Bike Count Prediction

By

Vinod.M

Synopsis

1) Introduction
    a) Problem Statement
    b) Business Understanding
    c) Data Understanding
2) Exploratory Data analysis Or Data-Preprocessing
    a) Missing Value
    b) Outliers
    c) Feature Selection
    d) Feature Scaling
    e) Visualization
3) Model Development
    a) Sampling
    b) Linear Regression
    c) Decision Tree
    d) Random Forest
    e) XGboost

4) Error-Metrics
5) Finding and Conclusion

Appendix:

R program,Python program

Introduction

Problem Statement:

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings .I need to design a system that predicts the bike count based on environmental and seasonal settings .Because it is important to estimate the bike count so that the company can plan accordingly as it changes based on environmental and seasonal settings.

Business Understanding

It is important to determine the bike rental count in bike rental business. In case of Bike rental business there are many factor which determine the count such as season, temperature, humidity, wind speed ,Holiday ,Working day . Bike count is prime importance in Bike rental Business .It should be in an optimal amount to run the business in an success manner.So that I need to build a model which predict the count with the help of predictors available in the dataset

Two steps:

- Build the model with the help of training data, Check the accuracy of the model
- Deploy the best model into new dataset to predict the Bike count

<u>Data Understanding:</u>

<u>dataset:</u>

- In dataset, there is 731 observation and 16 Attributes
- Variables in dataset are:

a) instant: Record index of users
b) dteday: Date season: Season (1:springer, 2:summer, 3:fall, 4:winter)
c) yr: Year (0: 2011, 1:2012)
d) mnth: Month (1 to 12)
e) hr: Hour (0 to 23)
f) holiday: weather day is holiday or not (extracted from Holiday Schedule)
g) weekday: Day of the week working day: If day is neither weekend nor holiday is 1, otherwise is 0.
h) 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 – Weathers in different combination
i) temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale) - Normalised data
j) atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_min), t_min=-16, t_max=+50 (only in hourly scale) - Normalised data
k) hum: Normalized humidity. The values are divided to 100 (max) - Normalised data
l) windspeed: Normalized wind speed. The values are divided to 67 (max) –Normalised data
m)casual: count of casual users-on the spot user

n) registered: count of registered users-already registered user
o) cnt: count of total rental bikes including both casual and registered-summation of casual and registered

- 15-independent variable(Except-Count(cnt))
- One-dependent variable(only cnt)
- Missing value-No

Datatype of variable:

- Instant:integer
- dteday:Factor
- season:integer
- yr:integer
- mnth:integer
- holiday:integer
- weekday:integer
- Workingday:integer
- Weathersit:integer
- Temp:numerical
- Atemp:numerical
- Hum:numerical
- Windspeed:numerical
- Casual:integer
- Registered:integer
- Cnt:integer
- $ instant : int .
- $ dteday : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5 6 7 8 9 10 ...
- $ 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 ...

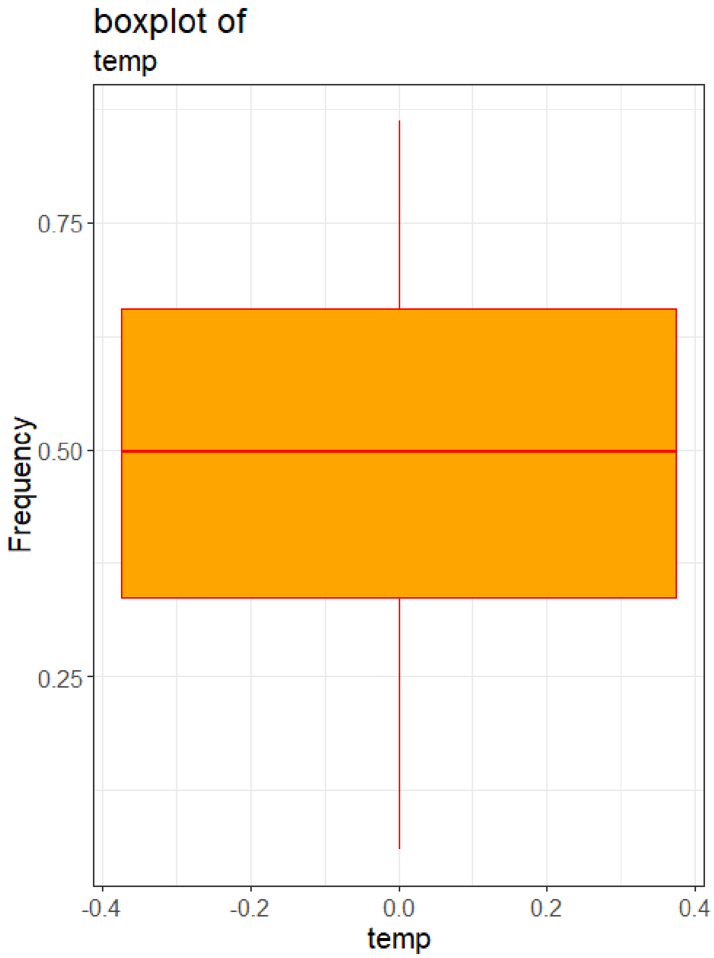
Exploratory Data analysis Or Data Preprocessing:

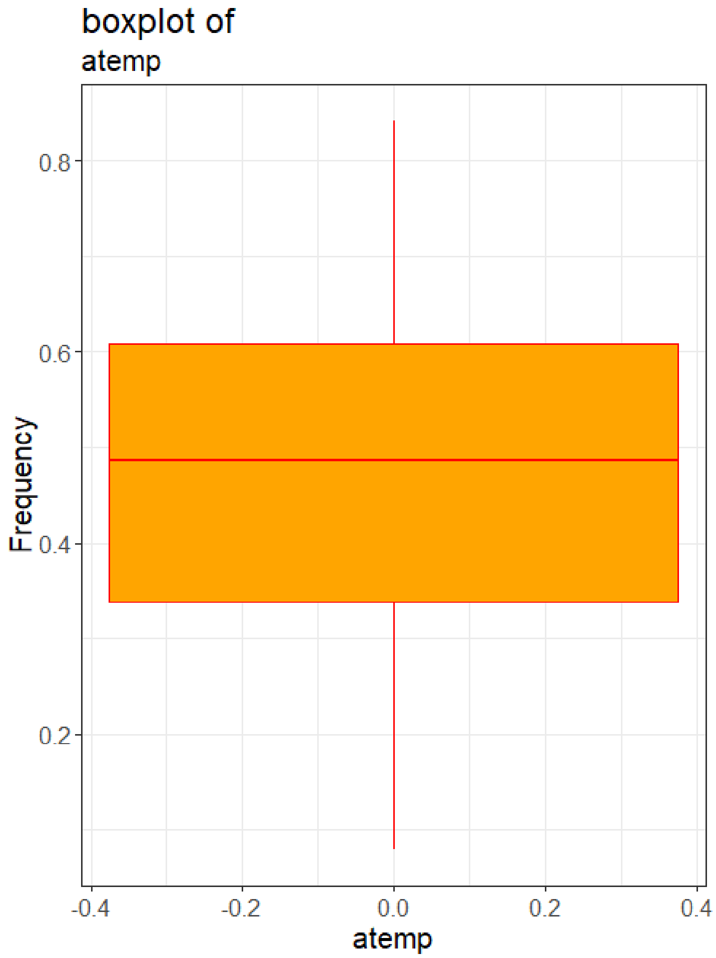
Missing value:

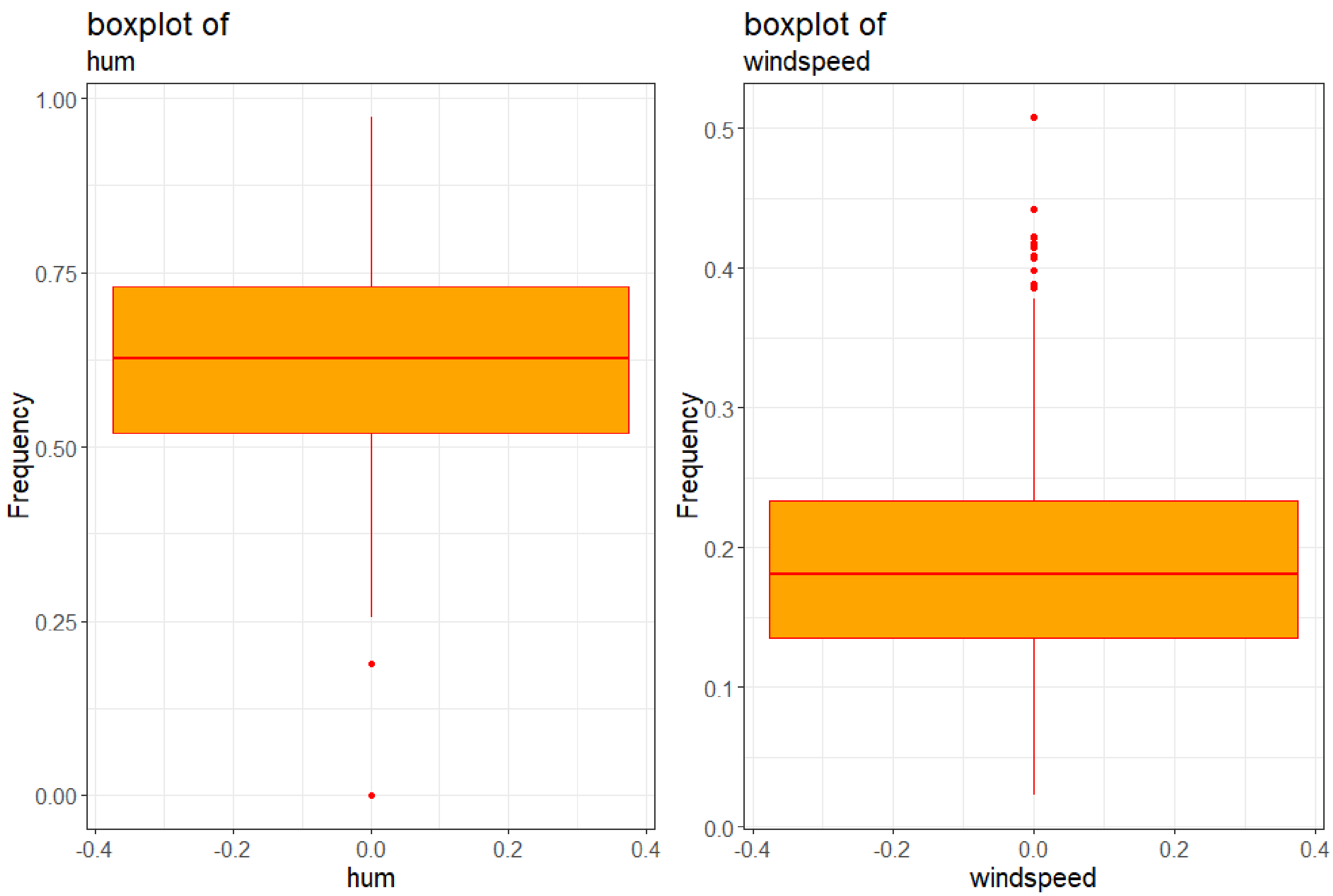
- Missing value is important step in data preprocessing process
- There are various method to impute the missing value such as mean,median,mode,KNN etc
- In this dataset there are no missing values ,all the values are filled in each cell ,so that there is not necessary to impute missing values ,as there is no missing values

Outliers:

- Outliers are the one in which the value move beyond the limit of 25% or 75% are removed or imputed
- In R outlier in the variable is found by using boxplot through visualization
- In R it can be found out by using boxplot.stats function
- In python it can be found out by using np.percentile function
- In the dataset I replaced the value with NA and I omitted the value because it will not hold good while handling the imputed dataset ,as it may produce inappropriate result


boxplot of
temp
0.75
0.50
0.25
Frequency
-0.4
-0.2
0.0
0.2
0.4
temp


boxplot of
atemp
0.8
0.6
0.4
0.2
Frequency
-0.4
-0.2
0.0
0.2
0.4
atemp



As there are some outliers mentioned in red dot which are to handled and remove those values because it may shrink the model accuracy

Feature selection:

Correlation is used to determine how well the variable are related between each other ,if there is more correlation between the independent and dependent variable it holds good,but if there is more correlation between the predictors then multicollinearity arises which leads to decrease in the accuracy of the model

Correlation Matrix:

	temp	atemp	windspeed	hum	cnt
temp	1.00	0.99	-0.16	0.13	0.63
atemp	0.99	1.00	-0.18	0.14	0.63
windspeed	-0.16	-0.18	1.00	-0.25	-0.23
hum	0.13	0.14	-0.25	1.00	-0.10
cnt	0.63	0.63	-0.23	-0.10	1.00

In this temp and atemp are highly correlated ,in which one variable need to be removed which leads to redundancy in the data,so atemp is removed from the dataset.

