BIKE RENTAL COUNT

DATA SCIENCE PROJECT

AARTI NIRMAL

Contents

1.1 Problem statement –

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

1.2 Dataset

Load the .csv file for analyses the data. As we observed that in csv file first column is index so we load the file without file column.

Sample data (Top 10 rows of data)

	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	2011-01-01	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.1604460	331	654	985
2	2011-01-02	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.2485390	131	670	801
3	2011-01-03	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.2483090	120	1229	1349
4	2011-01-04	1	0	1	0	2	1	1	0.200000	0.212122	0.590435	0.1602960	108	1454	1562
5	2011-01-05	1	0	1	0	3	1	1	0.226957	0.229270	0.436957	0.1869000	82	1518	1600
6	2011-01-06	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.0895652	88	1518	1606
7	2011-01-07	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.1687260	148	1362	1510
8	2011-01-08	1	0	1	0	6	0	2	0.165000	0.162254	0.535833	0.2668040	68	891	959
9	2011-01-09	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.3619500	54	768	822
10	2011-01-10	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.2232670	41	1280	1321

In the dataset -

Number of observations: 731

Count of variable:

Independent Variable: 15

Dependent Variable: 1(“cnt”)

Total Count of Variable: 16

Now we have to prepare a model to predict of bike rental count on daily based on the environmental and seasonal settings. And also the **dependent variable is continuous so it is a regression problem**.

Data Attributes:

1. instant: Record index
2. dteday: Date
3. season: Season (1:springer, 2:summer, 3:fall, 4:winter)
4. yr: Year (0: 2011, 1:2012)
5. mnth: Month (1 to 12)
6. holiday: weather day is holiday or not (extracted fromHoliday Schedule)
7. weekday: Day of the week
8. workingday: If day is neither weekend nor holiday is 1, otherwise is 0.

9. weathersit: (extracted fromFreemeteo)
    a. Clear, Few clouds, Partly cloudy, Partly cloudy
    b. Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
    c. Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
    d. Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
10. temp: Normalized temperature in Celsius.
    a. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)
11. atemp: Normalized feeling temperature in Celsius.
    a. The values are derived via (t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale)
12. hum: Normalized humidity. The values are divided to 100 (max)
13. windspeed: Normalized wind speed. The values are divided to 67 (max)
14. casual: count of casual users
15. registered: count of registered users
16. cnt: count of total rental bikes including both casual and registered

1.3Exploratory Data Analysis

```
'data.frame':	731 obs. of  15 variables:
 $ dteday    : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ weekday   : int  6 0 1 2 3 4 5 6 0 1 ...
 $ workingday: int  0 0 1 1 1 1 1 0 0 1 ...
 $ weathersit: int  2 2 1 1 1 1 2 2 1 1 ...
 $ temp      : num  0.344 0.363 0.196 0.2 0.227 ...
 $ atemp     : num  0.364 0.354 0.189 0.212 0.229 ...
 $ hum       : num  0.806 0.696 0.437 0.59 0.437 ...
 $ windspeed : num  0.16 0.249 0.248 0.16 0.187 ...
 $ casual    : int  331 131 120 108 82 88 148 68 54 41 ...
 $ registered: int  654 670 1229 1454 1518 1518 1362 891 768 1280 ...
 $ cnt       : int  985 801 1349 1562 1600 1606 1510 959 822 1321 ...
```

Uniques values of each variable :

1. dteday: 731
2. season: 4
3. yr: 2
4. mnth: 12
5. holiday: 2
6. weekday: 7
7. workingday: 2
8. weathersit: 3
9. temp: 499
10. atemp: 490
11. hum: 595

12. windspeed: 650
13. casual: 606
14. registered: 679
15. cnt: 696

Now we have unique value of each variable. And it clear that in this dataset 7 are categorical variables.

I.e.

1. $ season : int 1 1 1 1 1 1 1 1 1 1 ...
2. $ yr : int 0 0 0 0 0 0 0 0 0 0 ...
3. $ mnth : int 1 1 1 1 1 1 1 1 1 1 ...
4. $ holiday : int 0 0 0 0 0 0 0 0 0 0 ...
5. $ weekday : int 6 0 1 2 3 4 5 6 0 1 ...
6. $ workingday: int 0 0 1 1 1 1 1 0 0 1 ...
7. $ weathersit: int 2 2 1 1 1 1 2 2 1 1 ...

And 8 are continues variable. I.e.

1. $ dteday : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6
2. $ temp : num 0.344 0.363 0.196 0.2 0.227 ...
3. $ atemp : num 0.364 0.354 0.189 0.212 0.229 ...
4. $ hum : num 0.806 0.696 0.437 0.59 0.437 ...
5. $ windspeed : num 0.16 0.249 0.248 0.16 0.187 ...
6. $ casual : int 331 131 120 108 82 88 148 68 54 41 ...
7. $ registered: int 654 670 1229 1454 1518 1518 1362 891 768 1280 ...
8. $ cnt : int 985 801 1349 1562 1600 1606 1510 959 822 1321 ...

Now in EDA we observed that some of the variable not contains significant information. So for convenient and better understand we have to remove those variable. “**dteday**” is not contains much significant information for our model as it is not a time series problem. “**casual**” and “**registered**” are also not contains significant information as sum of these two variable is “**cnt**”. So we need to remove these three variable.

And also rename some variable for better understand.

1. "season"
2. "year"
3. "month"
4. "holiday"
5. "weekday"
6. "workingday"
7. "weathersit"
8. "temperature”
9. "atemp"
10. "humidity"
11. "windspeed"
12. "count"

1.4Data Understanding

For better understand the data we apply some visualization.

1.From season figure1.4.1 we can observe that for season 2,3 and 4 the count of bike is more than season 1. And the count for season2,3 and 4 is between 3500 to 8000.

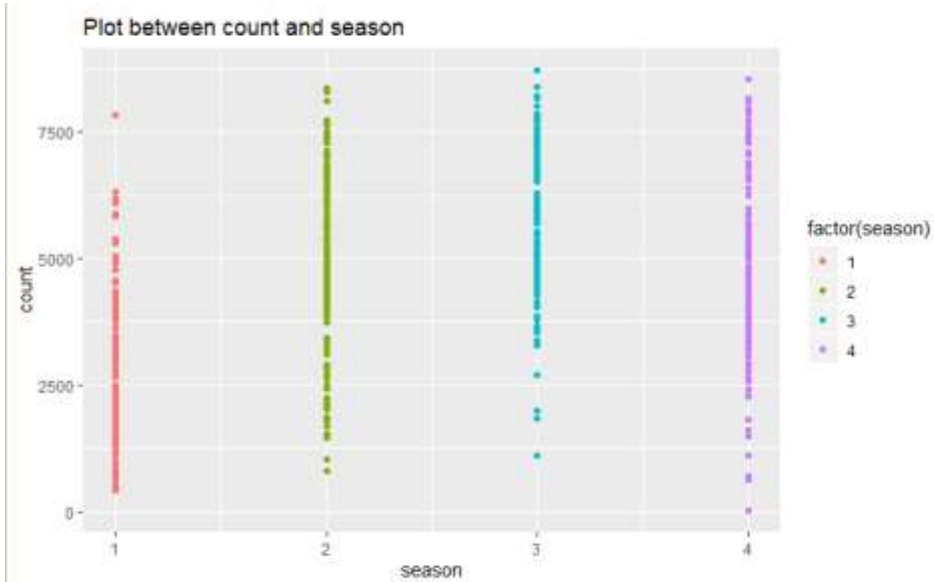


Figure 1.4.1

2.From figure 1.4.2 we can see that from month 4 to 12 the count of bike is heigher.

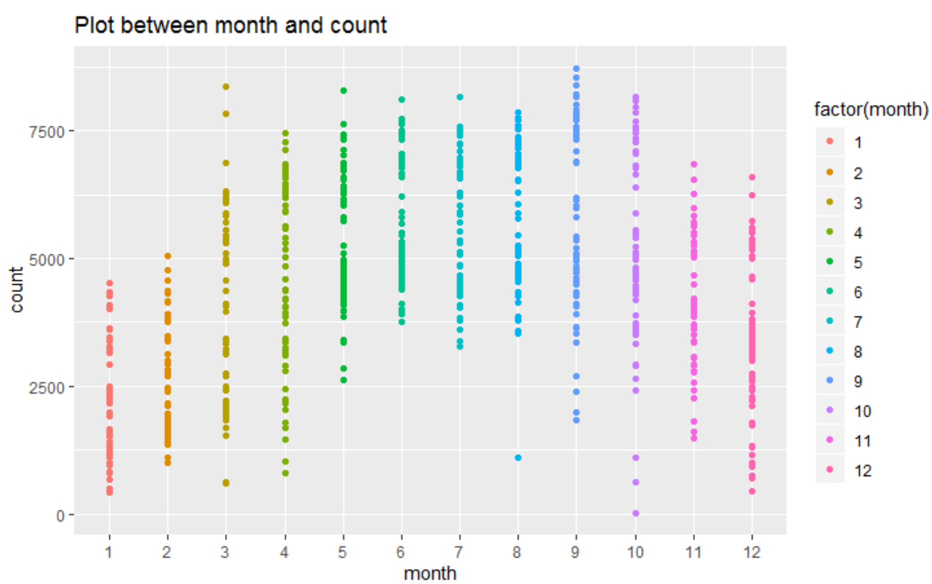


Figure 1.4.2

3.From figure 1.4.3 it is clearly see that count of bike is much higher on holidays.

