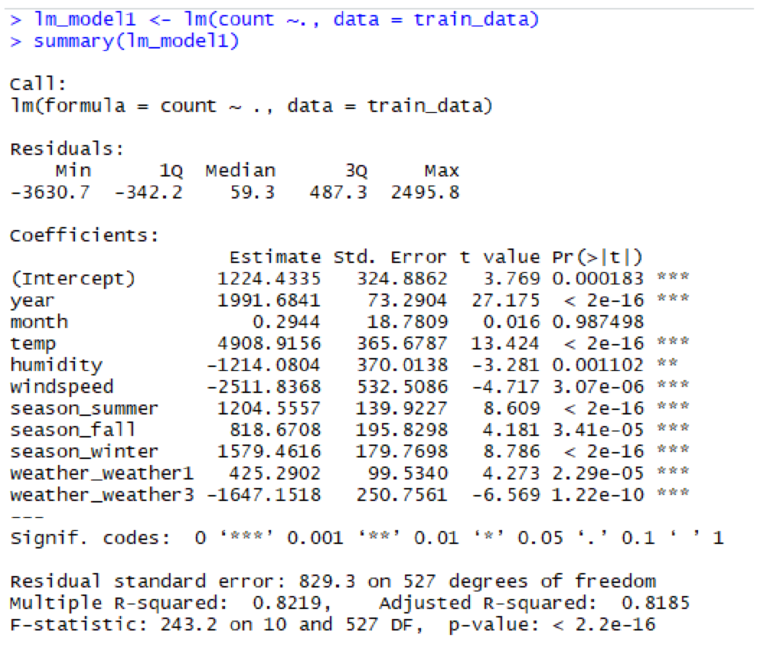
"windspeed"
"season_summer"
"season_fall"
"season_winter"
"weather_weather1"
"weather_weather3"
"count"

2.2 Modelling

2.2.1 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.

2.2.2 Multiple Linear Regression

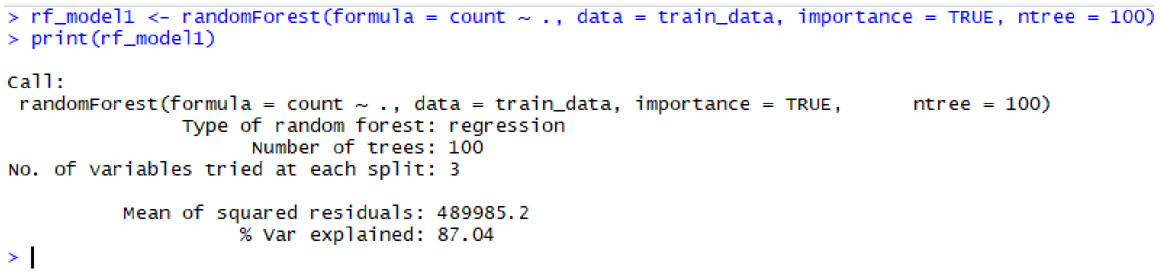


```
> lm_model1 <- lm(count ~., data = train_data)
> summary(lm_model1)

Call:
lm(formula = count ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-3630.7  -342.2    59.3   487.3  2495.8 

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)    
(Intercept)      1224.4335   324.8862   3.769 0.000183 ***
year             1991.6841    73.2904  27.175  < 2e-16 ***
month               0.2944    18.7809   0.016 0.987498    
temp             4908.9156   365.6787  13.424  < 2e-16 ***
humidity        -1214.0804   370.0138  -3.281 0.001102 ** 
windspeed       -2511.8368   532.5086  -4.717 3.07e-06 ***
season_summer    1204.5557   139.9227   8.609  < 2e-16 ***
season_fall       818.6708   195.8298   4.181 3.41e-05 ***
season_winter    1579.4616   179.7698   8.786  < 2e-16 ***
weather_weather1  425.2902    99.5340   4.273 2.29e-05 ***
weather_weather3 -1647.1518   250.7561  -6.569 1.22e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 829.3 on 527 degrees of freedom
Multiple R-squared:  0.8219,	Adjusted R-squared:  0.8185 
F-statistic: 243.2 on 10 and 527 DF,  p-value: < 2.2e-16
```

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.