'season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathersit'

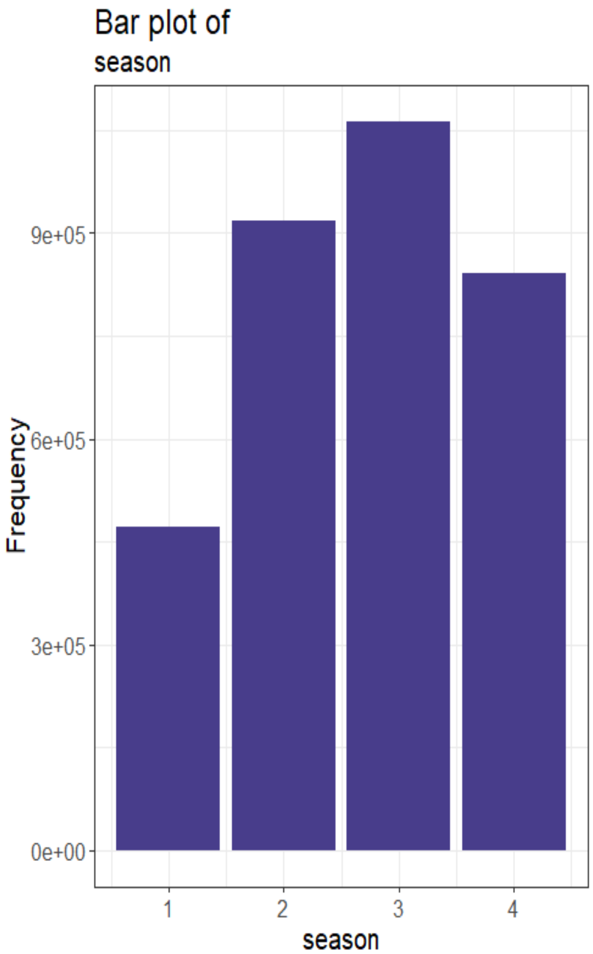
Feature scaling:

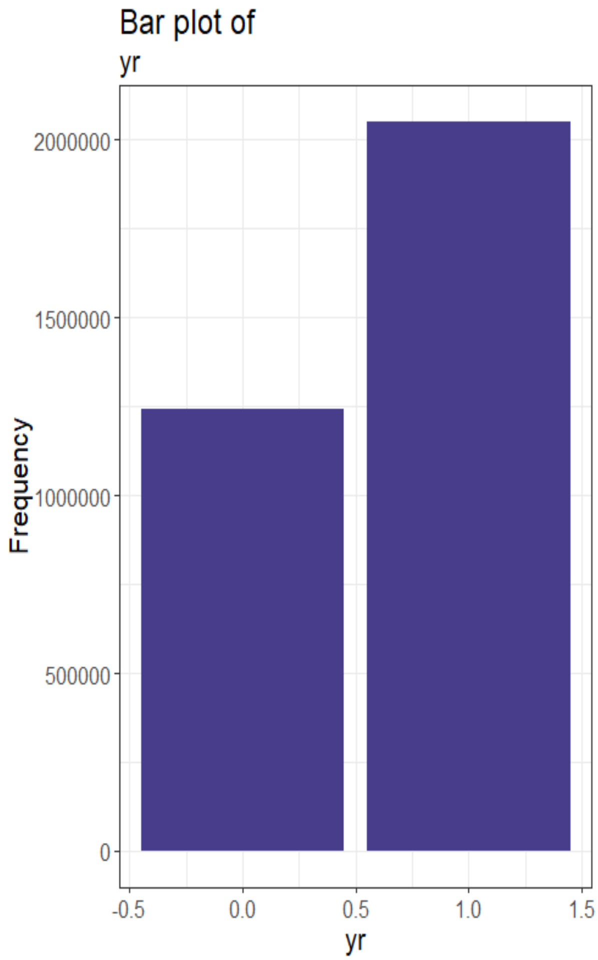
- In this dataset some of the variable are normalized such as temp,hum,windspeed.
- Some of the attributes such as season,year,month,holiday,weekday,workingday,weathersit are converted into factor variable,and then dummies are created for the variable for the purpose of effective model building