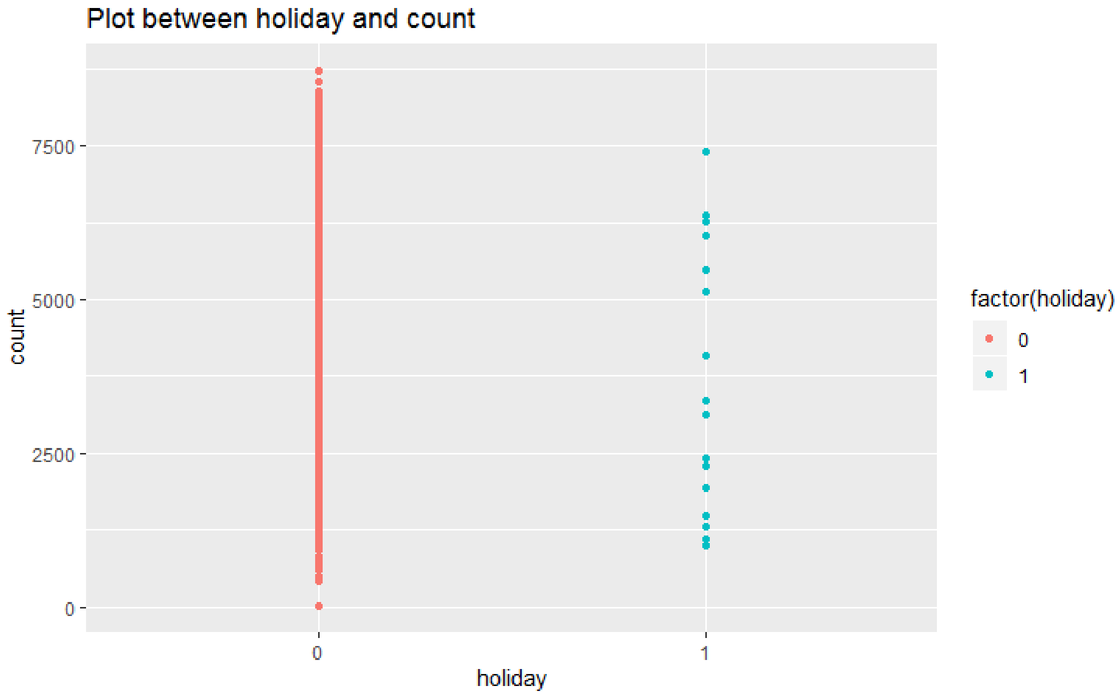


Figure 1.4.3

4.From below figure 1.4.4 it is clear that count of bike is higher in weather 1 and 2.

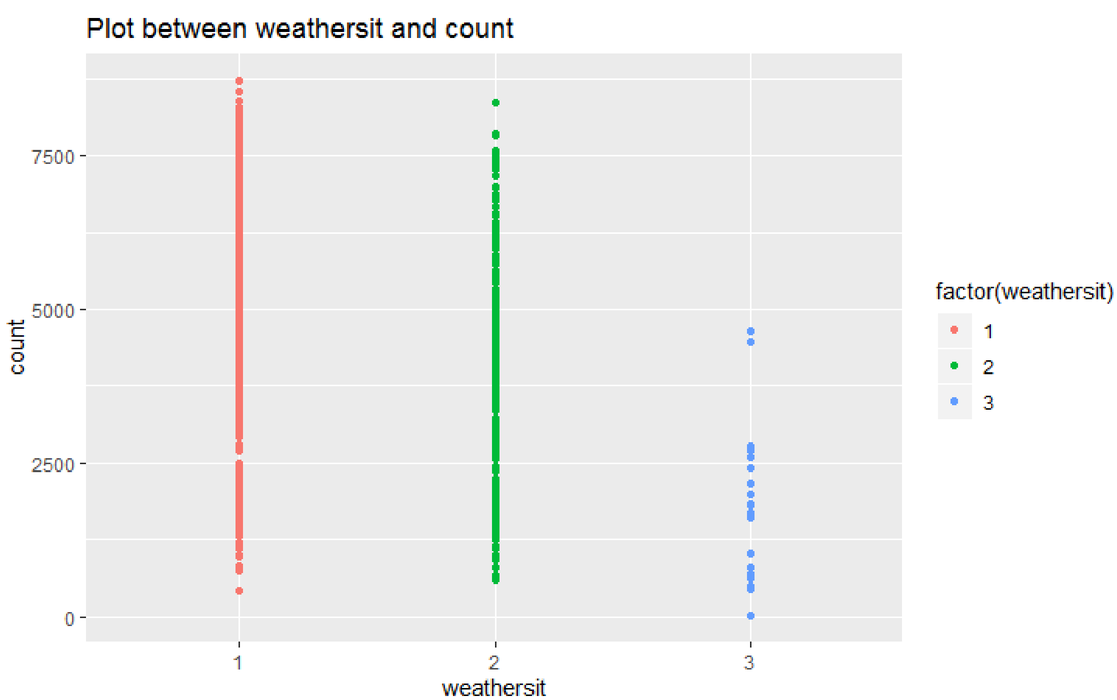


Figure 1.4.4

5. From the below figure 1.4.5 we can observed that the count of bike is more in 0.3 to 0.9 temperature and in 0.50 to 1 humodity.

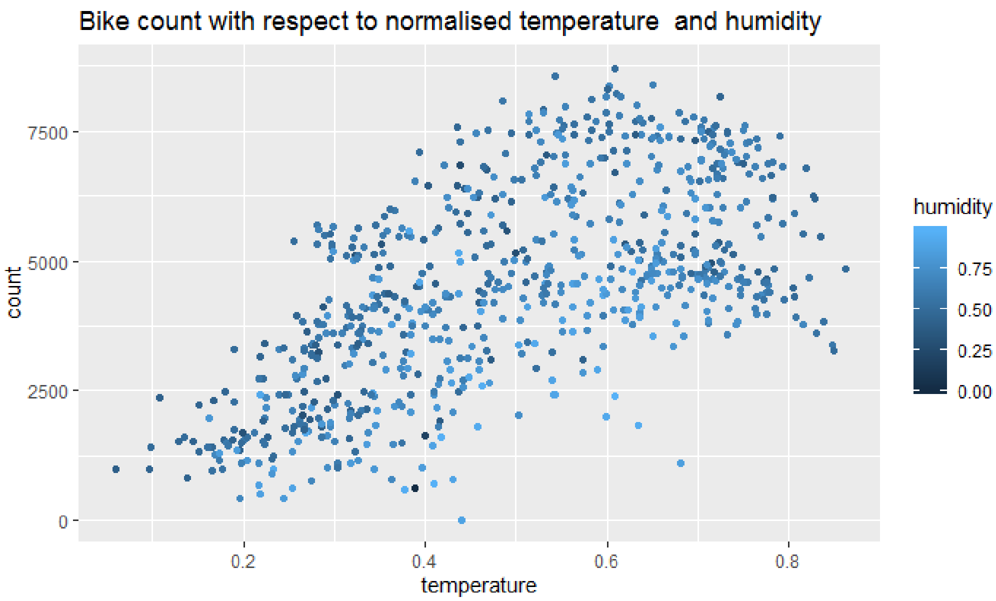


Figure 1.4.5

6.From the below figure 1.4.6 we can observed that count of bikes is high when temperature is 0.2 to 0.9 and humidity is more than 0.50 and windspeed is less than 0.3

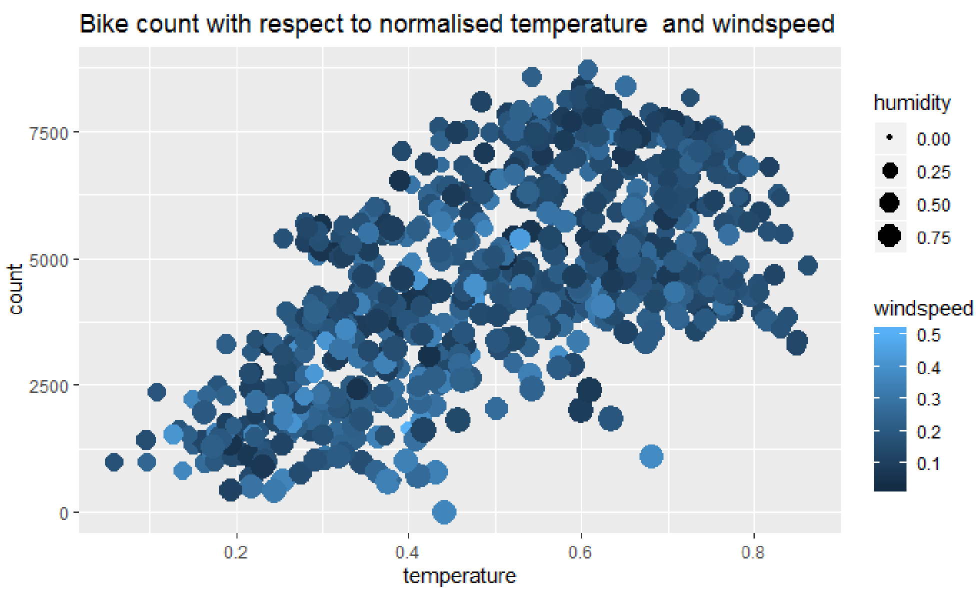


Figure 1.4.6

7.From below figure 1.4.7 we can observed that count of bike is high when temperature is 0.3to 0.9 and season is 2 and 3.

