Predicting Bike Rental Count

Akash Siripuram

29-05-2019

Contents

Chapter 1

Introduction

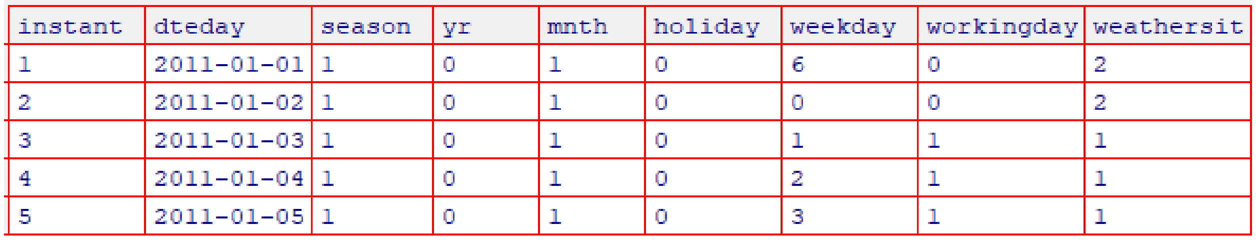
1.1 Problem Statement

There is a Bike Rental Shop. The count of
Number of Bikes are given and their count
with respect to season, weather, holidays, etc. Are stored in the form of Data. So, the problem is that they wanted us to predict the count based on the different factors as mentioned above. They need a model so that whenever the new data comes they will apply the model that we provide and get the prediction results.

1.2 Data

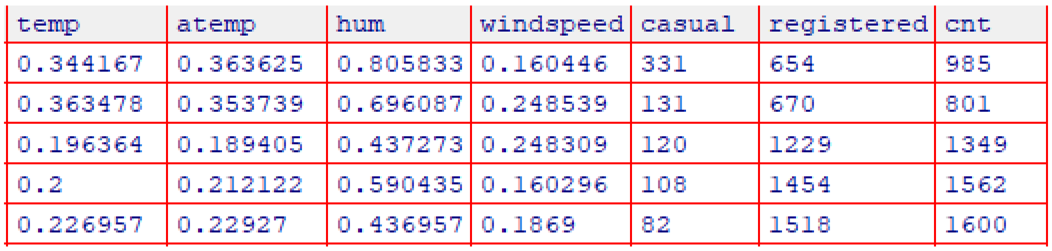
Our task is to build a regression model that will predict the bike rental count based on the historical data And predict for the future data. Below is the sample data mentioned

Table for Columns (1-9)



instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	2011-01-01	1	0	1	0	6	0	2
2	2011-01-02	1	0	1	0	0	0	2
3	2011-01-03	1	0	1	0	1	1	1
4	2011-01-04	1	0	1	0	2	1	1
5	2011-01-05	1	0	1	0	3	1	1

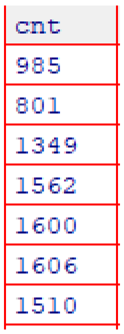
Table for Columns (10-16)



temp	atemp	hum	windspeed	casual	registered	cnt
0.344167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349
0.2	0.212122	0.590435	0.160296	108	1454	1562
0.226957	0.22927	0.436957	0.1869	82	1518	1600

Dependent Variable:

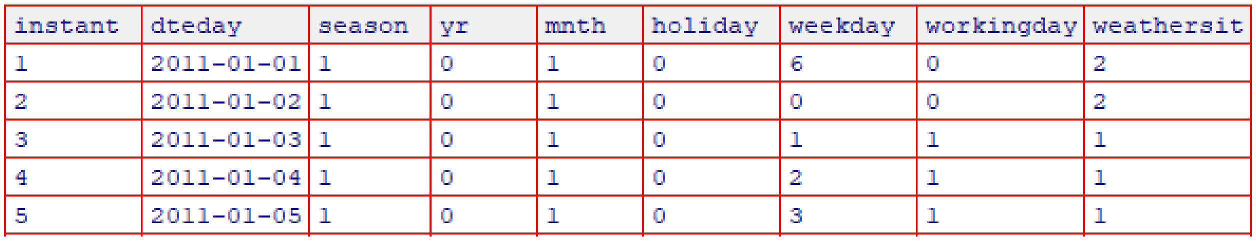
It is nothing but the variable that we what to predict or classify. Here as we are predicting the cnt Variable. The Cnt variable will become our Dependent Variable. There will be only one dependent Variable for whole data.



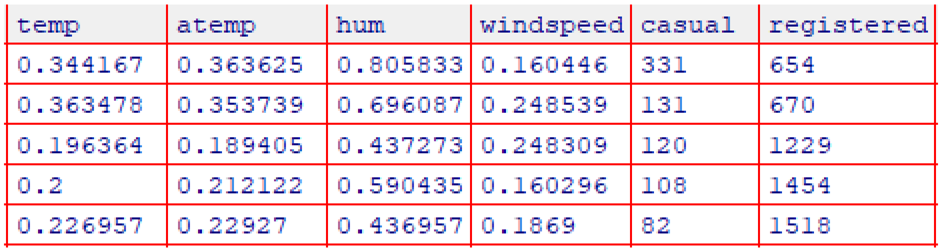
cnt
985
801
1349
1562
1600
1606
1510

Independent Variables:

The variables that helps in predicting the dependent variables are known as independent variables. They carry information that helps in predicting the dependent variables.There can be more than 1 independent variables in the data.



instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	2011-01-01	1	0	1	0	6	0	2
2	2011-01-02	1	0	1	0	0	0	2
3	2011-01-03	1	0	1	0	1	1	1
4	2011-01-04	1	0	1	0	2	1	1
5	2011-01-05	1	0	1	0	3	1	1



temp	atemp	hum	windspeed	casual	registered
0.344167	0.363625	0.805833	0.160446	331	654
0.363478	0.353739	0.696087	0.248539	131	670
0.196364	0.189405	0.437273	0.248309	120	1229
0.2	0.212122	0.590435	0.160296	108	1454
0.226957	0.22927	0.436957	0.1869	82	1518

Let’s see the type of variables we have :

Categorical Variables:

dteday: Date season: Season (1:springer, 2:summer, 3:fall, 4:winter)

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

mnth: Month (1 to 12) hr: Hour (0 to 23) holiday: weather day is holiday or not

weekday: Day of the week working day: If day is neither weekend nor holiday is 1, otherwise is 0.

weathersit:

1: Clear, Few clouds, Partly cloudy, Partly cloudy 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Let’s see the continuous variables:

Continuous Variables:

temp: Normalized temperature in Celsius.

atemp: Normalized feeling temperature in Celsius.

hum: Normalized humidity.

windspeed: Normalized wind speed.

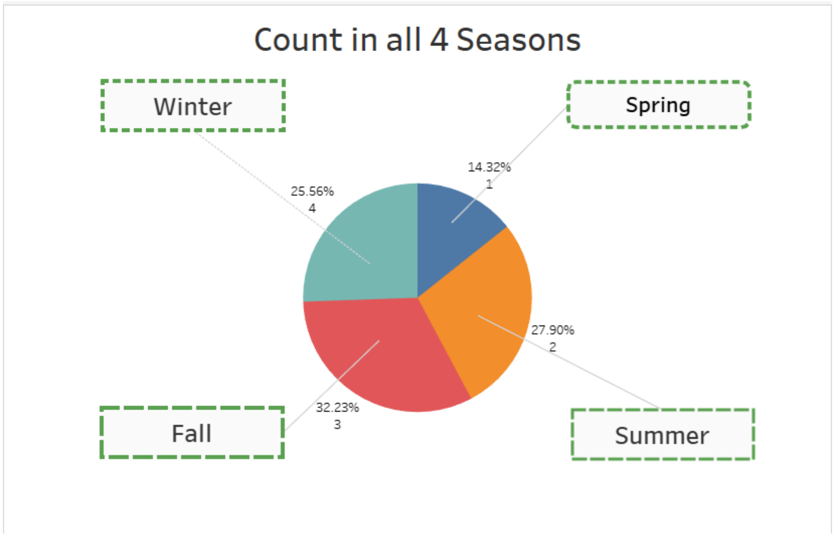
casual: count of casual users registered: count of registered users

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

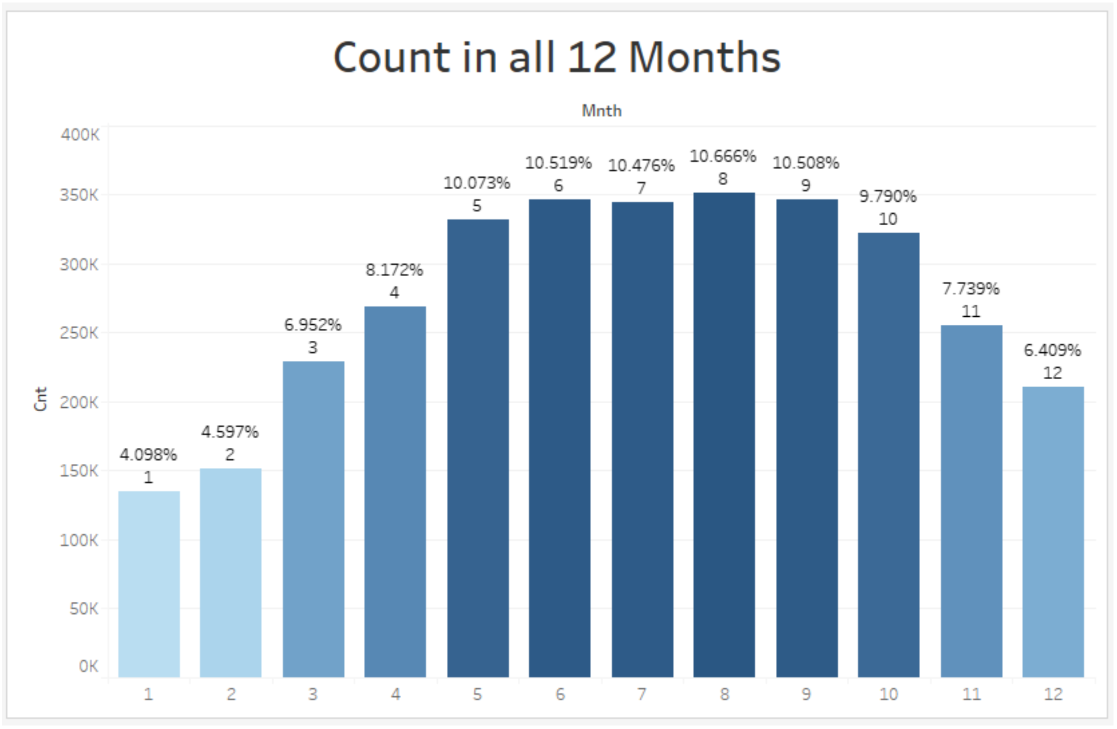
Instant: Record index

Let’s also see some visualizations of the categorical variables w.r.t dependent variable:

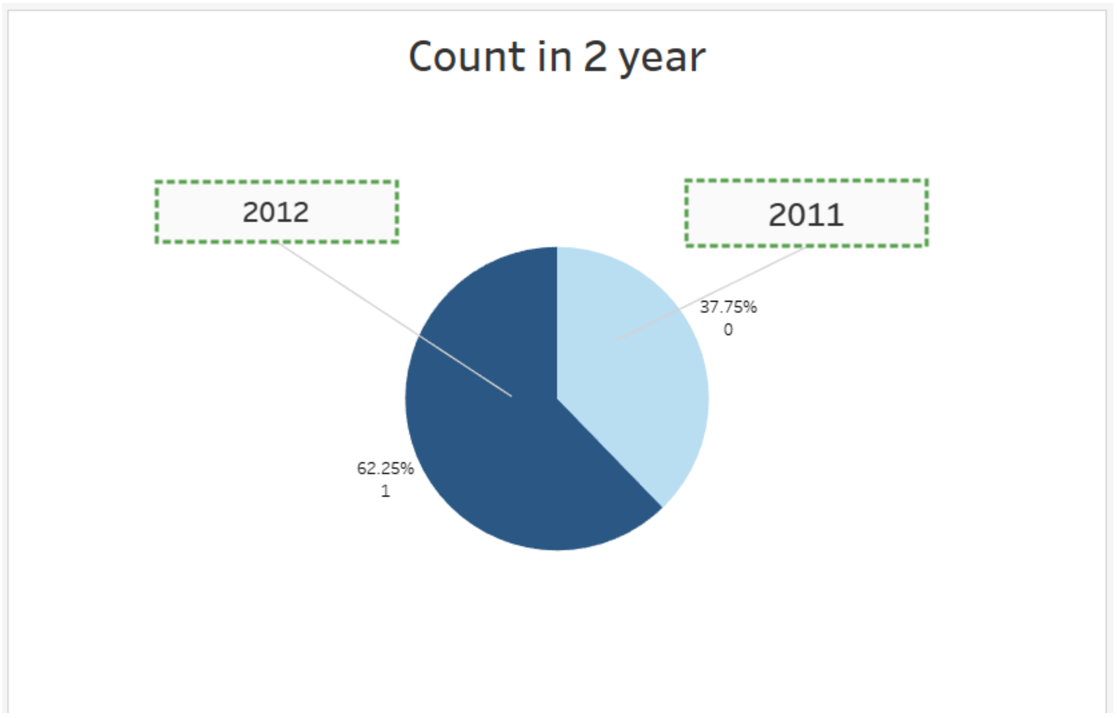
Count w.r.t Seasons:



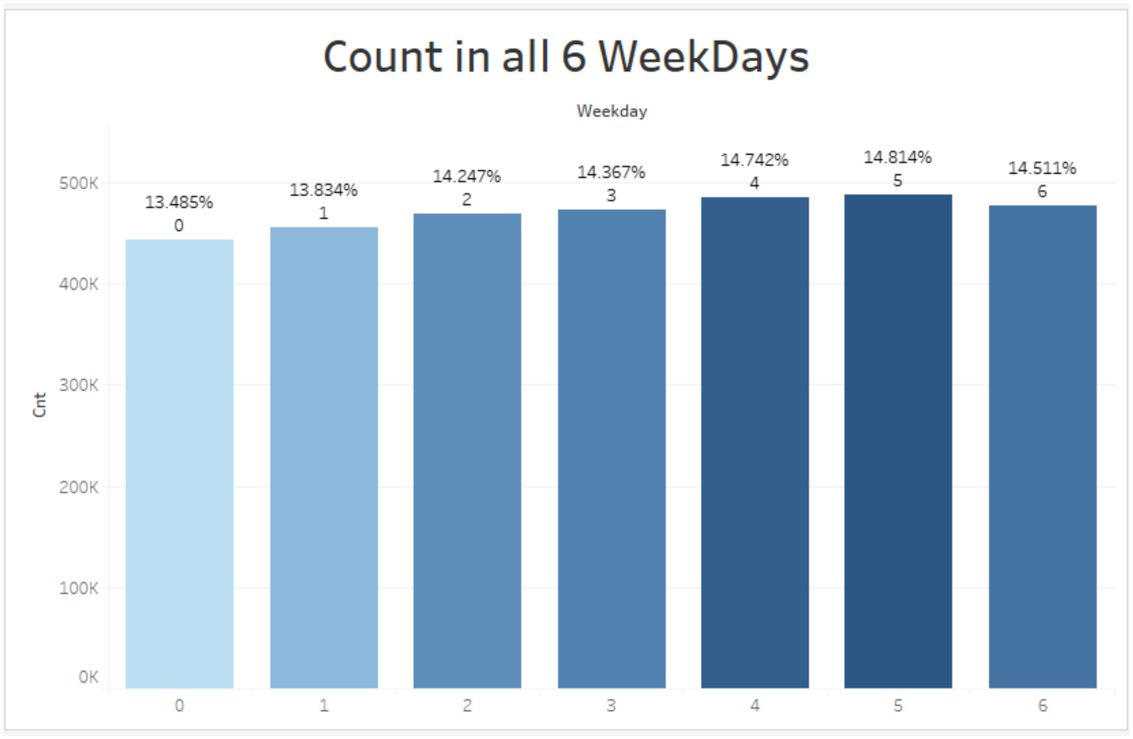
Count w.r.t 12 Months:



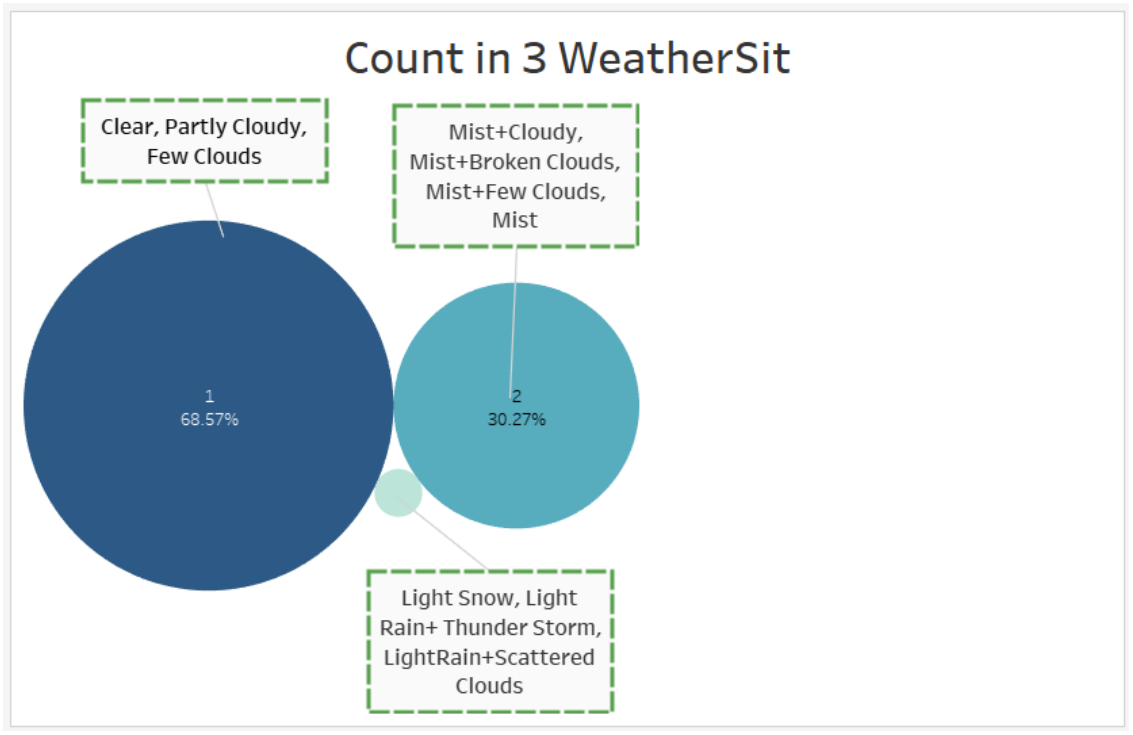
Count w.r.t Year:



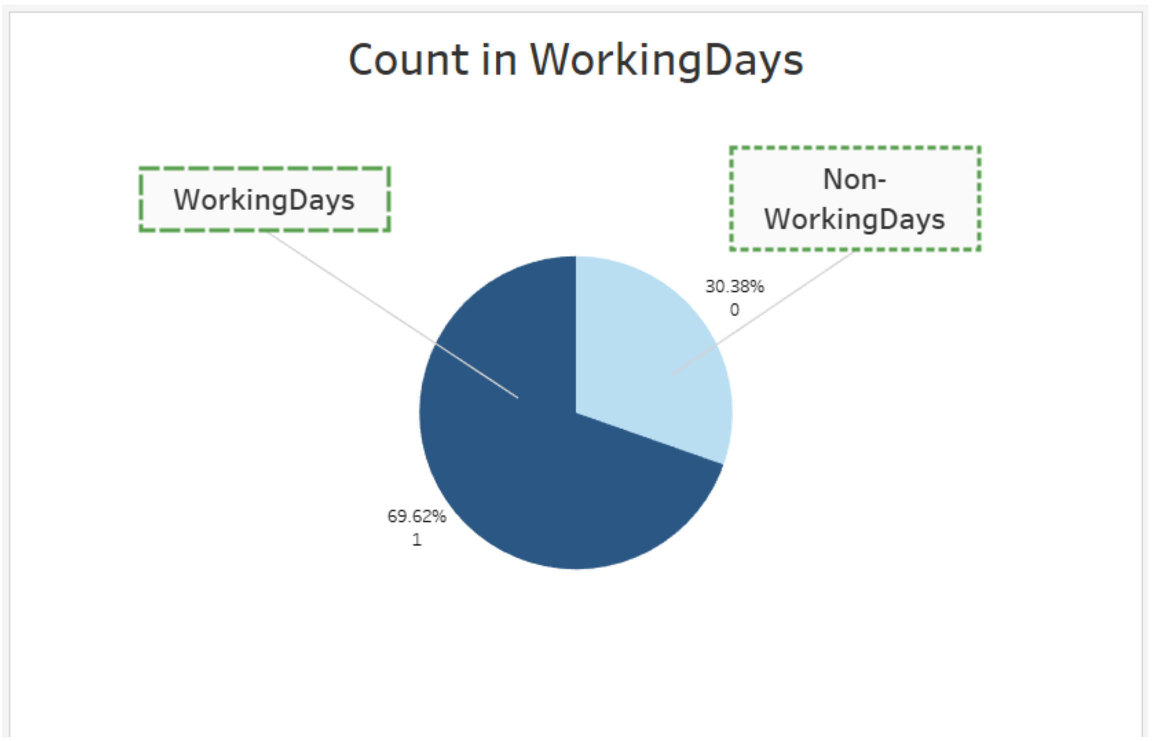
Count w.r.t Week Days:



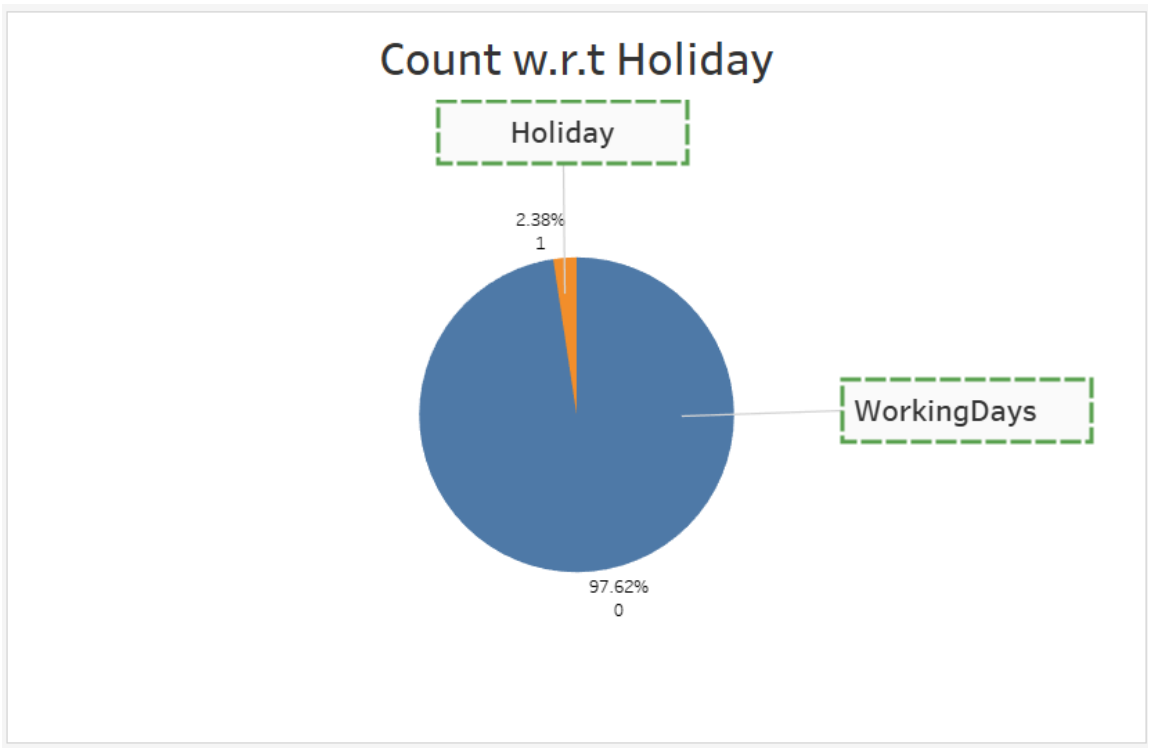
Count w.r.t Weather Sit:



Count w.r.t Working Days:



Count w.r.t Holiday:



Chapter 2

Methodology

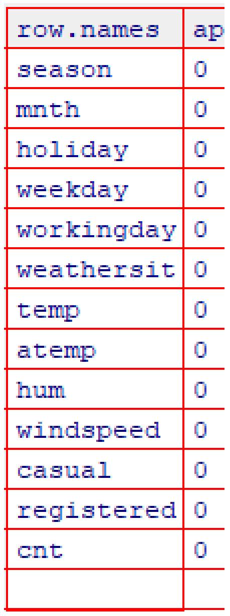
2.1 Pre Processing

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

Now, here we have done Missing value analysis, Outlier Analysis, Feature Selection and Feature Scaling. We have also checked them using the Visualizations. I have done visualizations in R, Python, Tableau. We will be showing each and every step one after another so, that you can understand one after another.

2.1.1 Missing Value Analysis:

Here we will be checking for the missing values i.e. Na Values so that it won't affect our model.
We found that we don't have any of the missing values.



row.names	ap
season	0
mnth	0
holiday	0
weekday	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	0
casual	0
registered	0
cnt	0

2.1.2 Outlier Analysis:

Actually, why we do outlier analysis is that to find the outliers and remove them.
So, what are outliers. In simple words let's take an example:

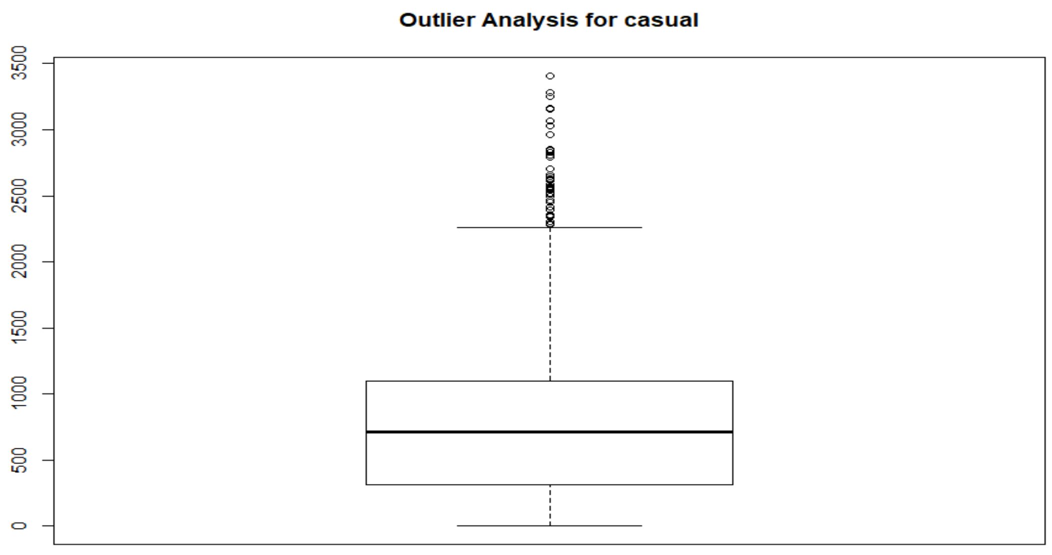
If I have a flower shop and I want to total the average sales in a month. But in the same month on a particular day I have sold many flowers that I didn't sell in other days. So, when counting the average, I will be having a problem because on that particular day I sold many flowers. So, when counting for each normal day average I will have the effect of that particular day on other days. Such the no of flowers sold on that day are known as outliers.

So, we have to remove them so that they won't affect our model.So, Outlier analysis works only on Continous variables let's check them.

temp	atemp	hum	windspeed	casual	registered	cnt
0.344167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349

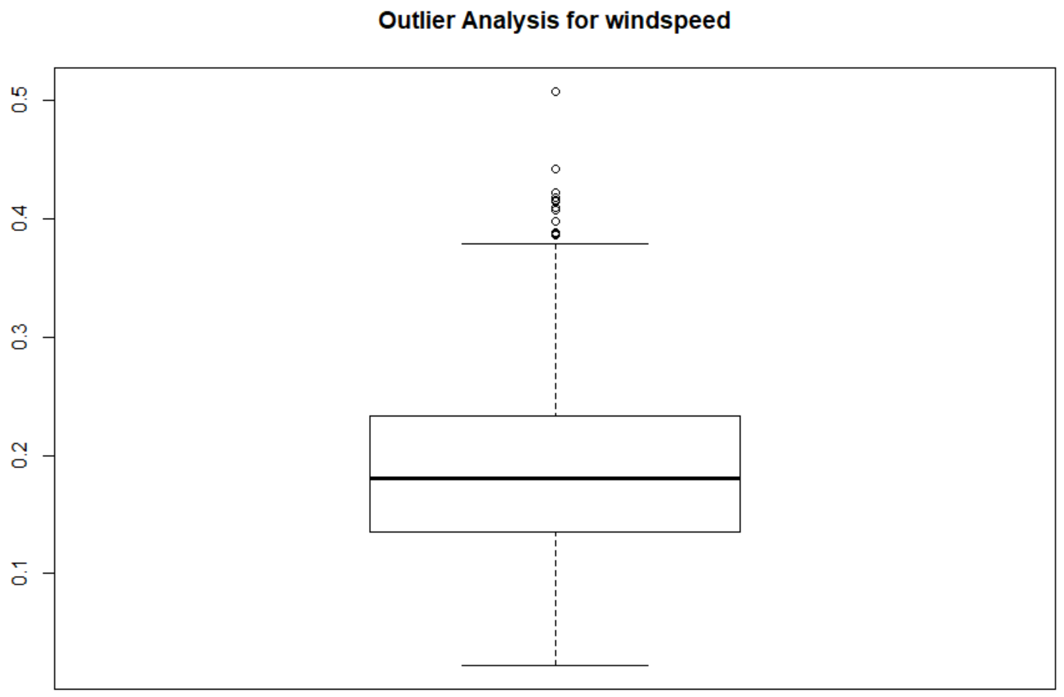
Let's see the outliers for each and every continuous variables:

Boxplot For Casual Variable:

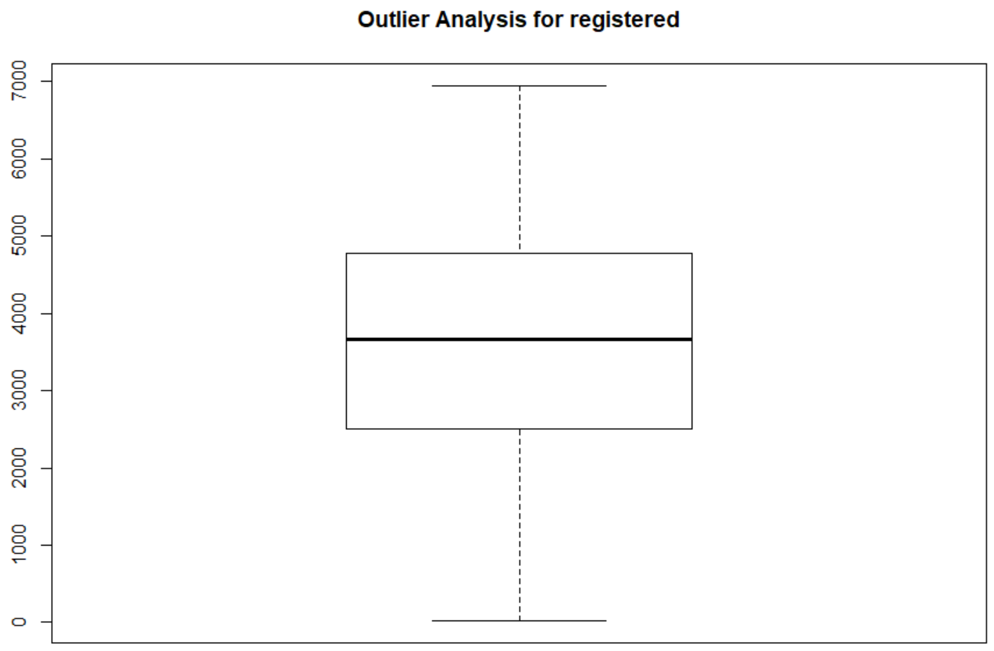


Outlier analysis of Casual variable

Boxplot For Windspeed Variable:

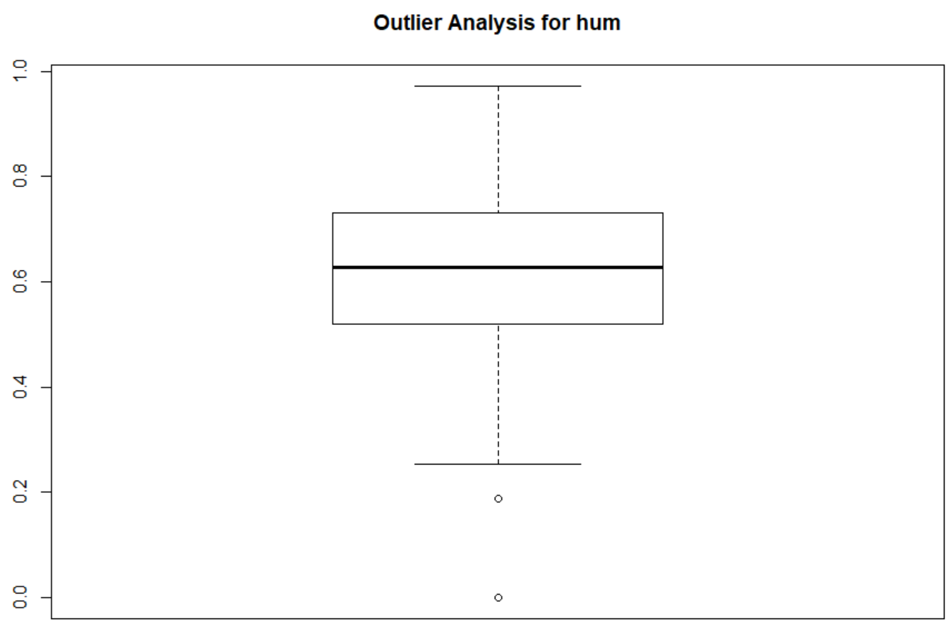


Boxplot For Registered Variable:



Outlier analysis of registered variable

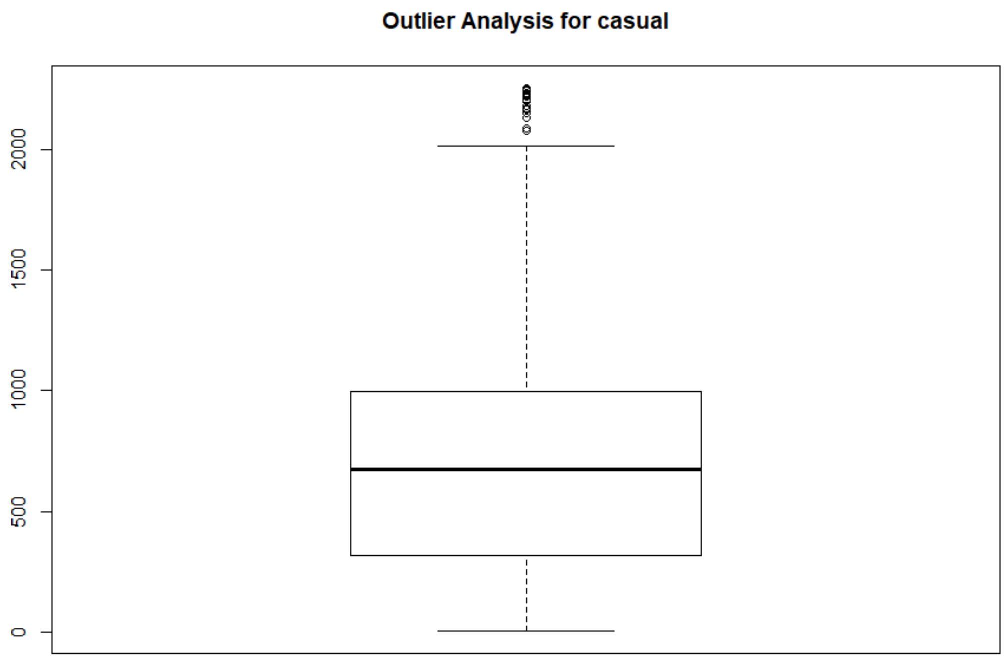
Boxplot For Hum Variable:



Values of hum variable

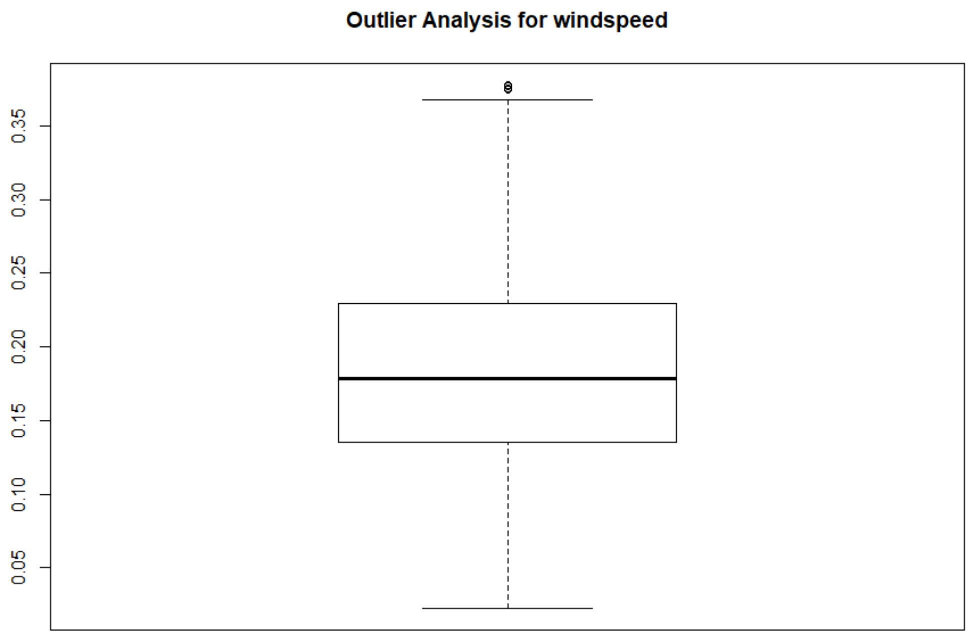
So, After Performing Outlier analysis :

Boxplot For Casual Variable:



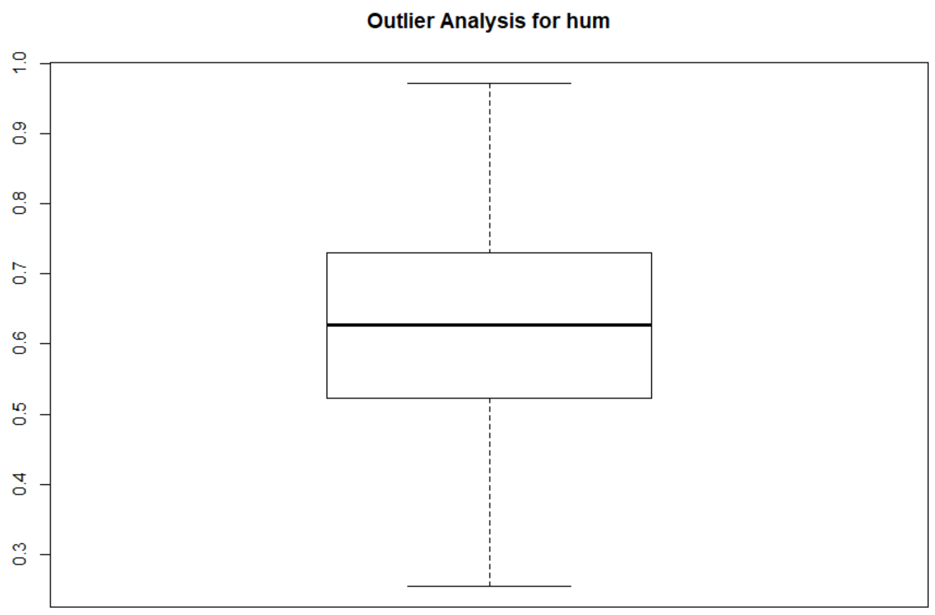
Outlier analysis of Casual variable

Boxplot For Windspeed Variable:



Outlier analysis of windspeed variable

Boxplot For Hum Variable:



Outlier analysis of hum variable

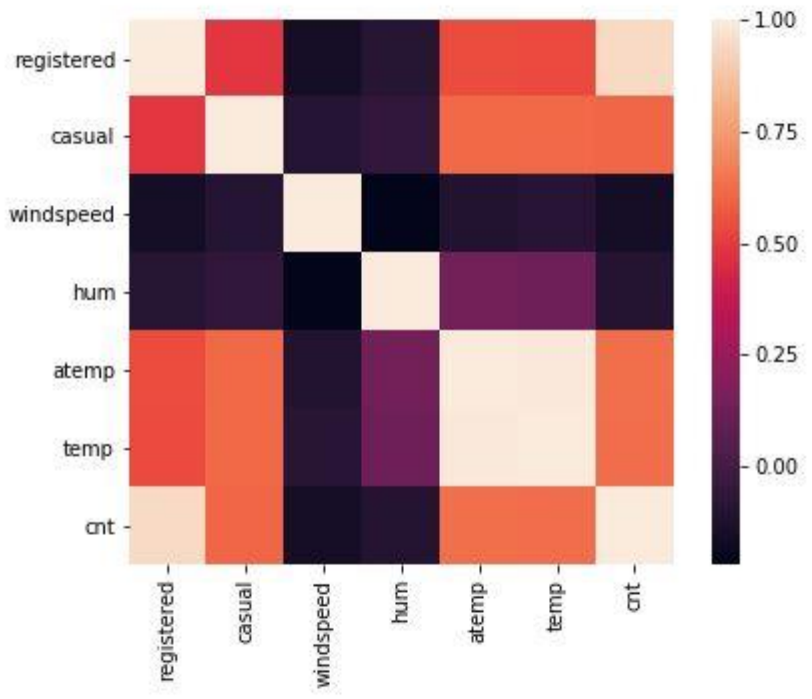
When we have seen the boxplots of continuous variables, we found that the outliers are present in variables named Casual, Windspeed, Hum variables.

So, when we have done outlier analysis, we have also shown you the boxplots after outlier analysis. We have used the best methods to reduce the outliers present in that particular variable.

2.1.3 Feature Selection:

Here what we do is that we will be finding the variables that give more information about out dependent variable. If any variable is not giving meaning then, it can be removed so that it won't decrease the model's accuracy.

We have done it by using correlation graph:



Here above we can see the variables, Hum and Windspeed are really affecting the cnt variable. So, we have to remove them so that we can increase our model's accuracy.