2.2.3 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.



```
> rf_model1 <- randomForest(formula = count ~ ., data = train_data, importance = TRUE, ntree = 100)
> print(rf_model1)

Call:
 randomForest(formula = count ~ ., data = train_data, importance = TRUE,      ntree = 100) 
               Type of random forest: regression
                     Number of trees: 100
No. of variables tried at each split: 3

          Mean of squared residuals: 489985.2
                    % Var explained: 87.04
> |
```

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

Chapter 3

3 Conclusion

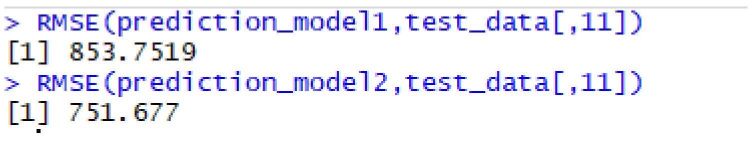
3.1 Model Evaluation

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.

3.1.1 Root Mean Squared Error Loss(RMSE)

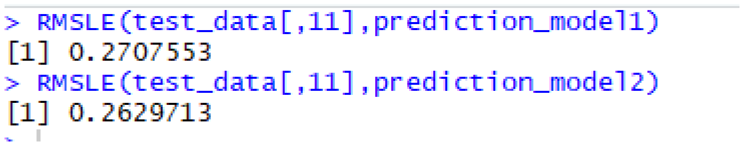
RMSE is one of the evaluation metrics used in regression problems. We will apply this measure to our models that we have generated in the previous section.



```
> RMSE(prediction_model1,test_data[,11])
[1] 853.7519
> RMSE(prediction_model2,test_data[,11])
[1] 751.677
```

3.1.2 Root Mean Squared Logarithmic Error Loss(RMSLE)

RMSLE can be obtained as follows



```
> RMSLE(test_data[,11],prediction_model1)
[1] 0.2707553
> RMSLE(test_data[,11],prediction_model2)
[1] 0.2629713
```

