PREDICTING BIKE RENTING

BY : HARSHIT KAPOOR

December 2019

Contents

Introduction (1)

1.1 Problem Statement

Expanding the business mostly increases the profit books of the company and helps in generating more profit but one should always analyze the basic factors which will effect the growth of it. In this project our aim is to ‘Predict the count of Bikes on daily basis’ which have been provided on rent in a particular area and examine if there is some fluctuation due to the environmental and seasonal changes.

1.2 Data

We will build a Regression Model on this data which will help us to examine the core things which play important role on customers’ life and behavior hence affecting the selling of Bikes. Below is the Snapshot of the data which we will be using in determining the same.

Table 1.1 : Bike renting Sample Data(Columns :- 1-9)

Instant	Dteday	season	Yr	Mnth	holiday	weekday	Workingday	weathersit
1	01-01-11	1	0	1	0	6	0	2
2	02-01-11	1	0	1	0	0	0	2
3	03-01-11	1	0	1	0	1	1	1
4	04-01-11	1	0	1	0	2	1	1
5	05-01-11	1	0	1	0	3	1	1
6	06-01-11	1	0	1	0	4	1	1

Table 1.2 : Bike renting Sample Data(Columns :- 10-14)

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
0.204348	0.233209	0.518261	0.089565	88	1518	1606

As you can see in the table 1.3 below we have the following 13 variables, using which we have to correctly predict the quantity of bikes.
Moreover Table 1.4 shows 3 dependent variables which are also called target variable, which have to be predicted using the other 13 variables.

Table 1.3: Predictor Variable

S.No	Variables
1	instant
2	dteday
3	season
4	Yr
5	Mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	Temp
11	atemp
12	Hum
13	windspeed

Table 1.4: Dependent Variable / Target Variable

S.No	Variables
1	Casual
2	registered
3	Cnt

Note :- As we have to predict the total numbers of bikes provided on rent, hence we will be considering only the **"cnt"** variable as dependent and will eliminate **"Casual"** and **"registered".**

The Cnt Variable is the sum of Casual and registered.

Methodology (2)

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.

2.1.1 Missing Value Analysis

The first step of Pre processing is to find the Missing observations after getting the data in the environment. We have checked the missing values in both R and Python and have observed that there is **No Missing Value** in the data, If there are some missing values in a data then we can overcome them with some technical methods like- Mean, Median, Mode or KNN imputation which help us to fill the empty spaces and make the data fit for algorithms to work upon it.

We can only use these methods if the missing values are less than 30% of the data, else it is recommended to drop that particular variable as it won't have the true values of itself and can make the algorithm bias towards a particular result.

2.1.2 Outlier Analysis

This Pre processing technique helps us to find the observations which are inconsistent with respect to the rest of the data and in this case we use a classic approach of removing outliers i.e. Tukey's method. We visualize the outliers using boxplots. In figures below we have plotted the boxplots of the 13 predictor variables. A lot of useful inferences can be made from these plots which will help us to understand our data well.

Here are the visualizations of boxplots which have been performed in **both R and Python** with respect to all the 13 predictor variables.

Box Plot in **R**

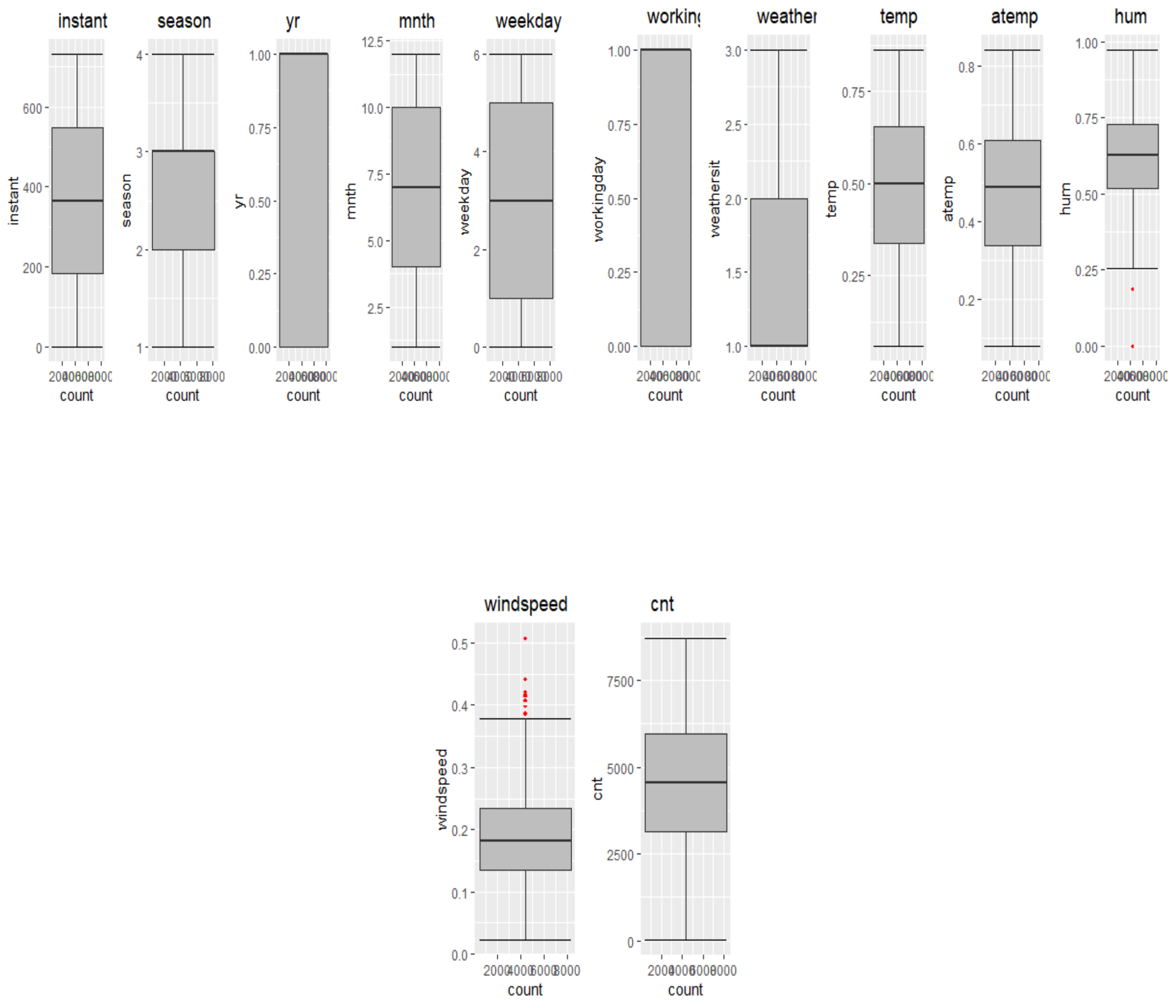
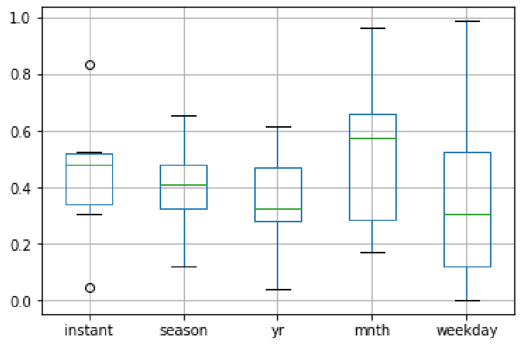
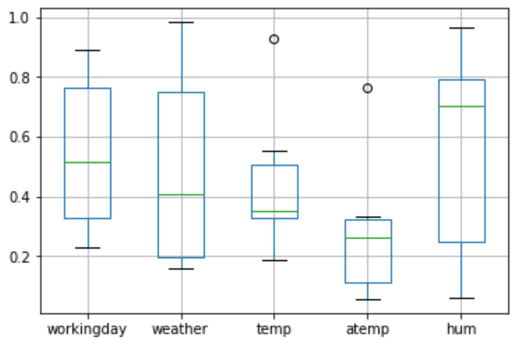