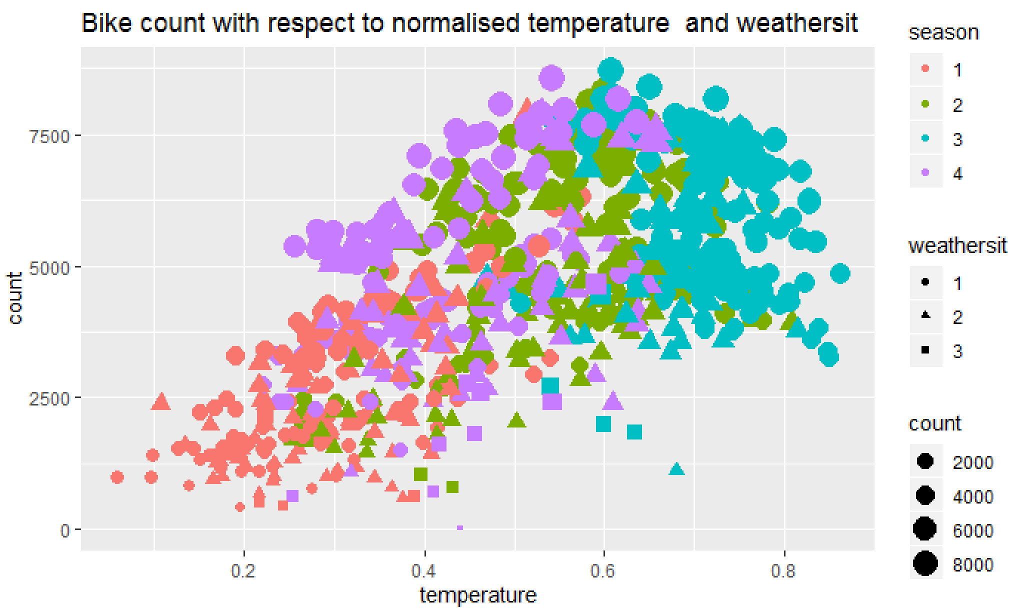

Bike count with respect to normalised temperature and weathersit
count
7500
5000
2500
0
0.2
0.4
0.6
0.8
temperature
season
1
2
3
4
weathersit
1
2
3
count
2000
4000
6000
8000

Figure 1.4.7

2.Methdology

2.1 Data Pre Processing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors. Data preprocessing is a proven method of resolving such issues. Data preprocessing prepares raw data for further processing.
For further process we firstly need to find the distribution of variable because mostly regression analysis require normally distributed data. Figure 2.1.1 and 2.1.2 distribution of variable temperature, humidity, weather and season.

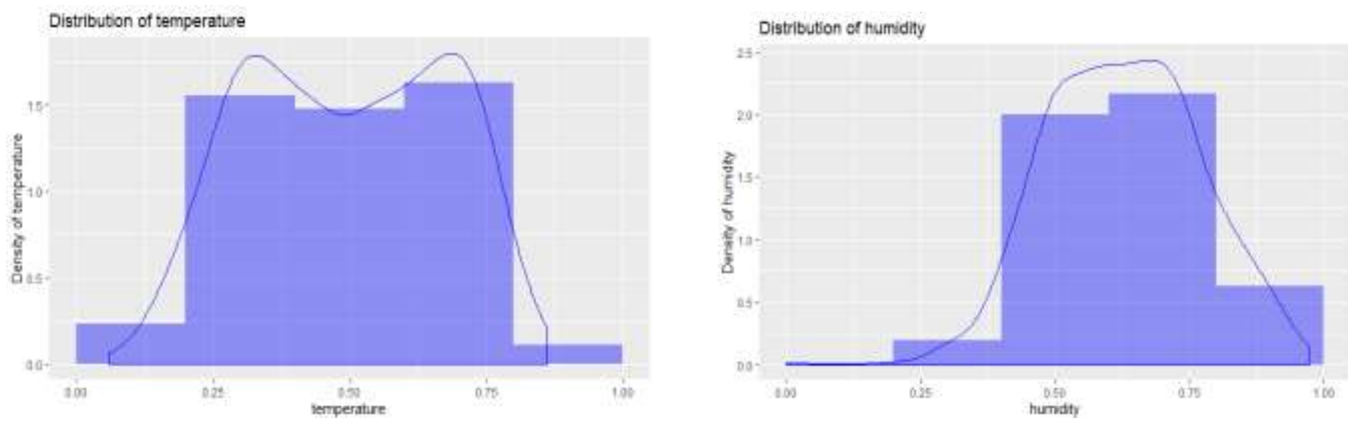


Figure 2.1.1

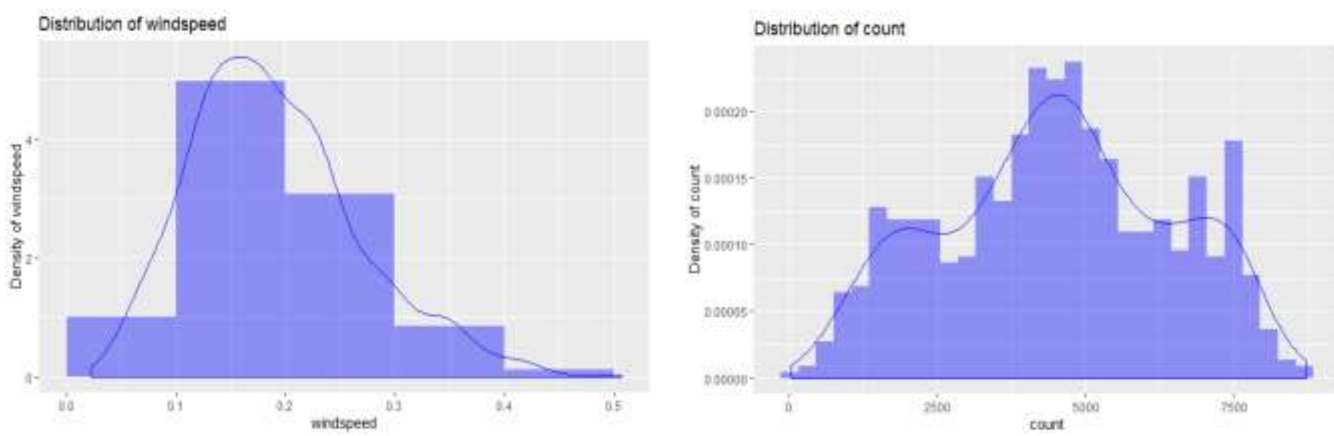


Figure 2.1.2

2.1.1 Missing Value Analysis

When we get the data for analysis, there is chances that values of some observation are blank, that blank value called missing value. These missing values affect in result, that may be in high or low. If a variable contains below 30% of missing values than we can consider that variable in our analysis otherwise we need to reject that.

In our data there isn't missing values. So we don't need to do missing value analysis

- Season 0
- Year 0
- month 0
- holiday 0
- weekday 0
- workingday 0
- weathersit 0
- temperature 0
- atemp 0
- humidity 0
- windspeed 0
- count 0

2.1.2 Outlier Analysis

Observation which lies an abnormal distance from other values from data set is known as outlier. These type of values major effects on our analysis and the result. For this we need to do outlier analysis.

In outlier analysis we need to first detect the outlier with some technique like graphical or statistical and then replace it with NA. In our analysis we use **Box Plot** method to detect the outlier, because it is graphical technique so it is easy to find out outlier. The box plot technique represents distribution of values in 25^{th}, 50^{th} and 75^{th} percentile. And values which are not lies in between 25^{th} to 75^{th} percentile that is outlier.

From figure 2.1.2.1 and 2.1.2.2 we apply box plot for outlier with respect to the target variable "count" and we found that there is no outlier in variable except humidity and windspeed.

So we have replace the outlier with NA i.e. missing value. And we apply KNN imputation to replace this missing values.