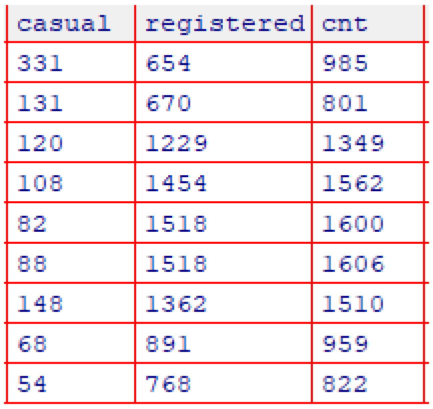
2.1.4 Feature Scaling:

Why do we do this?

We have the data which is of different ranges or units. So, when we wanted to predict the model. This really causes a huge problem to the model so, we will convert all the values in between the range of 0 and 1. We will be using Normalization to do this instead of Standardization.

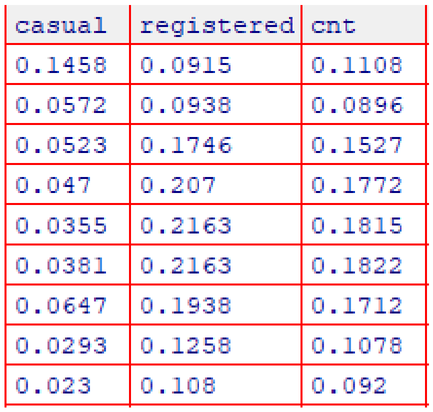
Let’s see with an example:

Before Feature Scaling:



casual	registered	cnt
331	654	985
131	670	801
120	1229	1349
108	1454	1562
82	1518	1600
88	1518	1606
148	1362	1510
68	891	959
54	768	822

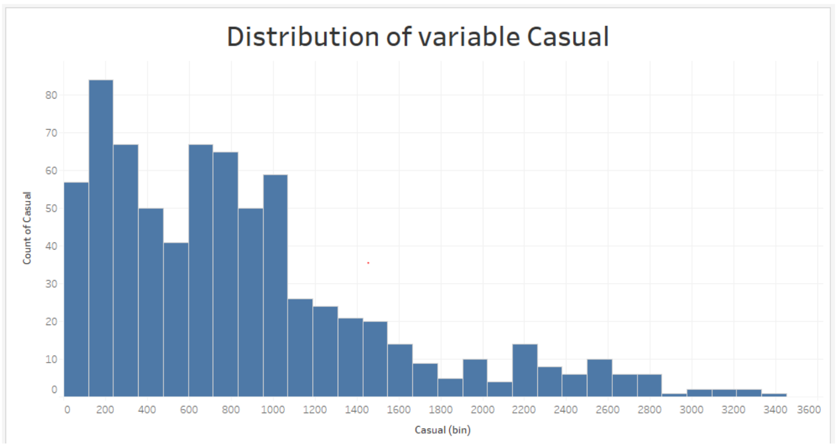
After Feature Scaling:



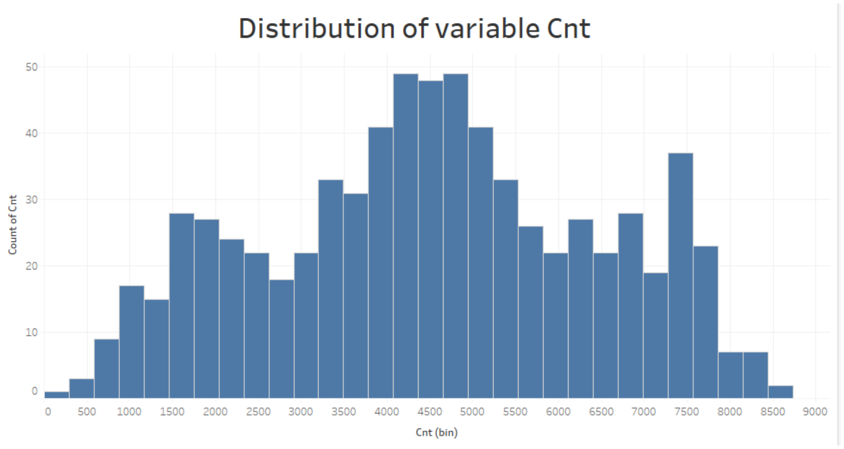
casual	registered	cnt
0.1458	0.0915	0.1108
0.0572	0.0938	0.0896
0.0523	0.1746	0.1527
0.047	0.207	0.1772
0.0355	0.2163	0.1815
0.0381	0.2163	0.1822
0.0647	0.1938	0.1712
0.0293	0.1258	0.1078
0.023	0.108	0.092

Let’s visualize the distributions of each and every variable:

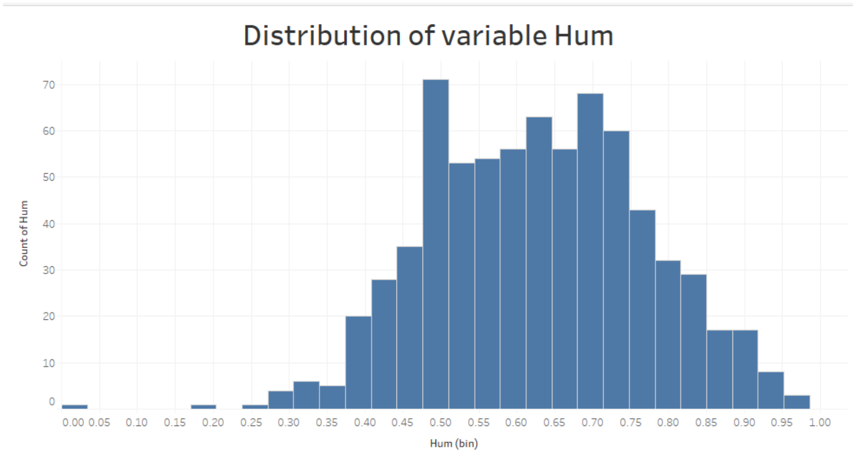
Distribution for Casual Variable:



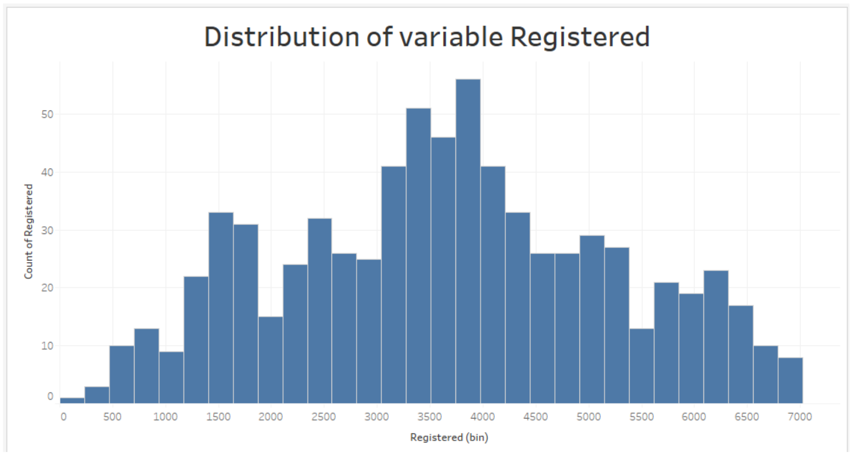
Distribution for Cnt Variable:



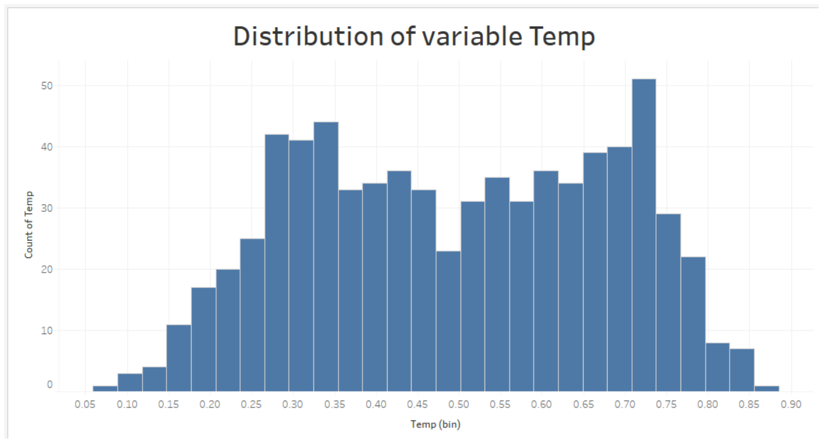
Distribution for Hum Variable:



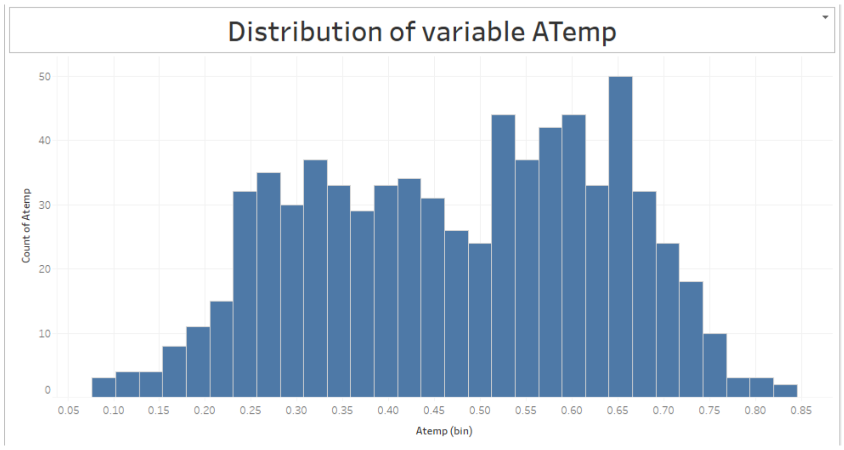
Distribution for Registered Variable:



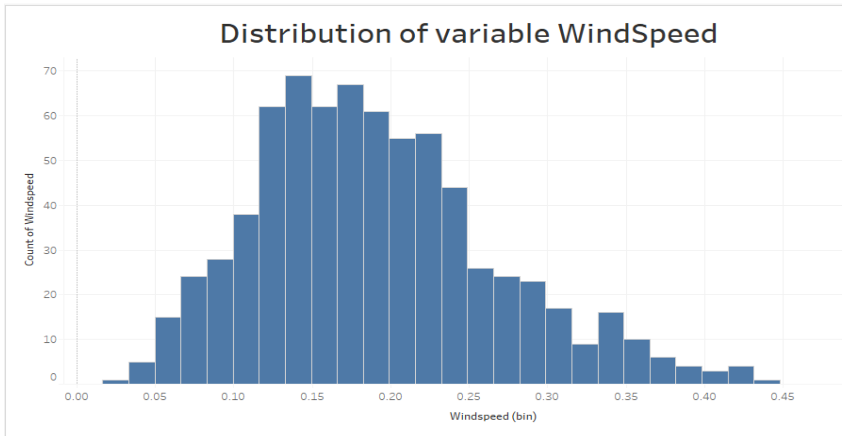
Distribution for Temp Variable:



Distribution for Atemp Variable:



Distribution for Windspeed Variable:



2.2 Modeling

2.2.1 Model Selection

We have reached here We have done with Exploratory data analysis and we a data that will give us the good accuracy model.