Figure 2.1 : 12 Boxplots of Predictor Variable(In R)

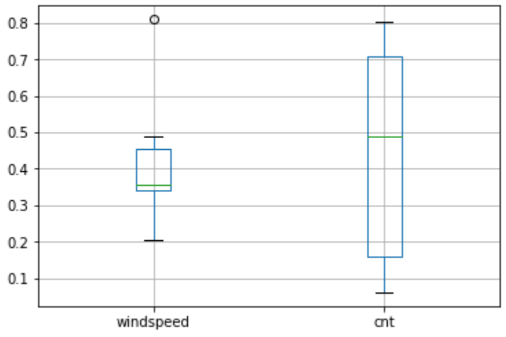
Box Plot in **Python**



(a)



(b)



(c)

Figure 2.2 (a, b, c) : 12 Boxplots of Predictor Variable(In Python)

After visualizing all the variables through boxplot it was seen that Variables with names **Hum**, **windspeed atemp** and **temp** were having some inconsistent variables which were then removed.

Note:- As the percentage of value was very less i.e. 1.91% (14 out of 731) I preferred to drop these variables rather than imputing them with other values.

2.1.3 Feature Selection

After excluding the unwanted observation it is necessary to check that which all variables are important to us in building the model so that we can focus on them only and exclude the rest of them. There is a possibility that many variables in our analysis are not important at all to the problem of class prediction. There are several methods of finding this but we have chosen **Correlation Analysis** and **Chi-Square test**.

- Correlation Analysis for continuous variable.
- Chi-Square test for categorical variable.

Here is the image which explains us about the Correlation between variables and help us to decide which variable is important for our prediction and which should not be considered.

One more important aspect of correlation analysis is that it also helps us to examine whether more than one variable are containing the same information and hence making the model bias. If this is found then only one variable should be allowed to impute its properties in the model.

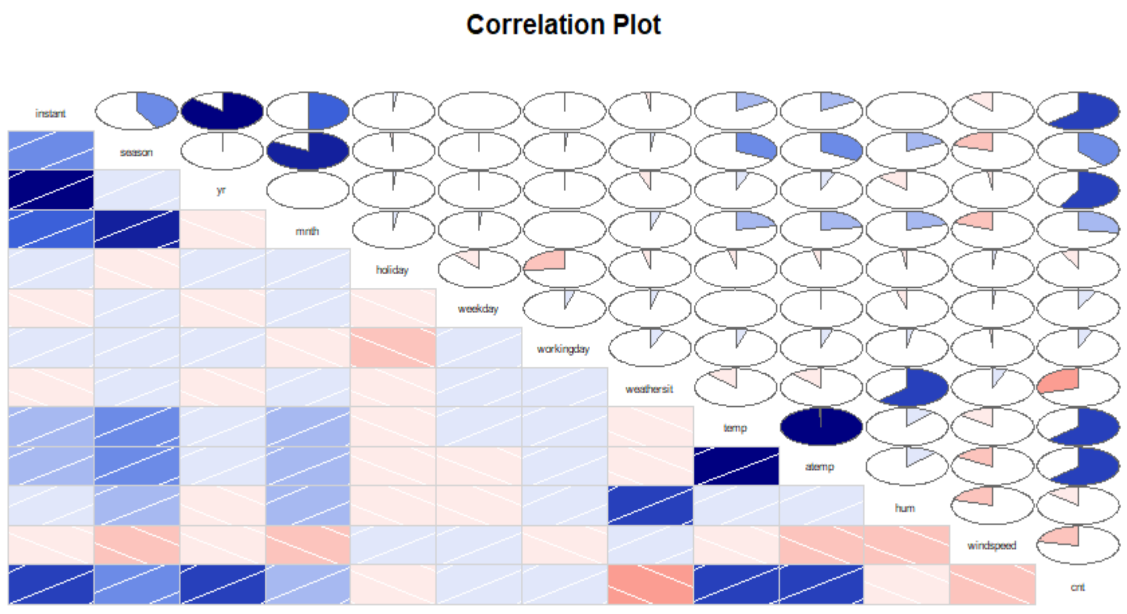


Figure 2.3: Correlation Analysis of continuous variables.(In R)