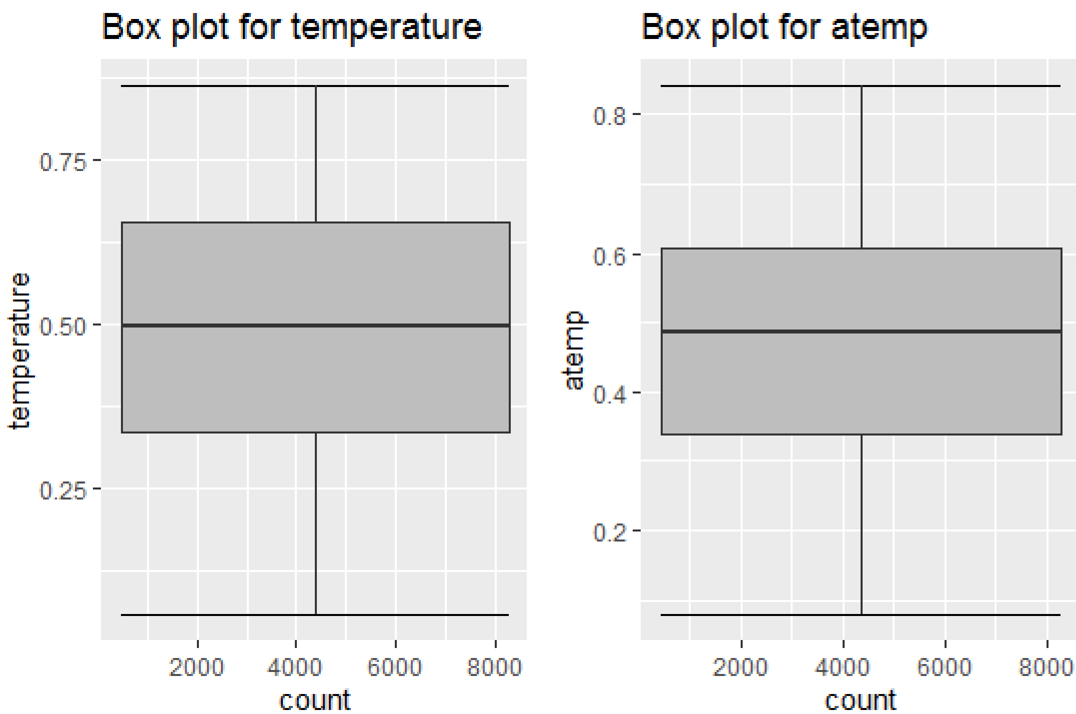


Figure 2.1.2.1

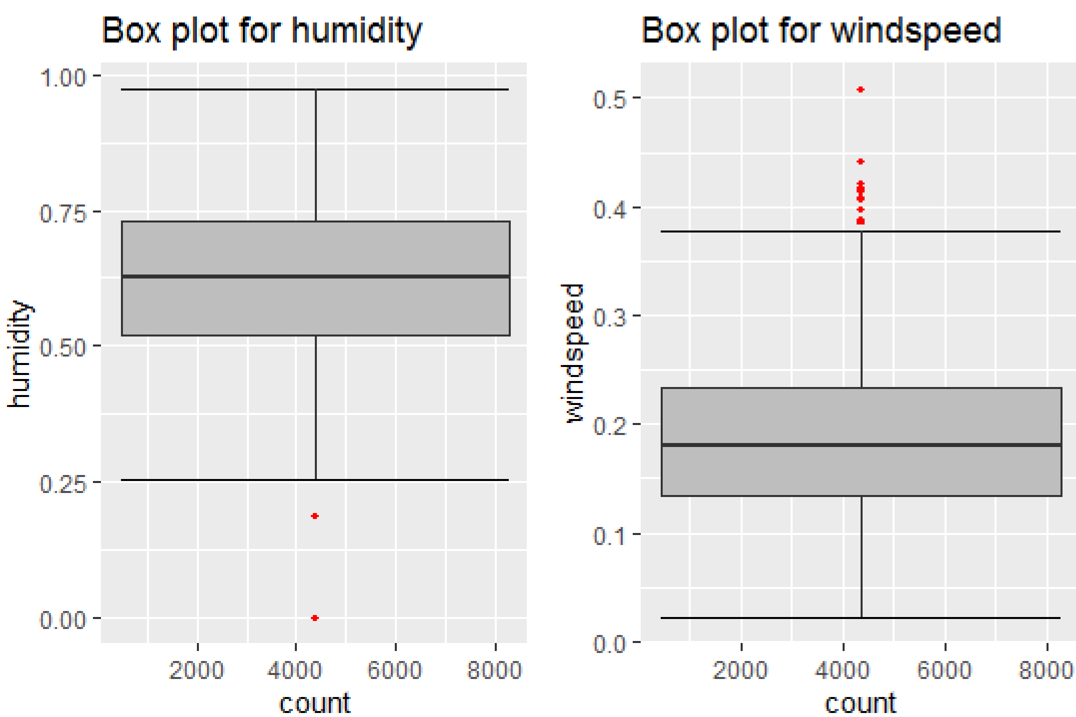


Figure 2.1.2.2

2.1.3 Feature Selection

Feature selection is another technique of pre processing of data. It is clear from its name that select the features from data. It is basically stands for extracting the relevant and meaningful feature out of the data. Its main objective is to remove unrelated attributes from data and reduce the complexity. Because not all the features are carrying significant information or some of them are carrying same information so for that we need to apply feature selection technique. For this we use **Correlation analysis** for numeric variables and **ANOVA test** for categorical variables.

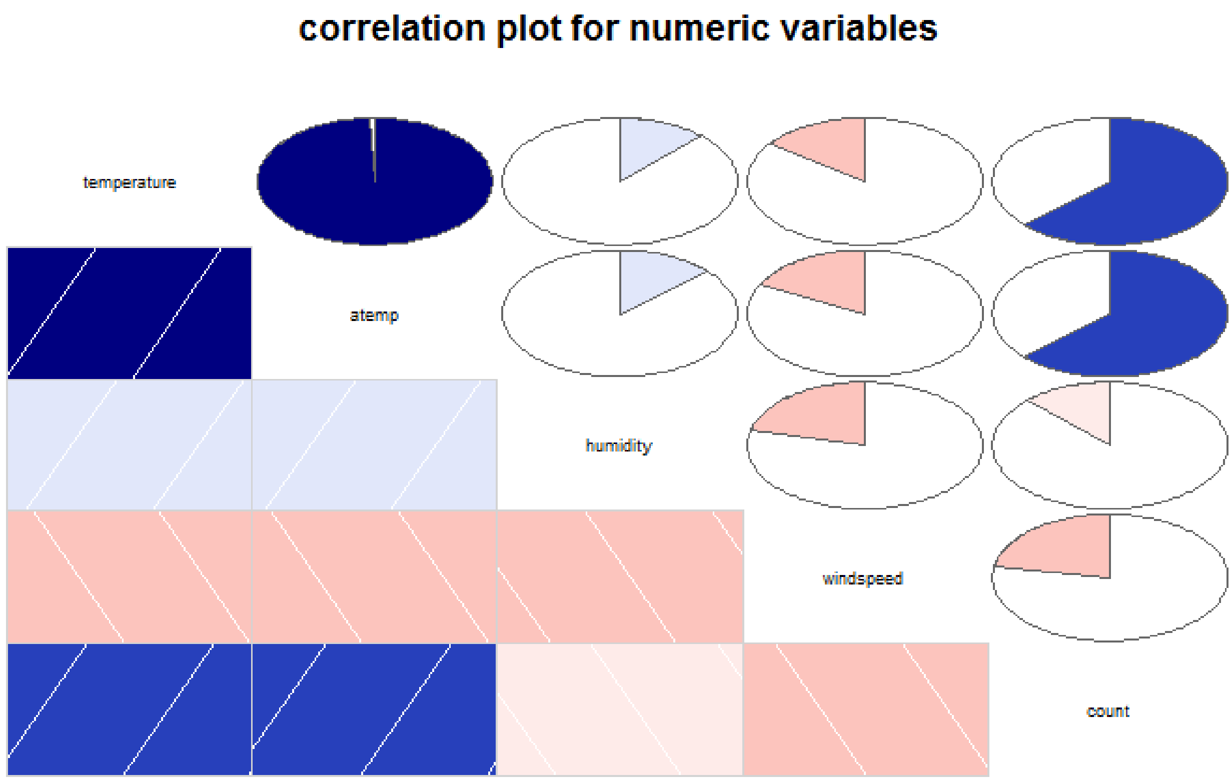


Figure 2.1.3.1

Figure 2.1.3.1 represent the correlation plot of continuous variable. The dark blue color represent that the variables are highly correlated with each other and dark red color represent that the variables are negatively correlated with each other. So according to the above plot variable **temperature** and **atemp** are highly correlated with each other so we can drop one of them, we go with **temperature**.