3.2 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

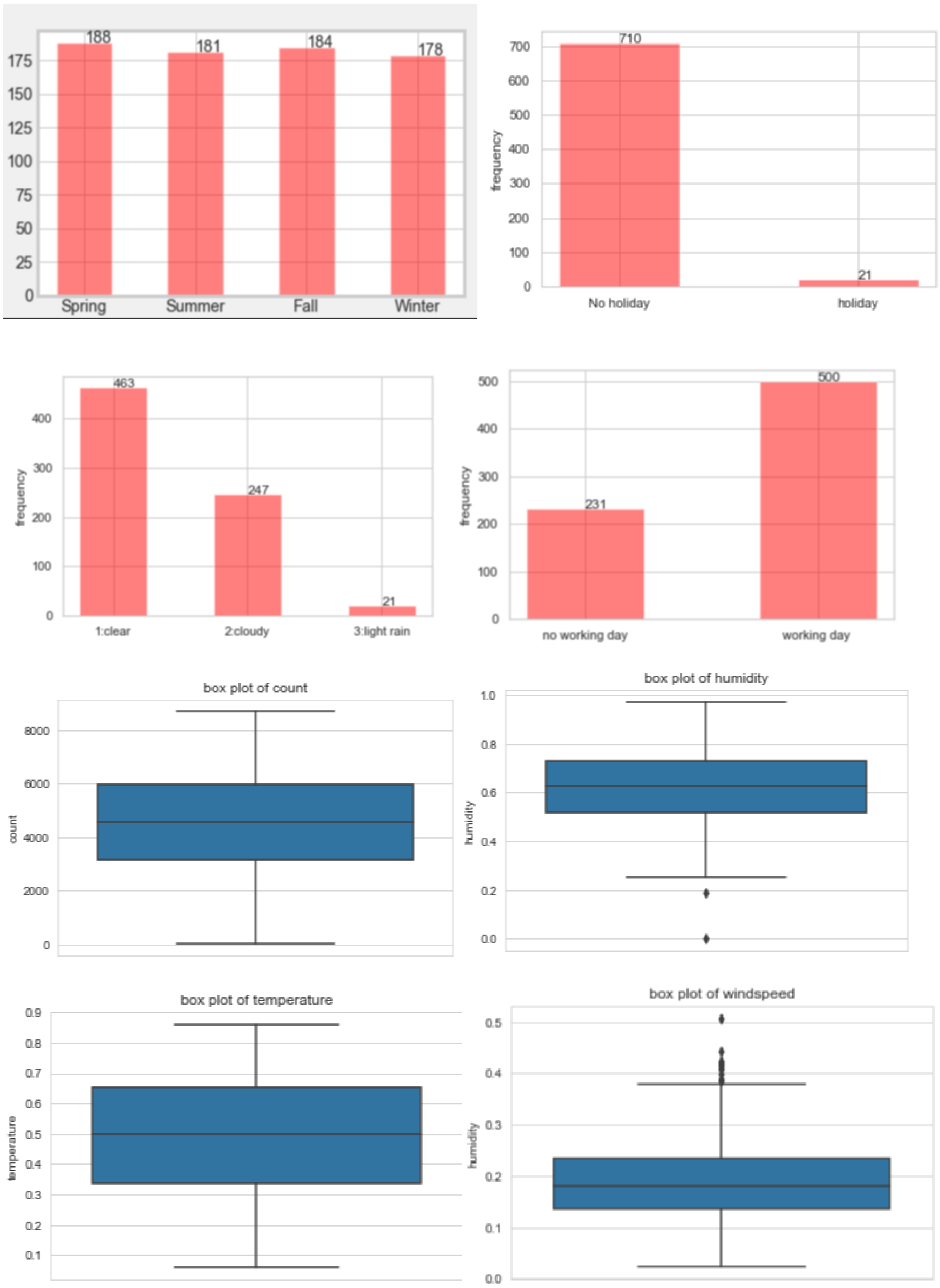
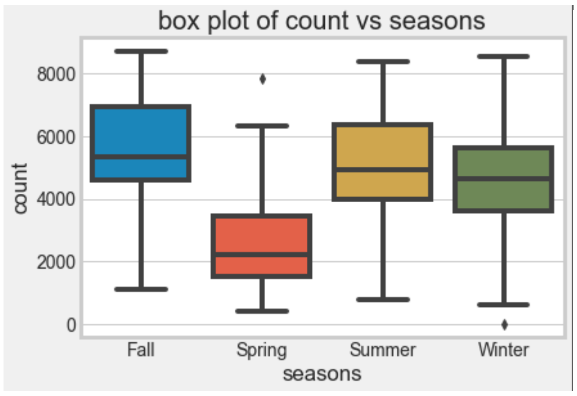
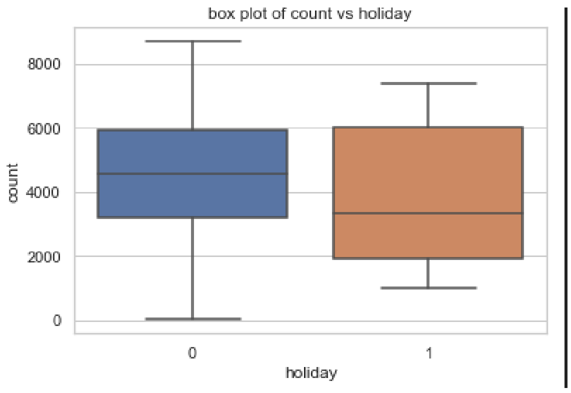
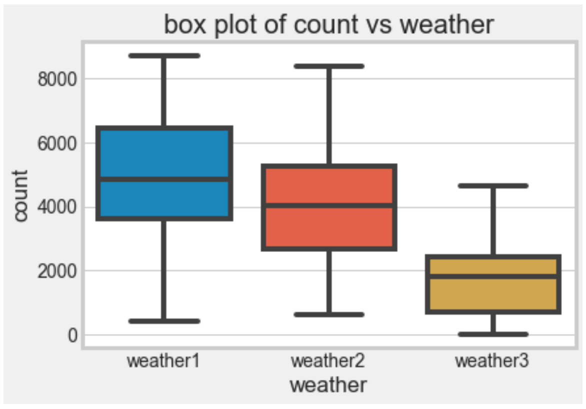
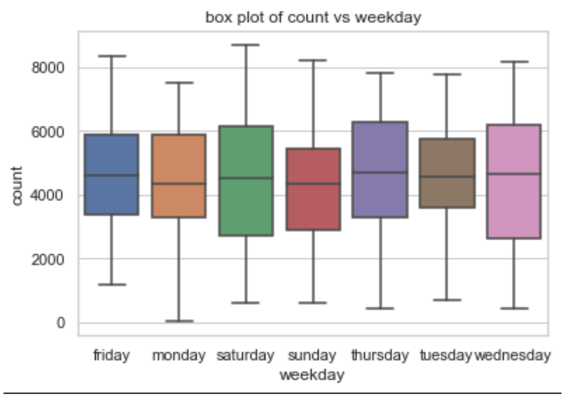