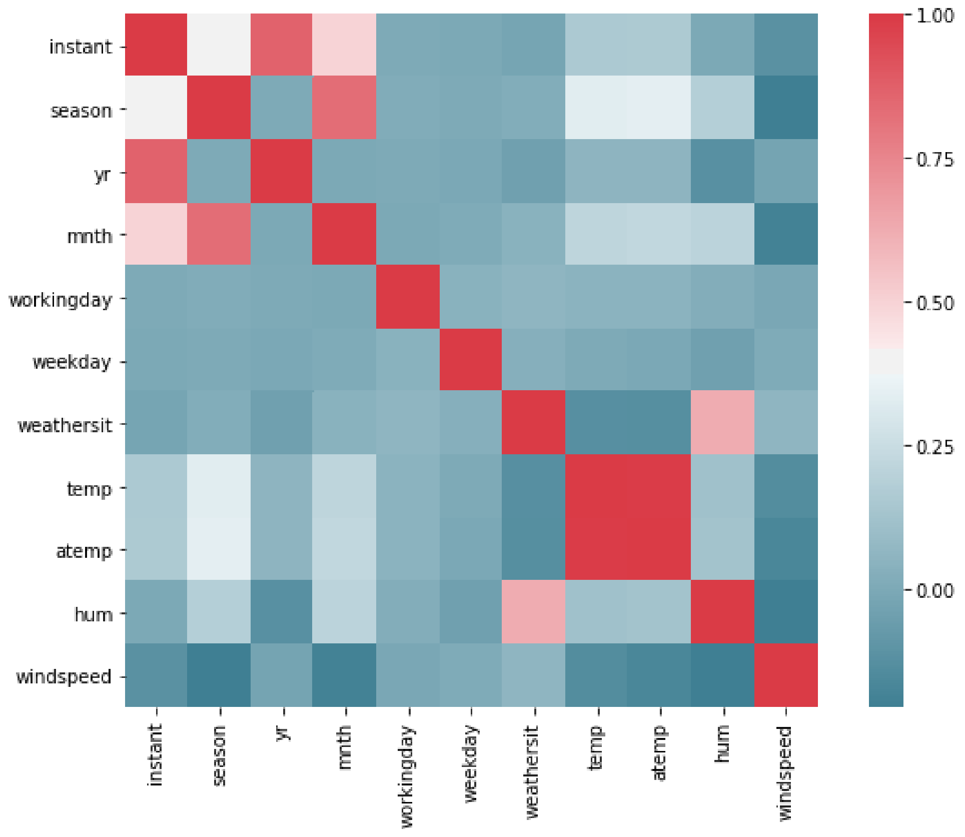


Figure 2.4: Correlation Analysis of continuous variables.(In Python)

After running correlation analysis in programming languages it has been noticed that variables with names

- Season and month ;
- temp and atemp ;
- Instant (with month and yr)

are positively correlated with each other and hence contain the same information which can result in the wrong assumption in the modeling stage hence one from them has to be removed before feeding this data to our model.

Now after sorting all the continuous data we will move forward towards our categorical variable i.e. dteday. After passing this variable through Chi-Square test the value came out to be greater than our threshold value hence we have to omit this variable also.

So, Variables with names **season, temp, instant** and **dteday** are removed.

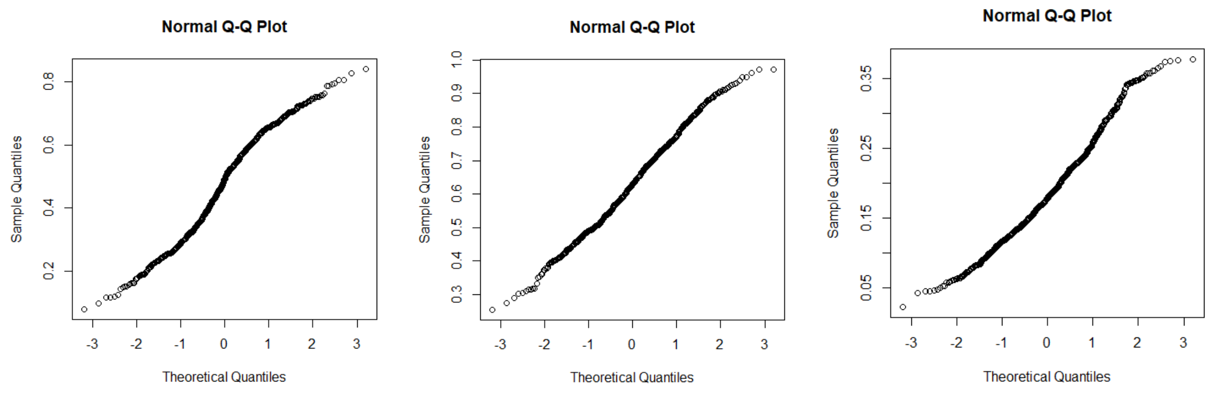
2.1.4 Feature Scaling

Now we are here with feature scaling which will help us to make our data in the same Range.

We have two methods for this, namely :-

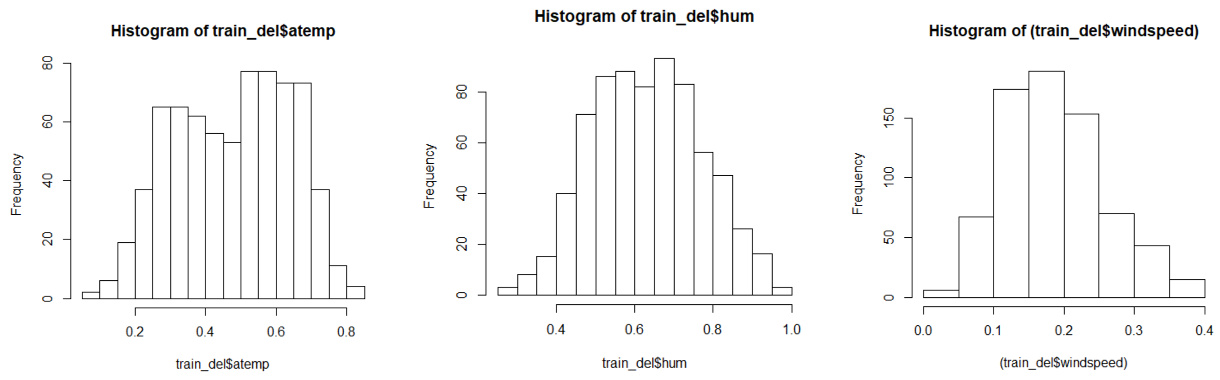
- **Normalization** - If the data is not normally distributed.
- **Standardization** - If the data is normally distributed

To find whether the data is uniformly distributed or not we have used some graphical technique like normality Graph(Q-Q Graph) and Histogram on the continuous variables so that we can drop them all on the same scale. The graphs have been depicted below.