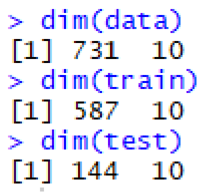
So, to develop a model we should be having two datasets named train and test.

We train our model with the train dataset and test that model for prediction or classification on test dataset.

So, in the whole project we should not touch even the test data after doing Sampling.

Sampling is a technique that helps to take a sample of data from the whole data by considering all the characteristics of the data.

So, we have our train and test data now.



```
> dim(data)
[1] 731  10
> dim(train)
[1] 587  10
> dim(test)
[1] 144  10
```

We have the train data of 80% and test data of 20%.
So, 587 is 80% of Actual data and 144 is the 20% of Actual data
We don't remove or change the variables during Sampling because we will be needing all the Actual dataset variables in both train and test datasets.

So, now how to do the training on the train dataset.
We have many different MLA's(Machine Learning Algorithms) as well as Statistical models that helps in Regression as well as Classification.

Some of the Regression Algorithms are:

1.Decision Tree
2.Random Forest
3.Linear Regression, etc.

Some of the Classification Algorithms are:

1.Decisison Tree
2.Random Forest
3.Logistic Regression.
4.SVM, etc.

So, as we are doing prediction we should be using regression algorithms.

We applied all the three algorithms on the Train datasets.

We found that Linear regression is providing the best results with an accuracy of 80% and error of 20%.

The above one is the data without feature scaling. So, for that data we got the accuracy of 80% using Linear regression.

But, when we have done feature scaling on the data in R, we found that when the train dataset is trained by three regression algorithms, we found that random forest is providing the good accuracy of the data.

But when we have done the same feature Scaling on the data in Python, we found that Linear regression is giving an accuracy of 82%.

So, we assume that Linear regression is working well on the train data in predicting the Count based on historical data.
So, we can use linear regression for predicting the future data.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models.

But, how did we measure the accuracy of the models?

We will be having some error metrics so with the help of them we are going to find the best model.
So, for classification we have different error metrics and for regression we have different error metrics.

For Classification:

1. Confusion Matrix
2. Accuracy
3. Misclassification Error
4. Specificity
5. Recall

   And many more.

For Regression:

1. MSE (Mean Squared Error)
2. RMSE (Root Mean Squared Error)
3. MAPE (Mean Absolute Percentage Error)
4. MAE (Mean Absolute Error)

   And many more.

So, as we are doing regression, we have to use regression metrics.
I have used MAPE which gives us the error percentage and we have to calculate the accuracy like this:

If I got 0.15235 as MAPE

So, my MAPE is 15%
And my accuracy is 100-15=85%

So, I have calculated in this manner and selected the best model.
We found that linear regression is providing the best results.

So, For Decision Tree MAPE gave 31%

Accuracy of this model will be 100-31= 69%

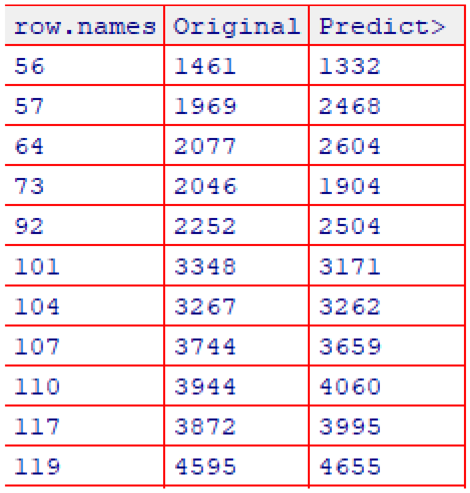
So, For Linear Regression MAPE gave 18%

Accuracy of this model will be 100-18= 82%

So, For Random Forest MAPE gave 36%

Accuracy of this model will be 100-36= 64%

So, by this we have also selected the best model that is Linear Regression and the Actual data and predicted data are provided side by side.



row.names	Original	Predict>
56	1461	1332
57	1969	2468
64	2077	2604
73	2046	1904
92	2252	2504
101	3348	3171
104	3267	3262
107	3744	3659
110	3944	4060
117	3872	3995
119	4595	4655

R code without Feature Scaling:

```
#-----------------------------------------------------------------------------------------------
#-----------------------------------------------------------------------------------------------
library(ggplot2)
library(corrgram)
library(caret)
library(rpart)
library(MASS)
#Reading the dataset
data=read.csv(file="BikeRental.csv",header=T)
#-----------------------------------------------------------------------------------------------
#Creating Copy so that we can use it anywhere in our program
data_c=data
#Dropping two variables instant and dteday
data=data[,-c(1,2,4,6)]
#-----------------------------------------------------------------------------------------------
head(data)
#checking the column names
colnames(data)
#checking the datatypes of each variable
str(data)
head(data,5)
#-----------------------------------------------------------------------------------------------
#Here Above we have found that the first 7 variables are of integer datatype for
#catagorical variable as it should be a factor for catogorical datatype

#converting the datatype
for(i in c('season','mnth','weekday','workingday','weathersit')){

  data[,i]=as.factor(data[,i])

}

#checking the datatypes
str(data)
#-----------------------------------------------------------------------------------------------
# Missing value analysis


missing_val=data.frame(apply(data,2,function(x){sum(is.na(x))}))
missing_val

#As we don't have any of the  missing values we can proceed

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

# Outlier analysis


#taking the numerical indexes
num_index=sapply(data,is.numeric)
```

```
#taking the numerical data based on numerical indexes taken before
num_data=data[,num_index]
#checking numerical data
num_data
#getting all the numerical data columns
cnames=colnames(num_data)
#-----------------------------------------------------------------------------------------
# Visualizing the Outliers using boxplots

boxplot(data['casual'], data=data, xlab="Values of casual variable",main="Outlier Analysis for Casual")
boxplot(data['windspeed'], data=data, xlab="Values of windspeed variable",main="Outlier Analysis for
windspeed")
boxplot(data['hum'], data=data, xlab="Values of hum variable",main="Outlier Analysis for hum")
boxplot(data['registered'], data=data, xlab="Outlier analysis of registered variable",main="Outlier Analysis
for registered")
#-----------------------------------------------------------------------------------------
#here we found that we are having outliers in variables like 'casual','windspeed','hum'
#Now performing outlier analysis on theses variables

for (i in cnames){
  print(i)
  val=data[,i][data[,i]%in%boxplot.stats(data[,i])$out]
  data[,i][data[,i]%in% val]=NA
}
#-----------------------------------------------------------------------------------------
#checking the number of NA values present

sum(is.na(data))

# Now Applying missing value analysis to replace NA values that are present

data$windspeed[is.na(data$windspeed)]=median(data$windspeed,na.rm=T)
data$casual[is.na(data$casual)]=median(data$casual,na.rm=T)
data$hum[is.na(data$hum)]=median(data$hum,na.rm=T)

#checking the no of outliers after outlier analysis,But let's visualize the variables to find the outlier's

sum(is.na(data))

#as it is 0 we proceed to Feature selection

#visualizing the variables again after outlier analysis

boxplot(data['casual'], data=data, xlab="Outlier analysis of Casual variable",main="Outlier Analysis for
casual")
boxplot(data['windspeed'], data=data, xlab="Outlier analysis of windspeed variable",main="Outlier
Analysis for windspeed")
boxplot(data['hum'], data=data, xlab="Outlier analysis of hum variable",main="Outlier Analysis for hum")

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

#Feature selection :

#correlation:
head(data)
num_data=data[,-c(1:5)]
```

```
corrgram(data,order=F,upper.panel=panel.pie,text.panel =panel.txt,main="Correlation plot")


#here windspeed and hum are not providing much information about the dependent variable cnt,
#so remove them

#Removing variables windspeed and hum
data=data[,-c(8,9)]

#Checking wheather is it removed or not
head(data)

#checking the columns that we have..
colnames(data)

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

#Sampling:

#Sampling: We are splitting data into train and test So, that we can use test dataset
# for prediction or classification.

set.seed(321)
indexTrain = createDataPartition(data$cnt, p = .80, list = FALSE)
train = data[ indexTrain,]
test  = data[-indexTrain,]

#Checking dimensions of actual dataset, Train and test dataset.

dim(data)
dim(train)
dim(test)

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

#Now using Machine Learning algorithms to predict the values :


#Using decision tree for regression

f=rpart(cnt~ .,data=train,method="anova")
fpredictions=predict(f,test[,-10])

#calculating mape

mape=function(y,yhat){
  mean(abs((y-yhat)/y))
}
mape(as.integer(test[,10]),as.integer(fpredictions))

#mape = 52.63 %
#accuracy = 47.36 %

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

#using linear regression for regression
```

```
lm_model=lm(cnt~.,data = train)

#predicting

p_lm=predict(lm_model,test[,1:9])

#Calculating mape

mape(test[,10],p_lm)

#Error Percentage = 20 %
#Accuracy = 80%

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

#Using randomforest for regression
head(test)
head(predicted)
x <- train
# Fitting model
fit <- randomForest(train$cnt ~ .,x,ntree=500)
summary(fit)
#Predict Output
predicted= predict(fit,test)

#Calculating mape

mape(test[,10],predicted)

#error percentage 56%
#accuracy 44%
#-----------------------------------------------------------------------------------------------

#So, when we compare All the Algorithms we found that linear regression is
#providing the best incase of accuracy and error percentage..


#comparing the results side by side, It is only 80% accurate.

original=test[,10]
original
#we are column binding
Combined=cbind(original,round(p_lm,0))
Combined_data=as.data.frame(Combined)
names(Combined_data)=c('Original','Predicted')
head(Combined_data,20)


# We found that linear regression is providing the best results.

# So, we can apply linear regression on the future data i.e bike Rental prediction Project.

#-----------------------------------------------------------------------------------------------
```