- ANOVA test for categorical variable with continues variable "count" :

```
[1] "season"
                Df    Sum Sq   Mean Sq F value Pr(>F)
dataBike[, i]    3 9.506e+08 316865289   128.8 <2e-16 ***
Residuals      727 1.789e+09   2460715
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
[1] "year"
                Df    Sum Sq   Mean Sq F value Pr(>F)
```

```
dataBike[, i]   1 8.798e+08 879828893   344.9 <2e-16 ***
Residuals     729 1.860e+09   2551038
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

[1] "month"
               Df    Sum Sq  Mean Sq F value Pr(>F)
dataBike[, i]  11 1.070e+09 97290206    41.9 <2e-16 ***
Residuals     719 1.669e+09  2321757
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

[1] "holiday"
               Df    Sum Sq  Mean Sq F value Pr(>F)
dataBike[, i]   1 1.280e+07 12797494   3.421 0.0648 .
Residuals     729 2.727e+09  3740381
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

[1] "weekday"
               Df    Sum Sq Mean Sq F value Pr(>F)
dataBike[, i]   6 1.766e+07 2943170   0.783  0.583
Residuals     724 2.722e+09 3759498

[1] "workingday"
               Df    Sum Sq  Mean Sq F value Pr(>F)
dataBike[, i]   1 1.025e+07 10246038   2.737 0.0985 .
Residuals     729 2.729e+09  3743881
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

[1] "weathersit"
               Df    Sum Sq   Mean Sq F value Pr(>F)
dataBike[, i]   2 2.716e+08 135822286   40.07 <2e-16 ***
Residuals     728 2.468e+09   3389960
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

With above ANOVA test we can conclude that holiday, weekday and workingday have p value greater than 0.05 so we can drop them.

After correlation and ANOVA we have following variables :

Continues Variables :

1. "temperature"
2. "humidity"
3. "windspeed"
4. "count"

Categorical Variables :

1. "season"
2. "year"
3. "month"
4. "weathersit"

2.1.4 Feature Scaling

Feature Scaling or Standardization: It is a step of Data Pre Processing which is applied to independent variables or features of data. It basically helps to normalise the data within a particular range. Sometimes, it also helps in speeding up the calculations in an algorithm[1].
Real world dataset contains features that highly vary in magnitudes, units, and range. Normalisation should be performed when the scale of a feature is irrelevant or misleading and not should Normalise when the the scale is meaningful[1].

The algorithms which use Euclidean Distance measure are sensitive to Magnitudes. Here feature scaling helps to weigh all the features equally.

	Temperature	**Humidity**	**Windspeed**
Count	731	731	731
Min	0.05913	0.2542	0.02239
25th percentile	0.33708	0.5223	0.13495
Median	0.49833	0.6292	0.17973
Mean	0.49538	0.6294	0.18703
75th percentile	0.65542	0.7302	0.23087
Max	0.86167	0.9725	0.37811

2.2Model Development

Now we have pre processed data. With this pre processed data we develop model to predict result. For this firstly we divide the data into train and test. Perform the algorithm on train data to develop model and get prediction and then apply that model on test data. Chose the algorithm which provide more accurate result.

2.2.1 Decision Tree

It is supervised machine learning algorithm. A predictive model based on a branching series of Boolean tests. It can be used for classification and regression. There are number of different types of decision trees that can be used in Machine learning algorithms.

Decision tree is a rule. Each branch connects nodes with “and” and multiple branches are connected by “or”. Extremely easy to understand by the business users. Build some intuitions about your customer base. E.g. “are customers with different family sizes truly different?”

2.2.2 Random Forest

To build n number of trees to have more accuracy on dataset.

The forest why because we build n number of decision trees. Random because to build any decision tree we are going to select randomly n number of observations.

For each decision tree, we use different-different observation from same dataset. RF called as an ensemble that consists of many decision trees. It can be used for classification and regression.

It will give you the estimate of what variable are important in classification which help us to classify or predict any value for the new test case.

2.2.3 Linear Regression

Linear regression only use for regression data not for classification data.

Prediction Model

– Simple linear regression

– Multiple linear regression

• Describe relationship among variables

• The one simple case is where a dependent variable may be related to independent or explanatory variable

3. Conclusion

From model develop we apply different machine algorithms and predict the result. Now we conclude which model is more accurate for bike count.

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. For this we calculate MAPE and Rsquared for each model.

The mean absolute percentage error (**MAPE**), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics. Mean absolute percentage error is commonly used as a loss function for regression problems and in model evaluation, because of its very intuitive interpretation in terms of relative error.

R-squared is a statistical measure that represents the goodness of fit of a regression model. The ideal value for r-square is 1. The closer the value of r-square to 1, the better is the model fitted.
R-square is a comparison of residual sum of squares *(SS_{res})* with total sum of squares*(SS_{tot})*. Total sum of squares is calculated by summation of squares of perpendicular distance between data points and the average line.
Here the result of each model in python and R.

In Python

	Model Name	MAPE train	MAPE Test	rsquare train	rsquare test
0	Decision Tree	62.260133	36.948093	0.677563	0.646470
1	Linear Regression	43.781407	20.042233	0.837646	0.838132
2	Random Forest	15.682924	19.827128	0.981282	0.888468

In R

	Model	MAPE.Train	MAPE.Test	RSquare.Train	RSquare.Test
1	Decision Tree	53.17167	23.52019	0.8181892	0.8430761
2	Ramdon Forest	23.73341	10.27263	0.9635951	0.9744973
3	Linear Regression	44.75091	17.39293	0.8231024	0.8886749

3.2 Model Selection

From above model evaluation we should go with **Random Forest** because from both python and R we get the lowest MAPE for both train and test data and RSquared values which is nearest to 1.

4.Code

4.1 R code

```
#Remove previous data to start new analysis
rm(list = ls())

#set working directory
setwd("I:/DATA Scientist Assignments/Bike rental project")

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

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

#Check weather directory has been update or not
getwd()

#Load the data for analysis
dataBike = read.csv("day.csv", header = T)[-1]

#Extract sample top 10 data
head(dataBike, 10)

#----------------------------------Exploratory Analysis---------------------------------#
#Structure of data
str(dataBike)
length(colnames(dataBike))
names(dataBike)

#remove variables those are not contains significant information
dataBike = subset(dataBike, select = -c(dteday,casual,registered))

names(dataBike)

#Rename the variable
names(dataBike)[names(dataBike) == "yr"] = "year"
names(dataBike)[names(dataBike) == "mnth"] = "month"
names(dataBike)[names(dataBike) == "hum"] = "humidity"
names(dataBike)[names(dataBike) == "cnt"] = "count"
names(dataBike)[names(dataBike) == "temp"] = "temperature"
```

```
#convert into factor data type
dataBike$season = as.factor(dataBike$season)
dataBike$year = as.factor(dataBike$year)
dataBike$holiday = as.factor(dataBike$holiday)
dataBike$weathersit = as.factor(dataBike$weathersit)
dataBike$workingday = as.factor(dataBike$workingday)
dataBike$month = as.factor(dataBike$month)
dataBike$weekday = as.factor(dataBike$weekday)

#---------------------Data understanding--------------------#
#load library ggplot2 form apply graphical views
library(ggplot2)
#plot b/w count and season
ggplot(dataBike, aes(x = temperature, y = count)) +
  scale_x_continuous(breaks = seq(0, 1, by = 0.2))+
  geom_point(aes(color = humidity))+
  labs( title = "Bike count with respect to normalised temperature  and humidity")

ggplot(dataBike, aes(x = temperature, y = count, color = windspeed, size = humidity)) +
  scale_x_continuous(breaks = seq(0, 1, by = 0.2))+
  geom_point()+
  labs( title = "Bike count with respect to normalised temperature  and windspeed")

ggplot(dataBike, aes(x = temperature, y = count, color = season, size = count, shape = weathersit)) +
  scale_x_continuous(breaks = seq(0, 1, by = 0.2))+
  geom_point()+
  labs( title = "Bike count with respect to normalised temperature  and weathersit")

ggplot(dataBike, aes(x = windspeed)) +
  geom_histogram(aes(y =..density..),
                 breaks = seq(0, 0.5, by = 0.1),
                 fill="blue",
                 alpha = .4,position="dodge") +
  geom_density(col=4) +
  labs(title="Distribution of windspeed") +
  labs(x="windspeed", y="Density of windspeed") +
  theme(legend.position="top")

#--------------------------------Data Pre Processing--------------------------#
#-----------------Missing value analysis------------------#
missingValue = data.frame(apply(dataBike,2, function(x){sum(is.na(x))}))
rm(missingValue)
```

```
#---------------Outlier Analysis----------------------#
#data manupulation: convert string categories into factor numeric
for(i in 1:ncol(dataBike)){
  if(class(dataBike[,i]) == 'factor'){
    dataBike[,i] = factor(dataBike[,i], labels = (1:length(levels(factor(dataBike[,i])))))
  }
}
rm(i)

#Boxplot to find outlier
library(ggplot2)
number_index = sapply(dataBike, is.numeric)
numeric_data = dataBike[, number_index]
cnames = colnames(numeric_data)
for(i in 1:length(cnames)){
  assign(paste0("DB", i), ggplot(aes_string(y = (cnames[i]), x = "count"), data = subset(dataBike))+
                      stat_boxplot(geom = "errorbar", width = 0.5) +
                      geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                                   outlier.size=1, notch=FALSE) +
                      theme(legend.position="bottom")+
                      labs(y=cnames[i],x="count")+
                      ggtitle(paste("Box plot for",cnames[i])))
}
rm(i)