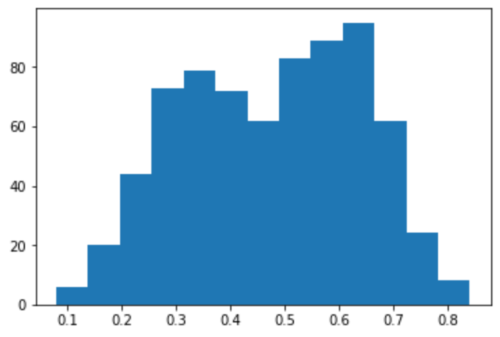
(a) (b) (c)

Figure 2.5 (a, b, c) :- Normality graph of atemp, hum and windspeed(In R).

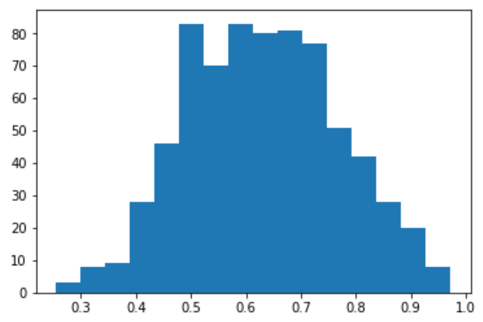


(a) (b) (c)

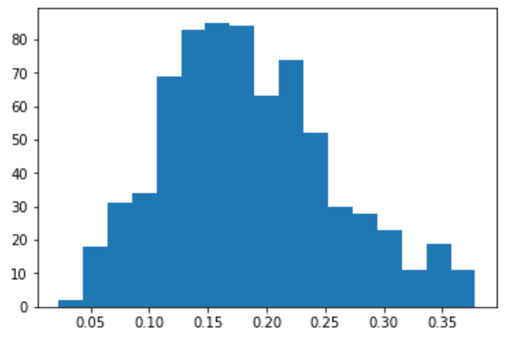
Figure 2.6 (a, b, c) :- Histogram of atemp, hum and windspeed(In R).



(a)



(b)



(c)

Figure 2.7 (a, b, c) :- Histogram of atemp, hum and windspeed (In Python).

Now when we have drawn the different graphs to study the nature of observations, it was noticed that they were **uniformly distributed.** Hence we will be moving ahead considering the data as Uniformly Distributed.

So, we have converted the observation of continuous variables(atemp, windspeed, hum) and defined them on a single scale.

The formula of standardization is :-

$$Z = \frac{x - \mu}{\sigma}$$

where,

x = value,

μ = Mean,

σ = Stand. Deviation

2.2 Modeling

2.2.1 Model Selection

Now when we have made the data free from all the default values and have eliminated the variables which do not play important role in predicting the outcome, It's time to try our data on the models and to check the predicted values.

We need to analyse the data and generate models for the types of data sets.

The dependent variable can fall in either of the Three categories:

1. Nominal
2. Ordinal
3. Interval/Ratio

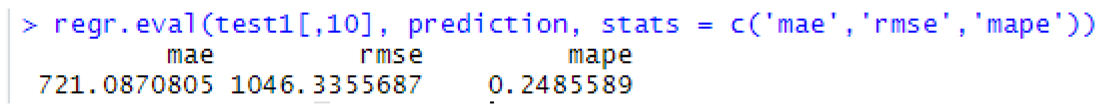
Our dependent variable (cnt) falls under Interval variable therefore normal method is to do a **Regression** analysis.
You always start your model building from simple to complex. Therefore I have selected the below provided pattern.

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

2.2.2 Decision Tree.

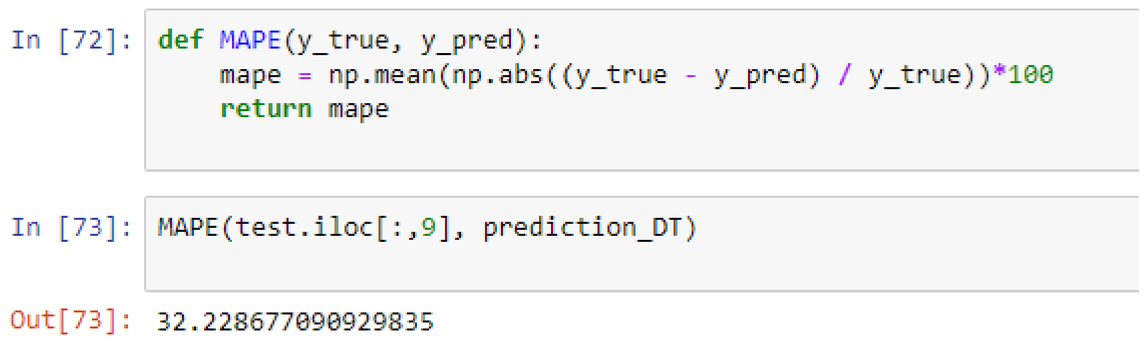
This model is based on branching series of Boolean test and can be used for both **classification** and **regression**. This model is the simplest and can be easily understood by the business users also.

After dividing the data into test and train the **MAPE** came out to be 0.2485589 that means the **error percentage** of it is **24.85589%** in R, where as 32.228677% was observed in Python.



```
> regr.eval(test1[,10], prediction, stats = c('mae','rmse','mape'))
         mae         rmse         mape
 721.0870805 1046.3355687    0.2485589
```

Figure 2.8 :- Values of MAE, RMSE, MAPE (In R).



```
In [72]: def MAPE(y_true, y_pred):
             mape = np.mean(np.abs((y_true - y_pred) / y_true))*100
             return mape

In [73]: MAPE(test.iloc[:,9], prediction_DT)

Out[73]: 32.228677090929835
```

Figure 2.9 :- Values of MAPE (In Python).