R code with Feature Scaling:

```
# With feature Scaling  :

#Here we are having,The data which is i 0 to 1 range, MAPE(mean absolute percentage error)
# is not working so, we are using rmse to find the best suitable model.
#-----------------------------------------------------------------------------------------------

rm(list=ls())

library(ggplot2)
library(corrgram)
library(randomForest)
library(caret)
library(rpart)
library(MASS)
#Reading the dataset
data=read.csv(file="BikeRental.csv",header=T)

#Creating Copy so that we can use it anywhere in our program
data_c=data
#-----------------------------------------------------------------------------------------------
#Dropping two variables instant and dteday and
data=data[,-c(1,2,4,6)]
head(data)
#checking the column names
colnames(data)
#checking the datatypes of each variable
str(data)
head(data,5)

#here we have found that the first 7 variables are of integer datatype for
#catagorical variable as it shouldbe a factor for catogorical datatype


for(i in c('season','mnth','weekday','workingday','weathersit')){

  data[,i]=as.factor(data[,i])

}

#checking the datatypes
str(data)
#-----------------------------------------------------------------------------------------------

# Missing value analysis


missing_val=data.frame(apply(data,2,function(x){sum(is.na(x))}))
missing_val

#As we don't have any of the  missing values we can proceed

#-----------------------------------------------------------------------------------------------
```

```
# Outlier analysis

#taking the numerical indexes
num_index=sapply(data,is.numeric)
#taking the numerical data based on numerical indexes taken before
num_data=data[,num_index]
#checking numerical data
num_data
#getting all the numerical data columns
cnames=colnames(num_data)
#------------------------------------------------------------------------------------------------

# Visualizing the Outliers using boxplots

boxplot(data['casual'], data=data, xlab="Values of casual variable",main="Outlier Analysis for Casual")
boxplot(data['windspeed'], data=data, xlab="Values of windspeed variable",main="Outlier Analysis for
windspeed")
boxplot(data['hum'], data=data, xlab="Values of hum variable",main="Outlier Analysis for hum")

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

#here we found that we are having outliers in variables like 'casual','windspeed','hum',holiday
#Now performing outlier analysis on theses variables

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

#checking the number of NA values present

sum(is.na(data))
#------------------------------------------------------------------------------------------------

# Now Applying missing value analysis to replace NA values that are present

data$windspeed[is.na(data$windspeed)]=median(data$windspeed,na.rm=T)
data$casual[is.na(data$casual)]=median(data$casual,na.rm=T)
data$hum[is.na(data$hum)]=median(data$hum,na.rm=T)


#checking the no of outliers after outlier analysis, as it is 0 we proceed to Feature selection
sum(is.na(data))

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


#visualizing the variables again after outlier analysis

boxplot(data['casual'], data=data, xlab="Outlier analysis of Casual variable",main="Outlier Analysis for
casual")
boxplot(data['windspeed'], data=data, xlab="Outlier analysis of windspeed variable",main="Outlier
Analysis for windspeed")
```

```
boxplot(data['hum'], data=data, xlab="Outlier analysis of hum variable",main="Outlier Analysis for hum")#-
------------------------------------------------------------------------------------------

#Feature selection :

#correlation:
head(data)
num_data=data[,-c(1:5)]
corrgram(num_data,order=F,upper.panel=panel.pie,text.panel =panel.txt,main="Correlation plot")

#-------------------------------------------------------------------------------------------
#here windspeed and hum are not providing much information about the dependent variable cnt,
#so remove them

#Removing variables windspeed and hum
data=data[,-c(8,9)]
head(data)

#checking the columns that we have..
colnames(data)

#Checking the values of the data on which feature Scaling to be applied
head(data,5)

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

#Feature Scaling

#Here as our data is of different ranges we are making the data to be in a particular range

#Checking the distribution
hist(data$casual)
hist(data$registered)
hist(data$cnt)

str(data)

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

#As we don't have normally distributed data we are going to use normalization
#Here we are using the normalization formula to set the data in a range
for(i in c(8,9,10)){
  data[,i]=formatC((data[,i]-min(data[,i]))/(max(data[,i])-min(data[,i])), digits = 4, format = "f")
}
#-------------------------------------------------------------------------------------------
#Checking the datatypes of variables
str(data)

#So we found that casual,registered,cnt are having character datatype
#So, We are Changing the datatype from character to numeric

for(i in c(8,9,10)){

  data[,i]=as.numeric(data[,i])

}
```

```
#Now again checking the datatypes of the variables
str(data)

# We found that that they became numerical..So proceed to next step..

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

#Sampling

#Sampling: We are splitting data into train and test So, that we can use test dataset
# for prediction or classification.

set.seed(321)
indexTrain = createDataPartition(data$cnt, p = .80, list = FALSE)
train = data[ indexTrain,]
test  = data[-indexTrain,]

#Checking actual dataset, Train and test datasets

dim(data)
dim(train)
dim(test)
head(train)

str(data)
#-------------------------------------------------------------------------------------------

#Now using Machine Learning algorithms to predict the values :

#using decision tree for regression

f=rpart(cnt~ .,data=train,method="anova")
rpredictions=predict(f,test[,-10])

#calculating rmse
#The least the rmse value is, the greater the model is.


rmse = function(m, o){
  print(sqrt(mean((m - o)^2)))
}
rmse(test[,10],rpredictions)
# Value we got is  = 0.07886405
# when we have aplied root to above No   0.28082743
#-------------------------------------------------------------------------------------------

# Using linear regression for regression

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

#predicting

p_lr=predict(lm_model,test[,1:9])

#calculating rmse
```

```
rmse = function(m, o){
  print(sqrt(mean((m - o)^2)))
}
rmse(test[,10],p_lr)
#rmse value is = 0.04293188

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

#Using randomforest for regression

x <- train
# Fitting model
fit <- randomForest(train$cnt ~ .,x,ntree=500)
summary(fit)

#Predict Output

rfpredicted= predict(fit,test)

#calculating rmse


rmse = function(m, o){
  print(sqrt(mean((m - o)^2)))
}
rmse(test[,10],rfpredicted)

#rmse value is 0.03534393

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


#So, when we compare All the Algorithms we found that Random Forest
#is having the least value.So, we can apply on our future data.


#comparing the results side by side

original=test[,10]
original
#we are column binding
Combined=cbind(original,round(rfpredicted,4))
Combined_data=as.data.frame(Combined)
names(Combined_data)=c('Original','Predicted')
head(Combined_data,20)

# The reason why I have done the with and without feature scaling is because
# When done feature Scaling in python , MAPE Worked and Linear regression gave 81% accuracy But
# In R , MAPE did not work it gave the value "Inf" and gave  less accuracy
# So, I have performed both the methods to show the diference.

# In both the cases i.e 1.With Feature Scaling and 2.Without Feature Scaling

#Random Forest worked well while predicting the Featured Scaled data,
#Linear regression worked well while predicting the Original data.
```

#Depending on the context we have to use the different models to predict the values in the data.

#--
#--

PYTHON CODE

```
#Loading Libraries
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from random import randrange, uniform
import seaborn as sns

#Reading Data and dropping some columns
data=pd.read_csv("BikeRental.csv")
data=data.drop(columns="dteday")
data=data.drop(columns="instant")
data=data.drop(columns="holiday")

#convert the datatypes of variables

data['season'] = data.season.astype("category")
data['yr'] = data.yr.astype("category")
data['mnth'] = data.mnth.astype("category")
data['weekday'] = data.weekday.astype("category")
data['workingday'] = data.workingday.astype("category")
data['weathersit'] = data.weathersit.astype("category")
data['temp'] = data.temp.astype(float)
data['casual'] = data.casual.astype(int)
data['atemp'] = data.atemp.astype(float)
data['hum'] = data.hum.astype(float)
data['windspeed'] = data.windspeed.astype(float)
data['registered'] = data.registered.astype(int)
data['cnt'] = data.cnt.astype(int)

#Checking the data
```

```
data.head(5)
```

	season	yr	mnth	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
0	1	0	1	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
1	1	0	1	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
2	1	0	1	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
3	1	0	1	2	1	1	0.200000	0.212122	0.590435	0.160296	108	1454	1562
4	1	0	1	3	1	1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

In [5]:

```
#missing value analysis

#Here we are checking the count of missing values in each variable

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

In [43]:

```
#Checking the Count
missing_val
```

Out[43]:

	0
season	0
yr	0
mnth	0
weekday	0

	0
workingday	0
weathersit	0
temp	0
atemp	0
hum	0
windspeed	46
casual	73
registered	0
cnt	0

```
#Checking the dimensions of the data
data.shape
```

```
 (731, 13)
```

```
#Outlier Analysis
#Finding outliers using boxplot

plt.boxplot(data["temp"])
#As we have no outliers we proceed to next variable
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1fce12d57f0>,
  <matplotlib.lines.Line2D at 0x1fce12d5b38>],
 'caps': [<matplotlib.lines.Line2D at 0x1fce12d5e80>,
  <matplotlib.lines.Line2D at 0x1fce12d5f60>],
 'boxes': [<matplotlib.lines.Line2D at 0x1fce12d5400>],
 'medians': [<matplotlib.lines.Line2D at 0x1fce12eb550>],
 'fliers': [<matplotlib.lines.Line2D at 0x1fce12eb898>],
 'means': []}
```

```
plt.boxplot(data["atemp"])


#Before removing outliers the windspeed variable is
plt.boxplot(data["windspeed"])

#As we have outliers we have to take note of this variable and proceed to nex
t variable
```

In [11]:

```
plt.boxplot(data['casual'])
#As we have outliers we have to take note of this variable and proceed to nex
t variable
```

In [12]:

```
plt.boxplot(data["temp"])
#As we have no outliers we proceed to next variable
```

Out[12]:

In [13]:

```
plt.boxplot(data["registered"])
#As we have no outliers we proceed to next task of removing the outliers
```

Out[13]:

In [65]:

```
#Here we are going to calculate the percentiles i.e 25,50,75 and calculate th
e outer regions and remove
#the outliers that are present outside the regions

#calculating 75th and 25th percentile
q75,q25=np.percentile(data['windspeed'],[75,25])
print(q75,q25)
#calculating 50th percentile
iqr= q75-q25
iqr
#calculating inner and outer fence
min1=q25-(iqr)
max1=q75+(iqr)
print(min1,max1)
data.loc[data['windspeed']<min1,'windspeed']=np.nan
data.loc[data['windspeed']>max1,'windspeed']=np.nan
```

```
0.2332145 0.13495
```

```
0.03668549999999998 0.3314789999999997
```

In [66]:

```
#calculating 75th and 25th percentile
q75,q25=np.percentile(data['casual'],[75,25])
print(q75,q25)
#calculating 50th percentile
iqr= q75-q25
iqr
#calculating inner and outer fence
min1=q25-(iqr)
max1=q75+(iqr)
print(min1,max1)
data.loc[data['casual']<min1,'casual']=np.nan
data.loc[data['casual']>max1,'casual']=np.nan
```

```
1096.0 315.5
-465.0 1876.5
```

In [67]:

```
#checking the no of missing values after making the outlier values NULL, You
can refer to missing value analysis done above to
# Check the missing values for each variable that we have done previously

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

Out[67]:

	0
season	0
yr	0
mnth	0
weekday	0
workingday	0
weathersit	0
temp	0

	0
atemp	0
hum	0
windspeed	46
casual	73
registered	0
cnt	0

In [68]:

```
#Setting the best fitted method for filling the NULL values.
data['windspeed']=data['windspeed'].fillna(data['windspeed'].mean())
data['casual']=data['casual'].fillna(data['casual'].mean())
```

In [69]:

```
#After doing outlier analysis the windspeed variable is
plt.boxplot(data["windspeed"])
```

Out[69]:

In [19]:

```
#After doing outlier analysis the casual variable is
plt.boxplot(data["casual"])
```

Out[19]:

In [46]:

```
#Now we have to do the feature selection :

#correlation plot for numerical variables
num_cnames=['registered','casual','windspeed','hum','atemp','temp','cnt']
df_corr=data.loc[:,num_cnames]
```

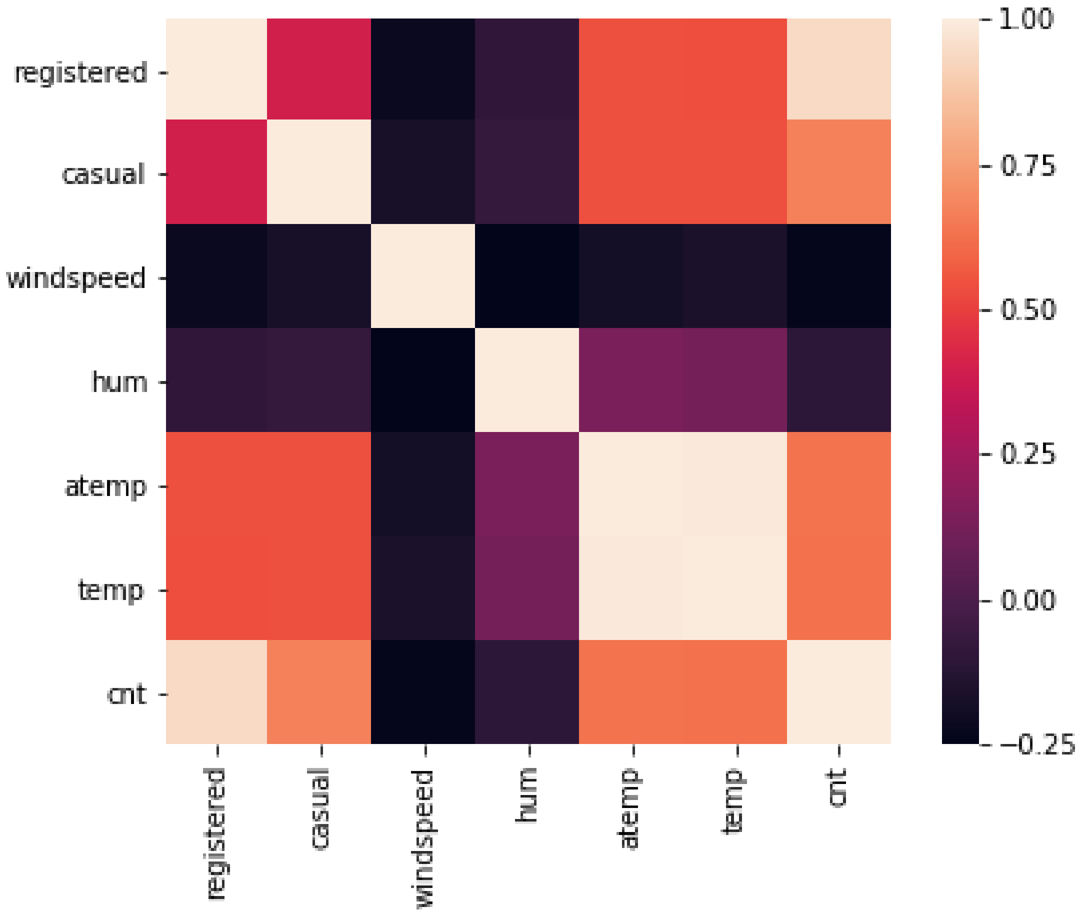
In [47]:

```
#set width and height of the plot
f,ax=plt.subplots(figsize=(7,5))

#generate co-relation matrix
corr=df_corr.corr()

#plotting
```

```
sns.heatmap(corr,mask=np.zeros_like(corr,dtype=np.bool),square=True,ax=ax)
```

Out[47]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1fce2a3ce80>
```



In [70]:

```
#here we can see that Both windspeed and hum are not having high relation wit
h cnt i.e our dependent variable so, lets's
#drop those two variables

data=data.drop(columns="windspeed")
data=data.drop(columns="hum")
```

In [71]:

```
#Checking the number of variables after removal
data.shape
```

Out[71]:

```
(731, 11)
```

In [72]:

```
#Checking the data
data.head(5)
```

Out[72]:

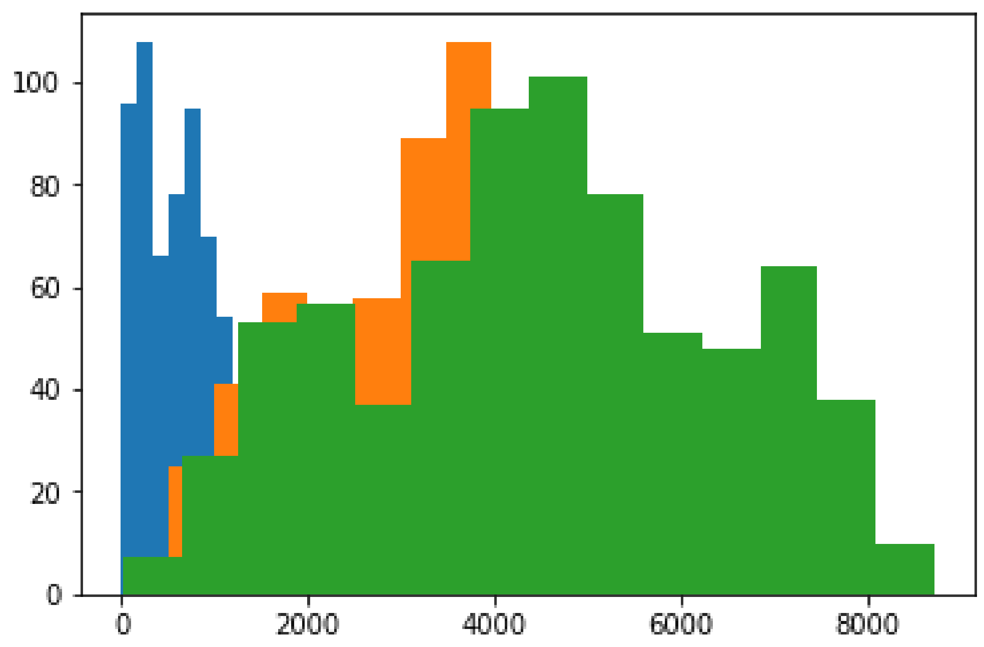
	seaso n	y r	mnt h	weekda y	workingda y	weathersi t	temp	atemp	casua l	registere d	cnt
0	1	0	1	6	0	2	0.34416 7	0.36362 5	331.0	654	985
1	1	0	1	0	0	2	0.36347 8	0.35373 9	131.0	670	801
2	1	0	1	1	1	1	0.19636 4	0.18940 5	120.0	1229	134 9
3	1	0	1	2	1	1	0.20000 0	0.21212 2	108.0	1454	156 2
4	1	0	1	3	1	1	0.22695 7	0.22927 0	82.0	1518	160 0

In [51]:

```
#performing feature scaling
df=data.copy()
num_cnames=['registered','casual','windspeed','hum','atemp','temp','cnt']
#normality check

plt.hist(data['casual'],bins='auto')
plt.hist(data['registered'],bins='auto')
plt.hist(data['cnt'],bins='auto')
```

Out[51]:

```
(array([  7.,  27.,  53.,  57.,  37.,  65.,  95., 101.,  78.,  51.,  48.,
         64.,  38.,  10.]),
 array([  22.        ,  642.85714286, 1263.71428571, 1884.57142857,
        2505.42857143, 3126.28571429, 3747.14285714, 4368.        ,
        4988.85714286, 5609.71428571, 6230.57142857, 6851.42857143,
        7472.28571429, 8093.14285714, 8714.        ]),
 <a list of 14 Patch objects>)
```



In [73]:

```
# Performing normalization:
cnames=['registered','casual','cnt']

for i in cnames:
    print(i)
    data[i]=(data[i]-np.min(data[i]))/(np.max(data[i])-np.min(data[i]))
```

```
registered
casual
cnt
```

In [74]:

```
#Sampling:

from sklearn.model_selection import train_test_split
train, test = train_test_split(data, test_size=0.2)
```

In [75]:

```
#Checking the No of Rows and columns of the train and test datasets
test.shape
```

Out[75]:

```
(147, 11)
```

In [76]:

```
train.shape
```

Out[76]:

```
(584, 11)
```

In [77]:

```
#Checking both the top 5 rows and columns
train.head(5)
```

Out[77]:

	season	yr	mnth	weekday	workingday	weathersit	temp	atemp	casual	registered	cnt
308	4	0	11	6	0	1	0.326667	0.323854	0.618104	0.397055	0.449149
137	2	0	5	3	1	2	0.550000	0.527158	0.286020	0.476321	0.440980
670	4	1	11	4	1	2	0.365833	0.369942	0.248527	0.794109	0.686148
67	1	0	3	3	1	2	0.295833	0.286608	0.101232	0.242564	0.215025
433	1	1	3	5	1	2	0.410833	0.397083	0.238350	0.592261	0.523125

In [31]:

```
test.head(5)
```

Out[31]:

	season	yr	mnth	weekday	workingday	weathersit	temp	atemp	casual	registered	cnt
281	4	0	10	0	0	1	0.540833	0.523983	0.358667	0.446722	0.631500
211	3	0	7	0	0	1	0.805833	0.729796	0.815212	0.398210	0.492407
16	1	0	1	1	0	2	0.175833	0.176771	0.061596	0.124603	0.112517
363	1	0	12	5	1	1	0.311667	0.318812	0.261918	0.359226	0.342499
462	2	1	4	6	0	1	0.437500	0.426129	0.358667	0.517615	0.786355

In [78]:

```
#Now we are making a copy of train and test datasets for future use
train_copy=train
```

```
test_copy=test
                                                                    In [79]:
#Now using machine learning models let's predict the values:

#Decision tree

from sklearn.tree import DecisionTreeRegressor
fit = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:9],train.iloc[:,1
0])
predictions_Dt=fit.predict(test.iloc[:,0:9])
                                                                    In [80]:
#Now calculating Mape for Decision tree regressor

def MAPE(y_true,y_pred):
    mape=np.mean(np.abs((y_true-y_pred)/y_true))
    print(mape)
MAPE(test.iloc[:,10],predictions_Dt)

#Error % = 35%
#Accuracy % = 65%
0.35889469126423207
                                                                    In [85]:
#Linear regression

import statsmodels.api as sm
model=sm.OLS(train.iloc[:,10].astype(float),train.iloc[:,0:9].astype(float)).
fit()
predictions_lm=model.predict(test.iloc[:,0:9])
                                                                    In [86]:
#Calculating Mape for Linear regression on the data
def MAPE(y_true,y_pred):
    mape=np.mean(np.abs((y_true-y_pred)/y_true))
    print(mape)
MAPE(test.iloc[:,10],predictions_lm)

#Error % = 19%
#Accuracy % = 81%
0.19966916340957316
                                                                    In [ ]:
#We conclude that linear regression is providing more accuarcy i.e 84% So, we
are considering the linear regression
# to use for our future datasets.
```