## Plotting plots together
gridExtra::grid.arrange(DB1, DB2,DB3, ncol=3)
gridExtra::grid.arrange(DB4,DB5,ncol=2)

#Remove outlier using boxplot
temp = dataBike
dataBike = temp

for (i in cnames){
  val = dataBike[,i][dataBike[,i] %in% boxplot.stats(dataBike[,i])$out]
  print(val)
  dataBike[,i][dataBike[,i] %in% val] = NA
}

#Actual value = 0.318333
#mean  0.6297861
#median  0.6294766
```

```
#knn 0.4374127
missingValue = data.frame(apply(dataBike,2,function(x){ sum(is.na(x))}))
dataBike$humidity[62]
dataBike$humidity[is.na(dataBike$humidity)] = mean(dataBike$humidity, na.rm = T)
dataBike$humidity[is.na(dataBike$humidity)] = median(dataBike$humidity, na.rm = T)
dataBike = knnImputation(dataBike, k=3)

#----------------feature selection-----------------#
#Correltion analysis for continous variales
corrgram(dataBike[,number_index], order = F ,upper.panel = panel.pie,
         text.panel = panel.txt, main ="correlation plot for numeric variables")
#variable temperature and atemp are highly correlated with each other

#ANOVA test for categorical variables
factorVal = sapply(dataBike, is.factor)
factorVariables = dataBike[, factorVal]
cat_variables = names(factorVariables)
 for(i in cat_variables){
   print(i)
   anovaresult = summary(aov(formula = count~dataBike[,i],dataBike))
   print(anovaresult)
 }

#remove variable after apply correlation and ANOVA
dataBike = subset(dataBike, select = -c(atemp,holiday,weekday,workingday))

#-------------feature scaling----------------------#
cat_del_ind = sapply(dataBike, is.numeric)
cat_del = dataBike[, cat_del_ind]
cnames_del = names(cat_del)
#skewness test
library(propagate)
for(i in cnames_del){
  print(i)
  skew = skewness(dataBike[,i])
  print(skew)
}
hist(dataBike$temperature, col = "blue", xlab = "temperature", ylab = "Frequency",
                    main = "histogram of teperature")
hist(dataBike$humidity, col = "blue", xlab = "Humidity", ylab = "Frequency",
                    main = "histogram of humidity")
hist(dataBike$windspeed, col = "blue", xlab = "windspeed", ylab = "Frequency",
                    main = "histogram of windspeed")
```

```
#summary
for(i in cnames_del){
  print(summary(dataBike[,i]))
}
#as summary the data is in normalised form so no need to scaling

#write the pre processed data to drive
write.csv(dataBike, "dataBike_count.csv", row.names = FALSE)


#----------------------Model Developemnt---------------------#
#Clean the environment
rmExcept("dataBike")

#MAPE
#calculate MAPE
MAPE = function(y, y1){
  mean(abs((y - y1)/y))
}

#R square
rsquare = function(y,y1){
  cor(y,y1)^2
}

#Divide data into train and test using stratified sampling method
set.seed(1234)
train.index = sample(1:nrow(dataBike), .80 * nrow(dataBike))
train = dataBike[ train.index,]
test  = dataBike[-train.index,]

#---------------------Decision tree---------------#
#Load Libraries
library(rpart)
library(MASS)

# ##rpart for regression
DT_model = rpart(count ~ ., data = train, method = "anova")

#Predict for train cases
train_DT = predict(DT_model, train[-8])
```

```
#predict for test cases
test_DT = predict(DT_model, test[-8])

#MAPE
MAPE_DT_train = (MAPE(train[,8], train_DT))*100
#MAPE_DT_train = 53.17%
MAPE_DT_test = (MAPE(test[,8], test_DT))*100
#MAPE_DT_test = 23.52%

#Rsquare
rquare_train_DT = rsquare(train[,8], train_DT)
#rquare_train = 0.8181892
rquare_test_DT = rsquare(test[,8], test_DT)
#rquare_test =  0.8430761

#------------------Random Forest-----------------#
library(randomForest)
#delevelop model using random forest
RF_model = randomForest(count~., dataBike, nTree = 500, importance = TRUE)

#apply on train data
RF_train_predict = predict(RF_model, train[,-8])

#apply on test data
RF_test_predict = predict(RF_model, test[,-8])

#MAPE for train
RF_MAPE_train = (MAPE(train[,8], RF_train_predict))*100
#MAPE 23.73%

#MAPE for test
RF_MAPE_test = (MAPE(test[,8], RF_test_predict))*100
#MAPE 10.27%

#RSquare for train
RSquare_train_RF= rsquare(train[,8], RF_train_predict)
#Rsquare 0.9635951

#RSquare for test
RSquare_test_RF = rsquare(test[,8], RF_test_predict)
#Rsquare 0.9744973

#---------------------Linear Regression---------------#
```

```
library(usdm)
cnames = c("temperature", "humidity" ,"windspeed")
vif(dataBike[,cnames])
vifcor(dataBike[,cnames], th = 0.9)

#develop linear regression model
LR_model = lm(count~., data = train)
summary(LR_model)

#apply on train data
LR_train = predict(LR_model, train[,-8])

#apply on test
LR_test = predict(LR_model, test[,-8])

#MAPE for train
LR_MAPE_train = (MAPE(train[,8],LR_train))*100
#MAPE 44.75%

#MAPE for test
LR_MAPE_test = (MAPE(test[,8], LR_test))*100
#MAPE 17.39%

#Rsquare for train
rsquare_train_LR = rsquare(train[,8],LR_train)
#0.8231024

#Rsquare for test
rsquare_test_LR = rsquare(test[,8], LR_test)
#0.8886749

#------------------Result-------------------#
result = data.frame(Model = c('Decision Tree', 'Ramdon Forest', 'Linear Regression'),
                    'MAPE Train' = c(MAPE_DT_train,RF_MAPE_train,LR_MAPE_train),
                    'MAPE Test' = c(MAPE_DT_test,RF_MAPE_test,LR_MAPE_test),
                    'RSquare Train' = c(rquare_train_DT,RSquare_train_RF,rsquare_train_LR),
                    'RSquare Test' = c(rquare_test_DT,RSquare_test_RF,rsquare_test_LR))
write.csv(result, "Result,csv", row.names = FALSE)
###############Thank You#####################
```

4.2 Python Code

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns
from random import randrange, uniform
```

```
os.chdir("I:\DATA Scientist Assignments\Bike rental project")
```

```
os.getcwd()
```

```
'I:\\DATA Scientist Assignments\\Bike rental project'
```

```
#Load data
dataBike = pd.read_csv("day.csv")
```

```
#view sample data(top 5 rows)
dataBike.head()
```

	temperature	humidity	windspeed	count	season_1.0	season_2.0	season_3.0	season_4.0	year_0.0	year_1.0	...	month_6.0	month_7.0	month_8.0	mor
0	0.344167	0.805833	0.160446	985.0	1	0	0	0	1	0	...	0	0	0	
1	0.363478	0.696087	0.248539	801.0	1	0	0	0	1	0	...	0	0	0	
2	0.196364	0.437273	0.248309	1349.0	1	0	0	0	1	0	...	0	0	0	
3	0.200000	0.590435	0.160296	1562.0	1	0	0	0	1	0	...	0	0	0	
4	0.226957	0.436957	0.186900	1600.0	1	0	0	0	1	0	...	0	0	0	

Explorartory Data Analysis

```
type(dataBike)
```

```
pandas.core.frame.DataFrame
```

```
dataBike.shape
```

```
(731, 16)
```

```
dataBike.dtypes
```

```
instant         int64
dteday         object
season          int64
yr              int64
mnth            int64
holiday         int64
weekday         int64
workingday      int64
weathersit      int64
temp          float64
atemp         float64
hum           float64
windspeed     float64
casual          int64
registered      int64
cnt             int64
dtype: object
```

```
dataBike.columns
```

```
Index(['instant', 'dteday', 'season', 'yr', 'mnth', 'holiday', 'weekday',
       'workingday', 'weathersit', 'temp', 'atemp', 'hum', 'windspeed',
       'casual', 'registered', 'cnt'],
      dtype='object')
```

```
dataBike.nunique()
```

```
instant       731
dteday        731
season          4
yr              2
mnth           12
holiday         2
weekday         7
workingday      2
weathersit      3
temp          499
atemp         690
hum           595
windspeed     650
casual        606
registered    679
cnt           696
dtype: int64
```

```
#drop variables which are not carrying significant information
#drop "instant" as it is only index number
dataBike = dataBike.drop(['instant'], axis = 1)
```

```
#drop "dteday" as it's contain only date
dataBike = dataBike.drop(['dteday'], axis = 1)
```

```
#drop "casual" and "registered" because sum of both these variable is "cnt"
dataBike = dataBike.drop(['casual', 'registered'], axis = 1)
```

```
dataBike.shape
```

```
(731, 12)
```

```
dataBike.columns
```

```
Index(['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday',
       'weathersit', 'temp', 'atemp', 'hum', 'windspeed', 'cnt'],
      dtype='object')
```

```
#renames variable for better understand
dataBike = dataBike.rename(columns = {'yr':'year','mnth':'month','temp':'temperature','hum':'humidity','cnt':'count'})
```

```
dataBike.columns
```

```
Index(['season', 'year', 'month', 'holiday', 'weekday', 'workingday',
       'weathersit', 'temperature', 'atemp', 'humidity', 'windspeed', 'count'],
      dtype='object')
```

```
#seperate continues and categorical variables
#continuse
cnames = ['temperature', 'atemp', 'humidity', 'windspeed', 'count']
```