2.2.3 Linear Regression

Linear Regression is **only used for Regression** model and has some assumptions i.e

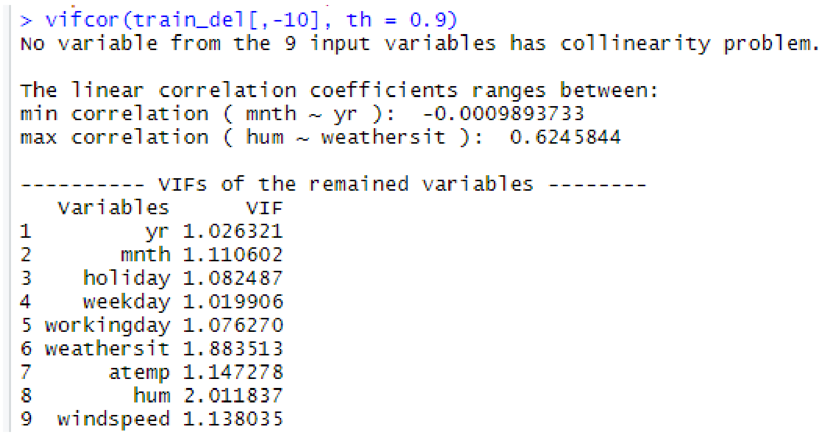
- Target variable should be normally distributed.
- There should be no or less Multicollinarity.
- No Auto correlation.

To check the Collinearity we first analyze the observation of **VIF**.

VIF (variance inflation factor) is used to find whether the variables have collinearity problem or not. Firstly it will find the pair of variables which have maximum correlation and will work upon it. Higher values signify that it is difficult to assess accurately the contribution of predictors to a model.

If the VIF is greater than 10 then it implies that the data has the collinearity problem and it needs to be resolved.

Below is the snapshot of observation found in R.

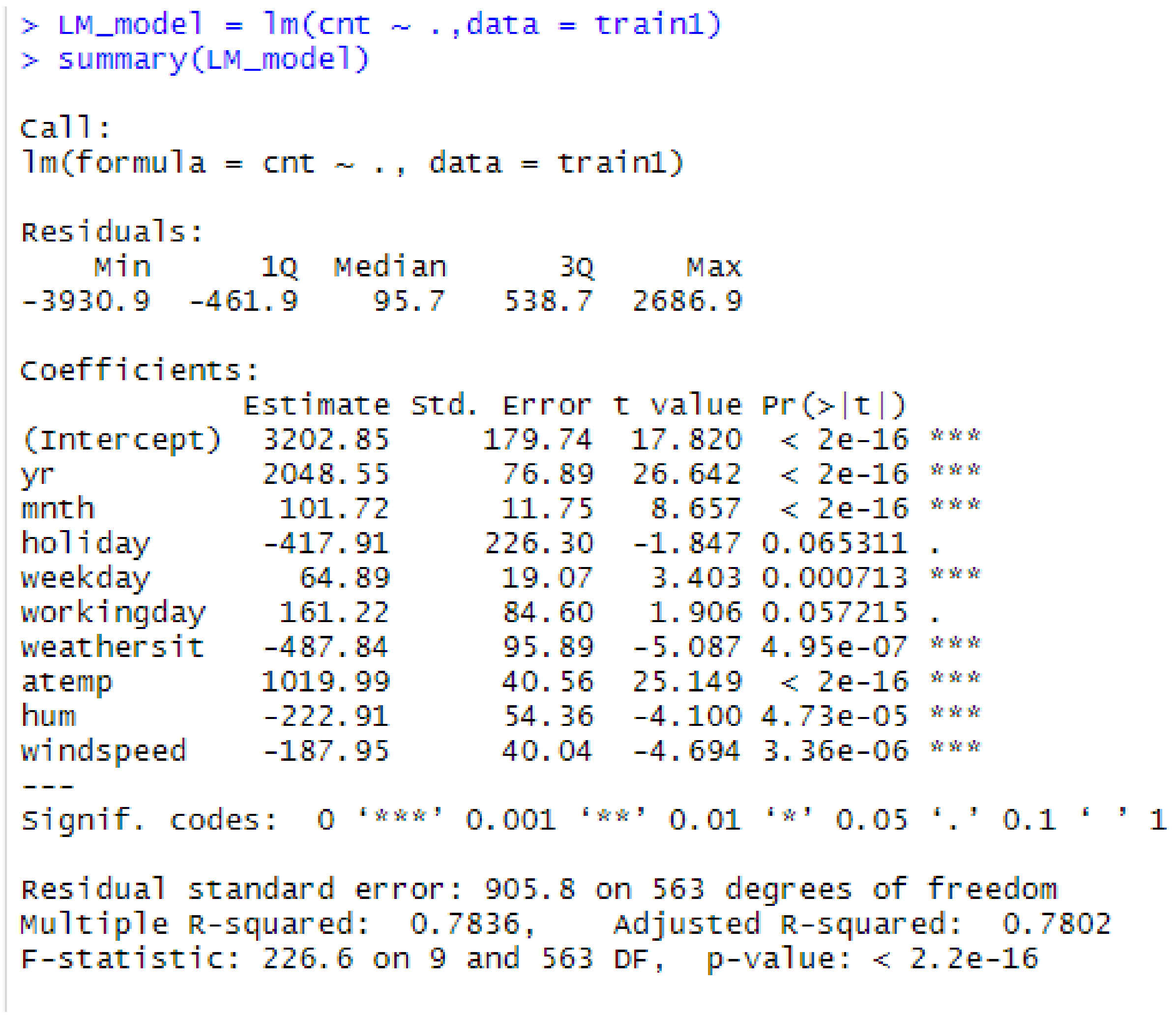


```
> vifcor(train_del[,-10], th = 0.9)
No variable from the 9 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation ( mnth ~ yr ):  -0.0009893733
max correlation ( hum ~ weathersit ):  0.6245844

---------- VIFs of the remained variables --------
   Variables      VIF
1         yr 1.026321
2       mnth 1.110602
3    holiday 1.082487
4    weekday 1.019906
5 workingday 1.076270
6 weathersit 1.883513
7      atemp 1.147278
8        hum 2.011837
9  windspeed 1.138035
```

Figure 2.10 :- Observations of VIF (In R).

Now it is clear that our variables are free from collinearity and will not make the model bias towards a particular decision.