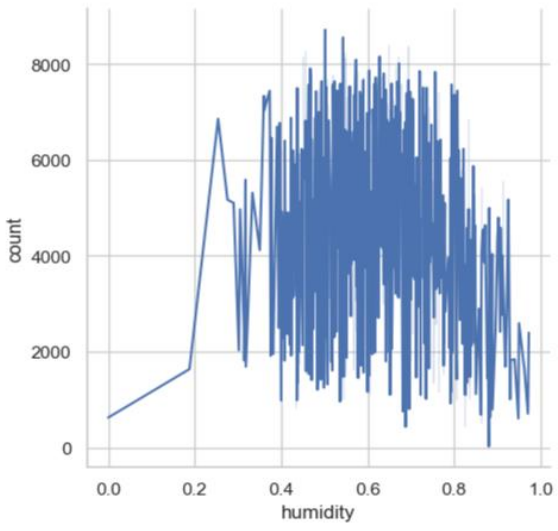
Figure: 3.1: frequency distribution of variables (python version)

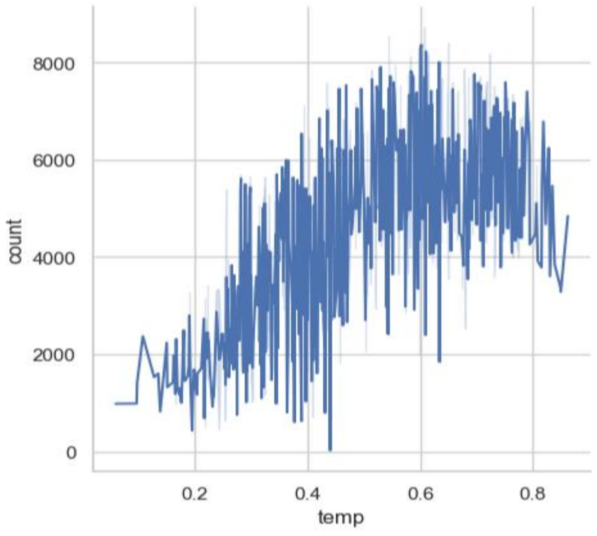

box plot of count vs seasons
count
seasons
Fall
Spring
Summer
Winter
0
2000
4000
6000
8000


box plot of count vs holiday
count
holiday
0
1
0
2000
4000
6000
8000


box plot of count vs weather
count
weather
weather1
weather2
weather3
0
2000
4000
6000
8000


box plot of count vs weekday
count
weekday
friday
monday
saturday
sunday
thursday
tuesday
wednesday
0
2000
4000
6000
8000