Data Pre Processing

Missing Value Analysis

```
missing_val = pd.DataFrame(dataBike.isnull().sum())
```

```
missing_val
```

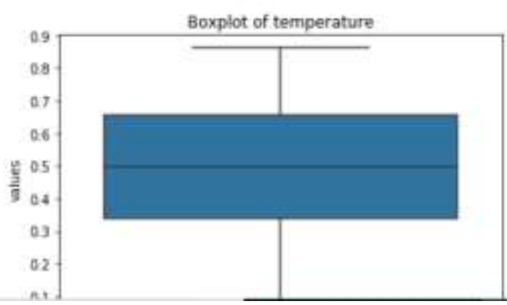
	0
season	0
year	0
month	0
holiday	0
weekday	0
workingday	0
weathersit	0
temperature	0
atemp	0
humidity	0
windspeed	0

Outlier Analysis

```
df = dataBike.copy()
dataBike = df.copy()
```

```
for i in cnames :
    print(i)
    sns.boxplot(y = dataBike[i])
    plt.xlabel(i)
    plt.ylabel("values")
    plt.title("Boxplot of "+i)
    plt.show()
```

temperature



```
#calulate iqr, min and max
for i in cnames:
    print(i)
    q75, q25 = np.percentile(dataBike.loc[:,i], [75 ,25])
    iqr = q75 - q25
    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print("Minimum : "+ str(min))
    print("Maximum : "+ str(max))
    print("IQR : "+ str(iqr))
```

```
temperature
Minimum : -0.14041600000000015
Maximum : 1.1329160000000003
IQR : 0.3183330000000001
atemp
Minimum : -0.06829675000000018
Maximum : 1.0147412500000002
IQR : 0.2707595000000001
humidity
Minimum : 0.20468725
Maximum : 1.0455212500000002
IQR : 0.21020850000000002
windspeed
Minimum : -0.012446750000000034
Maximum : 0.38061125
IQR : 0.0982645
count
Minimum : -1054.0
Maximum : 10162.0
IQR : 2804.0
```

```
#Replace with NA
for i in dataBike.columns :
    dataBike.loc[dataBike[i] < min,i] = np.nan
    dataBike.loc[dataBike[i] > max,i] = np.nan
```

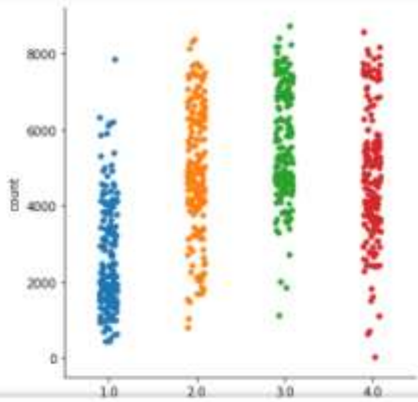
```
#find missing values
missing_val = pd.DataFrame(dataBike.isnull().sum())

#Reset index
missing_val = missing_val.reset_index()
```

```
missing_val
```

	index	0
0	season	0
1	year	0
2	month	0
3	holiday	0
4	weekday	0
5	workingday	0
6	weathersit	0
7	temperature	0
8	atemp	0
9	humidity	0
10	windspeed	0

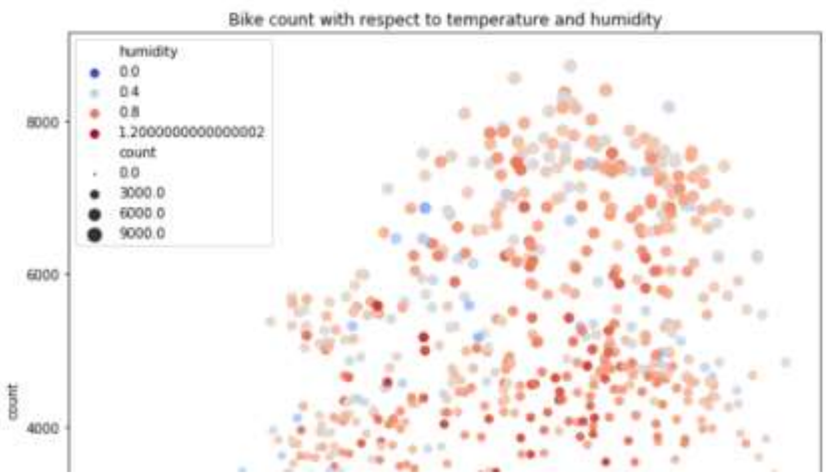
Data Understanding

```
In [26]: for i in cat_cnames :
             sns.catplot(x=i, y="count", data = dataBike)
             fname = str(i)+'.pdf'
             plt.savefig(fname)
```



From above plot we can observed that season 2,3,4 have more bike count than season 1.And if there is a holiday than the count is more. And the count of bike is more in weather 1.2.

```
f, ax = plt.subplots(figsize=(10,10))
sns.scatterplot(x="temperature", y = "count",
                hue = "humidity", size = "count",
                palette = "coolwarm", sizes = (1,100), linewidth=0,
                data = dataBike, ax=ax)
plt.title("Bike count with respect to temperature and humidity")
plt.xlabel("temperature")
plt.ylabel("count")
plt.savefig("countTempHumidity Plot.pdf")
```



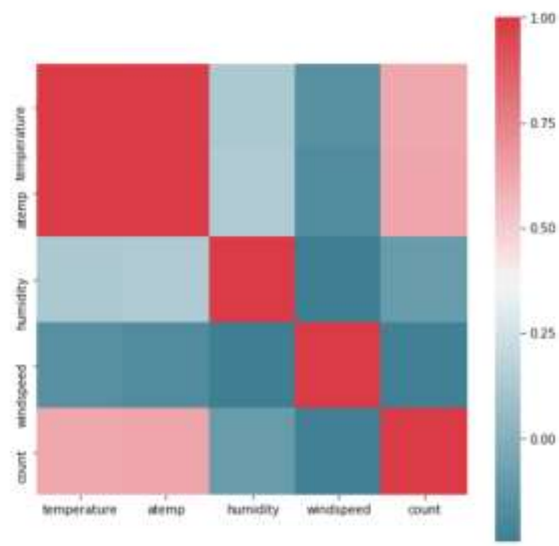
Feature Selection

```
#correlation analysis for continuse variables
#extract only numerical variable
data_num = dataBike.loc[:,cnames]

corrAna = data_num.corr()
```

```
corrAna
```

	temperature	atemp	humidity	windspeed	count
temperature	1.000000	0.991702	0.126963	-0.157944	0.627494
atemp	0.991702	1.000000	0.139988	-0.183643	0.631066
humidity	0.126963	0.139988	1.000000	-0.248489	-0.100659
windspeed	-0.157944	-0.183643	-0.248489	1.000000	-0.234545
count	0.627494	0.631066	-0.100659	-0.234545	1.000000

```
#Plot using seaborn library
f, ax = plt.subplots(figsize=(8, 8))
sns.heatmap(corrAna, mask=np.zeros_like(corrAna, dtype=np.bool), cmap=sns.diverging_palette(220, 10, as_cmap=True),
            square=True, ax=ax)
```



From the above plot we can observed that temperature and atemp are highly correlated with each other. So we can drop one of them, we drop atemp.

```
#ANOVA test for categorical variable and target numeric variable
import statsmodels.api as sm
from statsmodels.formula.api import ols

label = 'count'
for i in cat_cnames :
    frame = label + '~' + i
    model = ols(frame, data = dataBike).fit()
    anova = sm.stats.anova_lm(model,typ=2)
    print(anova)
```

```
                sum_sq     df           F        PR(>F)
season    4.517974e+08    1.0  143.967653  2.133997e-30
Residual  2.287738e+09  729.0         NaN           NaN
                sum_sq     df           F        PR(>F)
year      8.798289e+08    1.0  344.890586  2.483540e-63
Residual  1.859706e+09  729.0         NaN           NaN
                sum_sq     df          F        PR(>F)
month     2.147445e+08    1.0  62.004625  1.243112e-14
Residual  2.524791e+09  729.0        NaN           NaN
                sum_sq     df         F    PR(>F)
holiday   1.279749e+07    1.0  3.421441  0.064759
Residual  2.726738e+09  729.0       NaN       NaN
                sum_sq     df         F    PR(>F)
weekday   1.246109e+07    1.0  3.331091  0.068391
Residual  2.727074e+09  729.0       NaN       NaN
                  sum_sq     df         F    PR(>F)
workingday  1.024604e+07    1.0  2.736742  0.098495
Residual    2.729289e+09  729.0       NaN       NaN
                  sum_sq     df          F        PR(>F)
weathersit  2.422888e+08    1.0  70.729298  2.150976e-16
Residual    2.497247e+09  729.0        NaN           NaN
```

```
#Dimension Reduction
dataBike = dataBike.drop(["atemp","holiday", "weekday","workingday"], axis = 1)
```