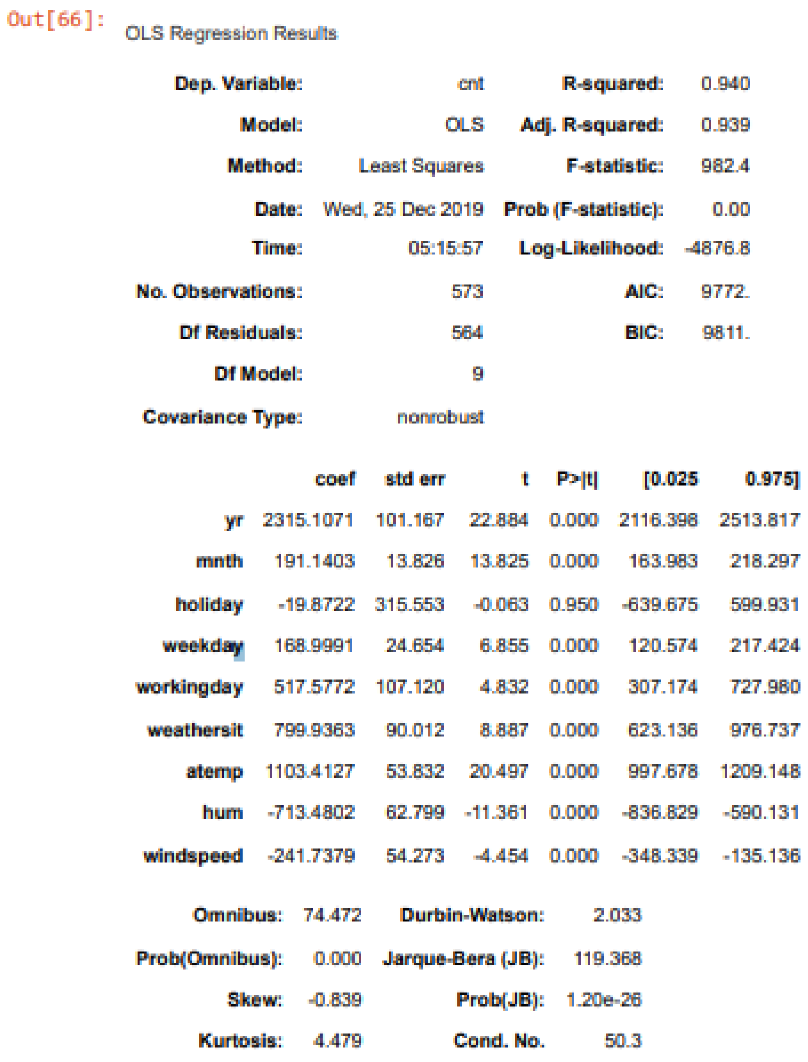
After running the codes in both the tools we have examined the summary which helps us to know more about the data in Linear Regression which is provided below.



```
> LM_model = lm(cnt ~ .,data = train1)
> summary(LM_model)

Call:
lm(formula = cnt ~ ., data = train1)

Residuals:
    Min      1Q  Median      3Q     Max 
-3930.9  -461.9    95.7   538.7  2686.9 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  3202.85     179.74  17.820  < 2e-16 ***
yr           2048.55      76.89  26.642  < 2e-16 ***
mnth          101.72      11.75   8.657  < 2e-16 ***
holiday      -417.91     226.30  -1.847 0.065311 .  
weekday        64.89      19.07   3.403 0.000713 ***
workingday    161.22      84.60   1.906 0.057215 .  
weathersit   -487.84      95.89  -5.087 4.95e-07 ***
atemp        1019.99      40.56  25.149  < 2e-16 ***
hum          -222.91      54.36  -4.100 4.73e-05 ***
windspeed    -187.95      40.04  -4.694 3.36e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 905.8 on 563 degrees of freedom
Multiple R-squared:  0.7836,	Adjusted R-squared:  0.7802 
F-statistic: 226.6 on 9 and 563 DF,  p-value: < 2.2e-16
```

Figure 2.11:- Summary of Data through Linear Regression (In R).

- We have observed the table and have looked into terms like Residual, R-squared, Adj R-squared, No of observation, t-value, probability value, degree of freedom.



```
Out[66]:
         OLS Regression Results

       Dep. Variable:              cnt          R-squared:     0.940
              Model:               OLS     Adj. R-squared:     0.939
             Method:     Least Squares         F-statistic:     982.4
               Date:  Wed, 25 Dec 2019  Prob (F-statistic):      0.00
               Time:          05:15:57      Log-Likelihood:   -4876.8
   No. Observations:               573                 AIC:     9772.
       Df Residuals:               564                 BIC:     9811.
           Df Model:                 9
    Covariance Type:         nonrobust

                    coef   std err        t   P>|t|    [0.025    0.975]
           yr  2315.1071   101.167   22.884   0.000  2116.398  2513.817
         mnth   191.1403    13.826   13.825   0.000   163.983   218.297
      holiday   -19.8722   315.553   -0.063   0.950  -639.675   599.931
      weekday   168.9991    24.654    6.855   0.000   120.574   217.424
   workingday   517.5772   107.120    4.832   0.000   307.174   727.980
   weathersit   799.9363    90.012    8.887   0.000   623.136   976.737
        atemp  1103.4127    53.832   20.497   0.000   997.678  1209.148
          hum  -713.4802    62.799  -11.361   0.000  -836.829  -590.131
    windspeed  -241.7379    54.273   -4.454   0.000  -348.339  -135.136

         Omnibus:  74.472    Durbin-Watson:     2.033
   Prob(Omnibus):   0.000  Jarque-Bera (JB):  119.368
            Skew:  -0.839          Prob(JB):  1.20e-26
        Kurtosis:   4.479          Cond. No.      50.3
```

Figure 2.12:- Summary of Data through Linear Regression (In Python).

- We have observed the table and have looked into terms like R-square, Adj R-square, No of observation, t-value, p-value, AIC and BIC.

The error rate in both the tools after running the algorithms (code provided at last) were:-

R = 1.58958e-01

i.e. 15.8958

Python = 19.07732163131703

2.2.4 Random Forest.

This method also works on **both classification** and **regression** model and is an ensemble of many decision Trees. This methods combines Breimen's Bagging idea and the random selection of features.

In this model the number of trees depends upon the error rate. If the error rate is not decreasing even after increasing the trees then we will stop there and will not feed more number of trees.

Below is the table with the error rate and accuracy found after running the codes in tools for different number of trees.

No of Trees	**100**	**300**	**500**	**700**	**1000**
Error Rate	7.4611%	7.3609%	7.1848%	7.1793%	7.2378%
Accuracy	92.5389%	92.6391%	92.8152%	92.8207%	92.7622%

Table 2.13:- Observation of Error Rate and Accuracy (In R).