count
humidity
0.0
0.2
0.4
0.6
0.8
1.0
0
2000
4000
6000
8000


count
temp
0.2
0.4
0.6
0.8
0
2000
4000
6000
8000

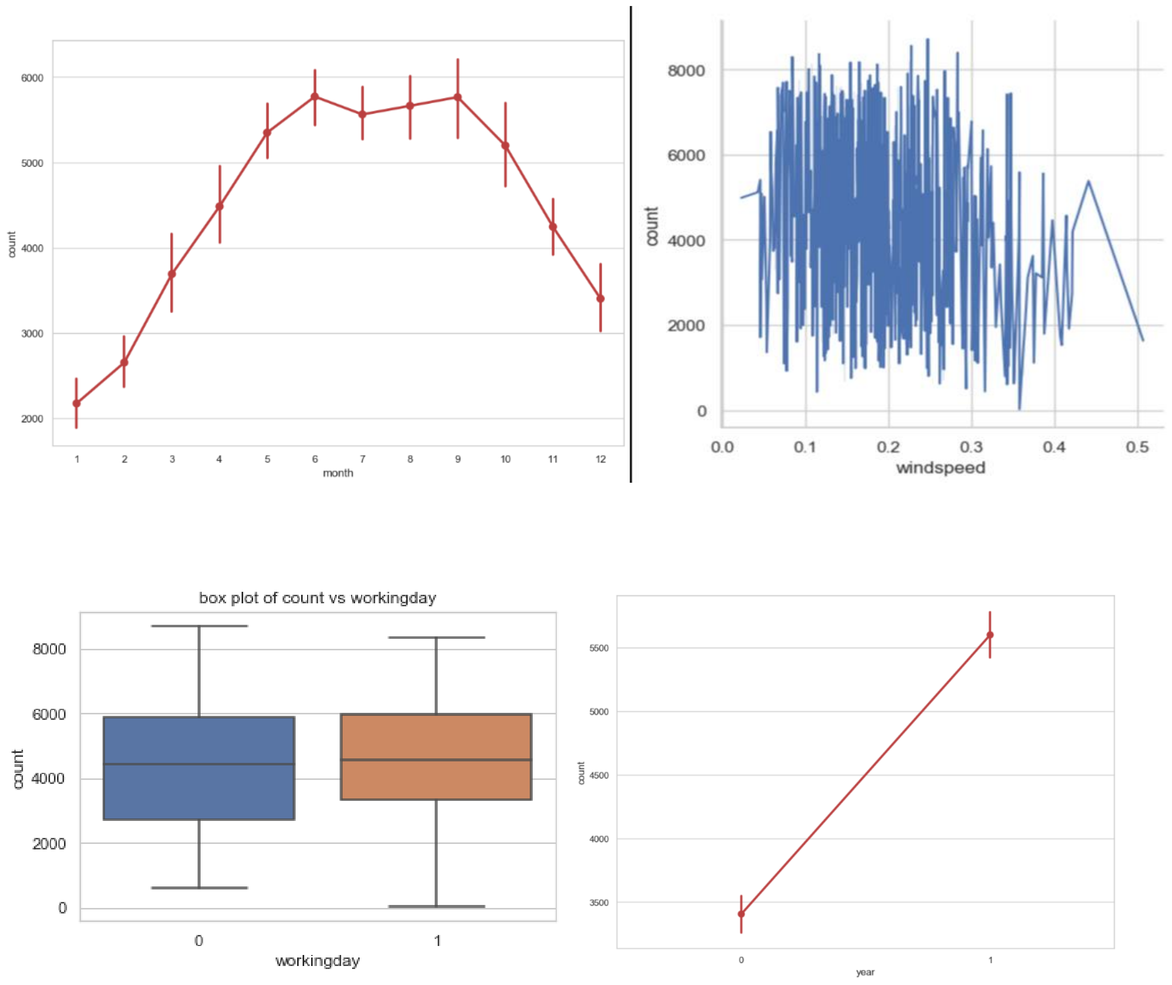


Figure: 3.2: Bi-variate Analysis (python version)

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)
```