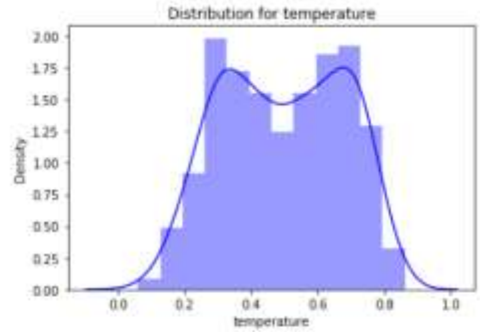
```
dataBike.shape
```

```
(731, 8)
```

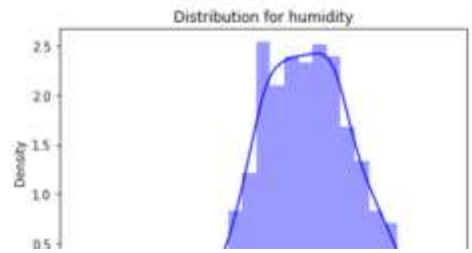
```
#update continuse and categorical variables
#continuse
cnames = ['temperature', 'humidity', 'windspeed', 'count']

#categorical
cat_cnames = ['season', 'year', 'month', 'weathersit']
```

Feature Scaling

```
#Checking weather the data is normaly distributed or not

for i in cnames :
    print(i)
    sns.distplot(dataBike[i], bins ='auto', color = 'blue')
    plt.title("Distribution for "+i)
    plt.ylabel("Density")
    plt.show()
```



humidity




```
dataBike.describe()
```

	season	year	month	weathersit	temperature	humidity	windspeed	count
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.627894	0.190486	4504.348837
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.142429	0.077498	1937.211452
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.000000	0.022392	22.000000
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.520000	0.134950	3152.000000
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.626667	0.180975	4548.000000
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.233214	5956.000000
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.507463	8714.000000

From above table we can conclude that data is already normaly distributed

> Model Development
>
> > Divide the data in train and test

```
#import libraries
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
```

```
temp = dataBike
dataBike = temp
```

```
#create categorical variables to dummy variables
dataBike = pd.get_dummies(dataBike,columns = cat_cnames)
```

```
dataBike.shape
```

(731, 25)

```
dataBike.head()
```

	temperature	humidity	windspeed	count	season_1.0	season_2.0	season_3.0	season_4.0	year_0.0	year_1.0	...	month_6.0	month_7.0	month_8.0	mor
0	0.344167	0.805833	0.160446	985.0	1	0	0	0	1	0	...	0	0	0	
1	0.363478	0.696087	0.248539	801.0	1	0	0	0	1	0	...	0	0	0	
2	0.196364	0.437273	0.248309	1349.0	1	0	0	0	1	0	...	0	0	0	
3	0.200000	0.590435	0.160296	1562.0	1	0	0	0	1	0	...	0	0	0	
4	0.226957	0.436957	0.186900	1600.0	1	0	0	0	1	0	...	0	0	0	

5 rows × 25 columns

```
#Calculate MAPE
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
    return mape
```

```
#divide data for predictor and target
x = dataBike.drop(['count'], axis = 1)
y = dataBike['count']
```

```
#divide data into train and test
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size = 0.20, random_state = 0)
```

> Decision Tree

```
#import library
from sklearn.tree import DecisionTreeRegressor
```

```
#Decision tree for regression
DT_model = DecisionTreeRegressor(max_depth=2).fit(xtrain, ytrain)
```

```
#prediction for train data
DT_train = DT_model.predict(xtrain)
```

```python
#prediction for test data
DT_test = DT_model.predict(xtest)
```

```python
#MAPE
train_MAPE_DT = MAPE(ytrain,DT_train)

#MAPE
test_MAPE_DT = MAPE(ytest,DT_test)

#rsquare
rsquare_train_DT = r2_score(ytrain,DT_train)

#rsquare
rsquare_test_DT = r2_score(ytest,DT_test)
```

```python
print("MAPE for train : " + str(train_MAPE_DT))
print("MAPE for test : " + str(test_MAPE_DT))
print("rsquare for train : " + str(rsquare_train_DT))
print("rsquare for test : " + str(rsquare_test_DT))
```

```
MAPE for train : 62.26013293672567
MAPE for test : 36.94809301452646
rsquare for train : 0.6775629218593628
rsquare for test : 0.6464697716428666
```

```python
data1 = {'Model Name' : ['Decision Tree'], 'MAPE train' : [train_MAPE_DT], 'MAPE Test' : [test_MAPE_DT],
         'rsquare train': [rsquare_train_DT], 'rsquare test': [rsquare_test_DT]}
result1 = pd.DataFrame(data1)
```

Linear Regression

```python
LR_model = sm.OLS(ytrain,xtrain).fit()
LR_model.summary()
```

OLS Regression Results

Dep. Variable:	count	**R-squared:**	0.838
Model:	OLS	**Adj. R-squared:**	0.832
Method:	Least Squares	**F-statistic:**	145.2
Date:	Wed, 14 Aug 2019	**Prob (F-statistic):**	4.07e-207
Time:	12:54:17	**Log-Likelihood:**	-4707.6
No. Observations:	584	**AIC:**	9457.
Df Residuals:	563	**BIC:**	9549.
Df Model:	20		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
temperature	4861.4866	470.566	10.331	0.000	3937.208	5785.765
humidity	-2046.1090	349.646	-5.852	0.000	-2732.879	-1359.339
windspeed	-3183.7171	471.041	-6.759	0.000	-4108.929	-2258.506

```python
# make the predictions by the model on train data
LR_train = LR_model.predict(xtrain)

# make the predictions by the model on test data
LR_test = LR_model.predict(xtest)
```

```python
#MAPE
train_MAPE_LR = MAPE(ytrain,LR_train)

#MAPE
test_MAPE_LR = MAPE(ytest,LR_test)

#rsquare
rsquare_train_LR = r2_score(ytrain,LR_train)

#rsquare
rsquare_test_LR = r2_score(ytest,LR_test)
```

```python
print("MAPE for train : " + str(train_MAPE_LR))
print("MAPE for test : " + str(test_MAPE_LR))
print("rsquare for train : " + str(rsquare_train_LR))
print("rsquare for test : " + str(rsquare_test_LR))
```

```
MAPE for train : 43.78140724487469
MAPE for test : 20.04223329543141
rsquare for train : 0.8376458444602071
rsquare for test : 0.838132097806852
```

```python
data2 = {'Model Name' : ['Linear Regression'], 'MAPE train' : [train_MAPE_LR], 'MAPE Test' : [test_MAPE_LR],
         'rsquare train': [rsquare_train_LR], 'rsquare test': [rsquare_test_LR]}
result2 = pd.DataFrame(data2)
```

Random Forest

```python
#import library for Random Forest
from sklearn.ensemble import RandomForestRegressor
```

```python
RF_model = RandomForestRegressor(n_estimators = 100).fit(xtrain, ytrain)
```

```python
# make the predictions by the model on train data
RF_train = RF_model.predict(xtrain)

# make the predictions by the model on test data
RF_test = RF_model.predict(xtest)
```

```python
#MAPE
train_MAPE_FR = MAPE(ytrain,RF_train)

#MAPE
test_MAPE_FR = MAPE(ytest,RF_test)

#rsquare
rsquare_train_FR = r2_score(ytrain,RF_train)

#rsquare
rsquare_test_FR = r2_score(ytest,RF_test)
```

```
print("MAPE for train : " + str(train_MAPE_FR))
print("MAPE for test : " + str(test_MAPE_FR))
print("rsquare for train : " + str(rsquare_train_FR))
print("rsquare for test : " + str(rsquare_test_FR))
```

```
MAPE for train : 15.682924468937266
MAPE for test : 19.827128049671185
rsquare for train : 0.9812821613359733
rsquare for test : 0.8884676686296503
```

```
data3 = {'Model Name' : ['Random Forest'], 'MAPE train' : [train_MAPE_FR], 'MAPE Test' : [test_MAPE_FR],
         'rsquare train': [rsquare_train_FR], 'rsquare test': [rsquare_test_FR]}
result3 = pd.DataFrame(data3)
```

```
result = result.append(result3)
```

```
result = result.reset_index(drop = True)
```

```
result
```

	Model Name	MAPE train	MAPE Test	rsquare train	rsquare test
0	Decision Tree	62.260133	36.948093	0.677563	0.646470
1	Linear Regression	43.781407	20.042233	0.837646	0.838132
2	Random Forest	15.682924	19.827128	0.981282	0.888468

```
Thank you
```

5. References

For data clean and modeling

https://edwisor.com/career-data-science

https://www.geeksforgeeks.org/python-how-and-where-to-apply-feature-scaling/

For Visualization

http://www.sthda.com/english/wiki/ggplot2-histogram-plot-quick-start-guide-r-software-and-data-visualization

https://www.guru99.com/r-scatter-plot-ggplot2.html#2

http://www.sthda.com/english/wiki/ggplot2-scatter-plots-quick-start-guide-r-software-and-data-visualization