No of Trees	**100**	**300**	**500**	**700**	**1000**
Error Rate	10.5748%	9.8730%	9.4508%	9.9543%	9.2378%
Accuracy	89.4252%	90.1270%	90.5492%	90.0457%	90.7622%

Table 2.14:- Observation of Error Rate and Accuracy (In Python).

Conclusion (3)

Now that we have a few models for predicting the target variable, we need to decide which one to choose.
There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In our case the last two, Interpretability and Computation Efficiency, do not hold much significance. Therefore we will use Predictive performance as the criteria to compare and evaluate models.

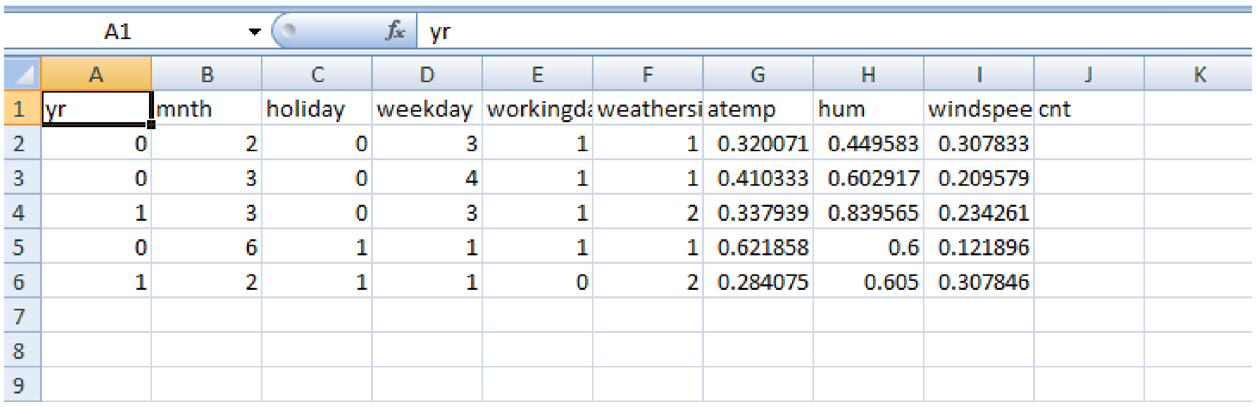
So,

We have observed that the **Random Forest** has performed pretty well as compared to rest of the two hence I will choose this model to forecast the renting of bikes.

The numbers of trees I will take for my predicted data is **700** as it shows the minimum error rate.

Prediction (4)

I have randomly made these 5 tables to check my model in excel. Below is the image of the same.



A1 | yr

	A	B	C	D	E	F	G	H	I	J	K
1	yr	mnth	holiday	weekday	workingda	weathersi	atemp	hum	windspee	cnt	
2	0	2	0	3	1	1	0.320071	0.449583	0.307833		
3	0	3	0	4	1	1	0.410333	0.602917	0.209579		
4	1	3	0	3	1	2	0.337939	0.839565	0.234261		
5	0	6	1	1	1	1	0.621858	0.6	0.121896		
6	1	2	1	1	0	2	0.284075	0.605	0.307846		
7											
8											
9											

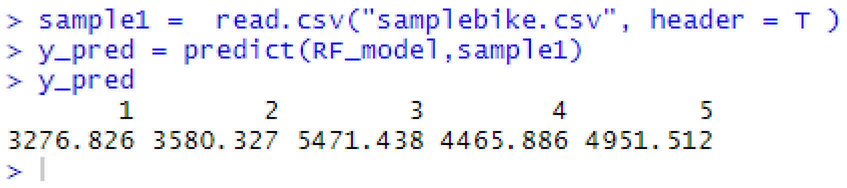
Figure 4.1:- Test data used for Prediction.

Now, after running the code I have collected the below mentioned observations.

No of Bike Rented	Day 1	Day 2	Day 3	Day 4	Day 5
R	3277	3580	5471	4465	4952
Python	3198	3428	5632	4572	4809

Table 4.2:- Observed Values in R and Python.

- In R the values were coming in decimal (can be seen below) and have been made round-off in the table.



```
> sample1 =  read.csv("samplebike.csv", header = T )
> y_pred = predict(RF_model,sample1)
> y_pred
       1        2        3        4        5 
3276.826 3580.327 5471.438 4465.886 4951.512 
> 
```

Figure 4.2:- Output of the prediction made on sample data

CODE's :-

In R :-

```
rm(list = ls())

setwd("C:/Users/DELL/Desktop/Project 2")

getwd()

#Import our data.

train=  read.csv("day.csv", header = T )


train$casual = NULL

train$registered = NULL


#Missing Value Analysis

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


#we have found ZERO missing Value.


#Outlier Analysis.

numeric_index = sapply(train, is.numeric)

numeric_data = train[,numeric_index]

cnames = colnames(numeric_data)
```

```
install.packages("ggplot2")

library("ggplot2")




for (i in 1:length(cnames))

{

  assign(paste0("gn",i),ggplot(train, aes_string(x = "cnt", y = (cnames[i]))) +

                           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 of",cnames[i])))

}



gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=6)

gridExtra::grid.arrange(gn7,gn8,gn9,gn10,gn11,gn12,ncol=6)

gridExtra::grid.arrange(gn13,ncol=1)



#drop of variables


val1 = train$hum[train$hum %in% boxplot.stats(train$hum)$out]

train = train[which(!train$hum %in% val1),]


val = train$windspeed[train$windspeed %in% boxplot.stats(train$windspeed)$out]
```

```
train = train[which(!train$windspeed %in% val),]



library(corrgram)

#Correlation Plot

corrgram(train[,numeric_index], order = FALSE,

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


#chi-square Test

print(chisq.test(table(train$cnt, train$dteday)))

#rejected as p value is greater than 0.05 "0.1965737837364"


#we will drop instant, season/month and temp/atemp and dteday ; weathersit/hum(can be
considered)


train_del = subset(train,

                    select = -c(instant, season, temp, dteday))




#Q-Q Graph

qqnorm(train_del$windspeed)

qqnorm(train_del$hum)

qqnorm(train_del$atemp)




hist((train_del$windspeed))

hist(train_del$hum)

hist(train_del$atemp)
```

```
#data is uniformlly distributed



#standardisation(as data is formally distributed)


dnames = c("atemp","hum","windspeed")

for(i in dnames){

  print(i)

  train_del[,i] = (train_del[,i] - mean(train_del[,i]))/sd(train_del[,i])

}


##### Decision Tree


#train-test data

train_index = sample(1:nrow(train_del), 0.8 * nrow(train_del))

train1 = train_del[train_index,]

test1 = train_del[-train_index,]



install.packages("rpart")

library(rpart)

fit = rpart(cnt ~ .,data =train1, method = "anova")

prediction = predict(fit, test1[-10])


library(DMwR)

regr.eval(test1[,10], prediction, stats = c('mae','rmse','mape','mse'))

#MAPE = 0.2485589
```

```
###### Linear Regression

install.packages("usdm")
library(usdm)
vifcor(train_del[,-10], th = 0.9)
#No collinearity found between variables

LM_model = lm(cnt ~ ., data = train_del)
summary(LM_model)
predict_LM = predict(LM_model, test1[,-10])
regr.eval(test1[,10], predict_LM, stats = c('mae','rmse','mape','mse'))
#MAPE = 1.58958e-01 => error rate

##### Random Forest

library("randomForest")
RF_model = randomForest(cnt ~ .,train_del, importance = TRUE, ntree = 1000)
treelist = RF2List(RF_model)

RF_predict = predict(RF_model, test1[,-10])
regr.eval(test1[,10], RF_predict, stats = c('mae','rmse','mape','mse'))
#MAPE(100) = 7.461142e-02 => error rate
#MAPE(300) = 7.360942e-02
#MAPE(500) = 7.184840e-02
#MAPE(700) = 7.179334e-02
#MAPE(1000) =7.237828e-02
```

```
########### Prediciting the model on sample Input.

sample1 =  read.csv("samplebike.csv", header = T )

y_pred = predict(RF_model,sample1)
y_pred

1        2        3        4        5
3276.826 3580.327 5471.438 4465.886 4951.512
```

In PYTHON :-

```
import os
import pandas as pd
import matplotlib as mlt
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns

os.chdir("E:\EDWISOR\Project 2")
###Importing Data
data = pd.read_csv("day.csv",sep = ',')
```

```
num_col = data._get_numeric_data().columns


num_col


data.shape


#Missing Value Analysis

missing_val.head()

#we have found ZERO missing Value.


del data["registered"]

del data["casual"]


data.shape

#Outlier Analysis.

#Plot the graph


df = pd.DataFrame(data = np.random.random(size=(6,5)), columns =
['instant','season','yr','mnth','weekday'])

df.boxplot()


df = pd.DataFrame(data = np.random.random(size=(6,5)), columns =
['workingday','weather','temp','atemp','hum'])

df.boxplot()


df = pd.DataFrame(data = np.random.random(size=(6,2)), columns = ['windspeed','cnt'])

df.boxplot()
```

```
cnames =
["instant","season","yr","mnth","workingday","weekday","weathersit","temp","atemp","hum","wi
ndspeed"]


for i in cnames:
    print(i)
    q75, q25 = np.percentile(data.loc[:,i],[75 ,25])
    iqr = q75 - q25

    min = q25 - (iqr*1.5)
    max = q75 + (iqr*1.5)
    print(min)
    print(max)

    data = data.drop(data[data.loc[:,i]< min].index)
    data = data.drop(data[data.loc[:,i]> max].index)

data.shape
# the data is left with 717 and 14 variables.

df_corr = data.loc[:,cnames]



f, ax = plt.subplots(figsize = (7,5))
corr = df_corr.corr()
sns.heatmap(corr, mask=np.zeros_like(corr, dtype=np.bool),cmap =
sns.diverging_palette(220,10,as_cmap=True),square=True,ax=ax)

#chi-square Test
```

```
chi2, p, dof, ex = chi2_contingency(pd.crosstab(data['cnt'],data['dteday']))

print(p)


# 0.24479148076360824

#rejected as p value is greater than 0.05  "0.24479148076360824"



#we will drop instant, season/month and temp/atemp and dteday(column data hence axis is 1,
rows = 0)

#drop of variables

data = data.drop(['instant','season','temp','dteday'], axis=1)


data.shape


%matplotlib inline

plt.hist(data['hum'], bins='auto')


#Standardization as the data is uniformally distributed

dnames = ["atemp","hum","windspeed"]


for i in dnames:

   print(i)

   data[i] = (data[i] - data[i].mean())/data[i].std()


data.head()


            #################   Models   ###################
```

```
from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeRegressor


train, test = train_test_split(data, test_size=0.2)


##### Decision Tree


fit = DecisionTreeRegressor(max_depth=2).fit(train.iloc[:,0:9], train.iloc[:,9])


prediction_DT = fit.predict(test.iloc[:,0:9])


def MAPE(y_true, y_pred):

    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100

    return mape


MAPE(test.iloc[:,9], prediction_DT)

# error rate => 32.228677



###### Linear Regression


import statsmodels.api as sm


model = sm.OLS(train.iloc[:,9], train.iloc[:,0:9]).fit()


model.summary()
```

```
predictions_LR = model.predict(test.iloc[:,0:9])


MAPE(test.iloc[:,9], predictions_LR)

#erros rate => 19.07732163131703


###### Random Forest


from sklearn.ensemble import RandomForestClassifier


RF_model = RandomForestClassifier(n_estimators = 1000, random_state= 0).fit(train.iloc[:,0:9],
train.iloc[:,9])


RF_model


y_pred = RF_model.predict(test.iloc[:,0:9])


def MAPE(y_true, y_pred):

    mape = np.mean(np.abs((y_true - y_pred) / y_true))*100

    return mape

MAPE(test.iloc[:,9], prediction_DT)


#MAPE(100) = 10.5747837378742 => error rate

#MAPE(300) = 9.8729610384042

#MAPE(500) = 9.4508368429104

#MAPE(700) = 9.9812095529152

#MAPE(1000)= 9.9542966374954
```

```
########### Prediciting the model on sample Input.

sample = pd.read_csv("samplebike.csv",sep = ',')

sample.head()

sample_pred = RF_model.predict(sample.iloc[:,0:9])

sample_pred
```