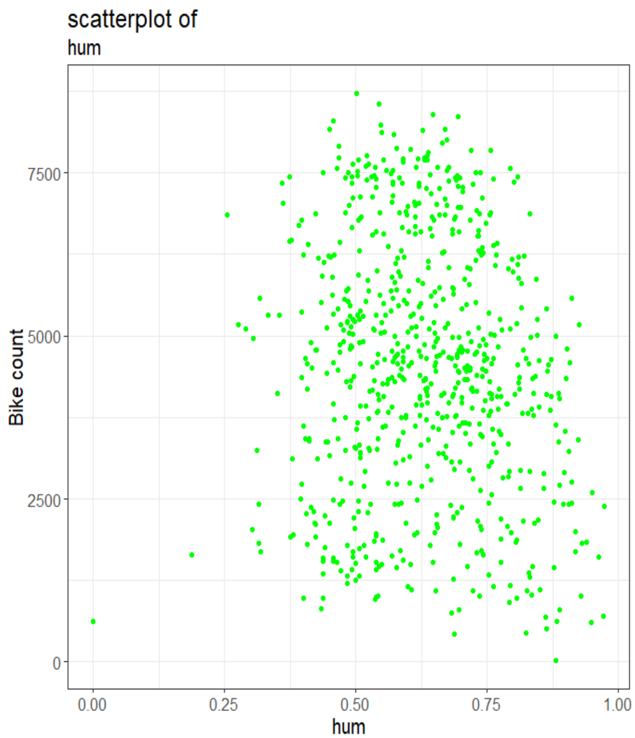
Visualization:

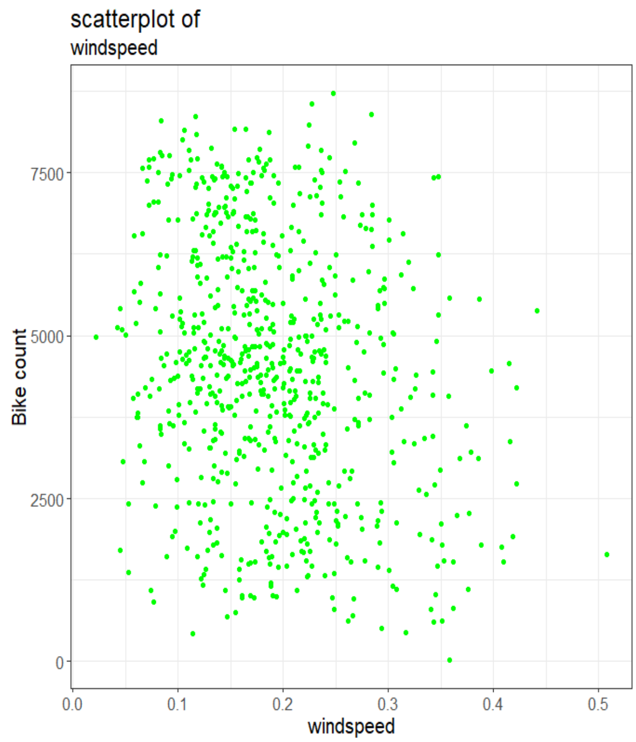
- Visualization is the pictorical representation of the data
- In R I has used ggplot library
- In python I has used matplotlibrary,seaborn library

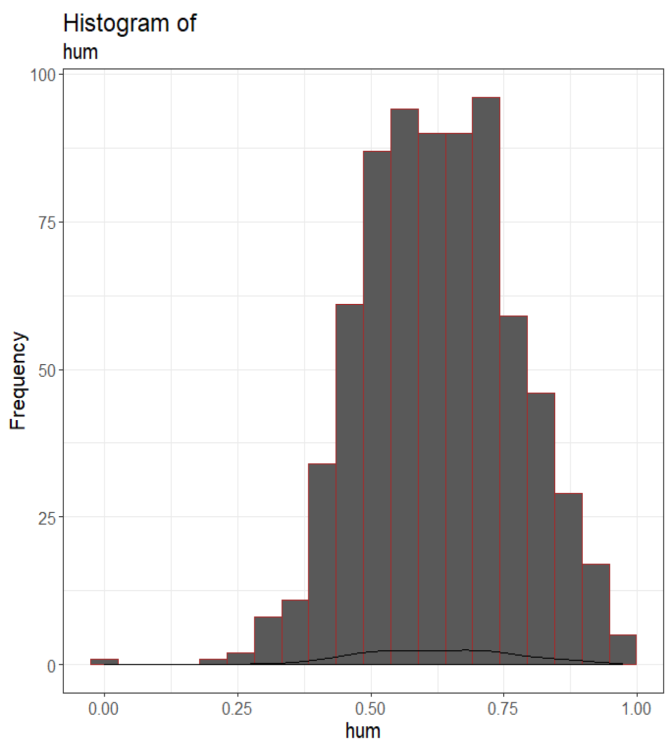
  In R:

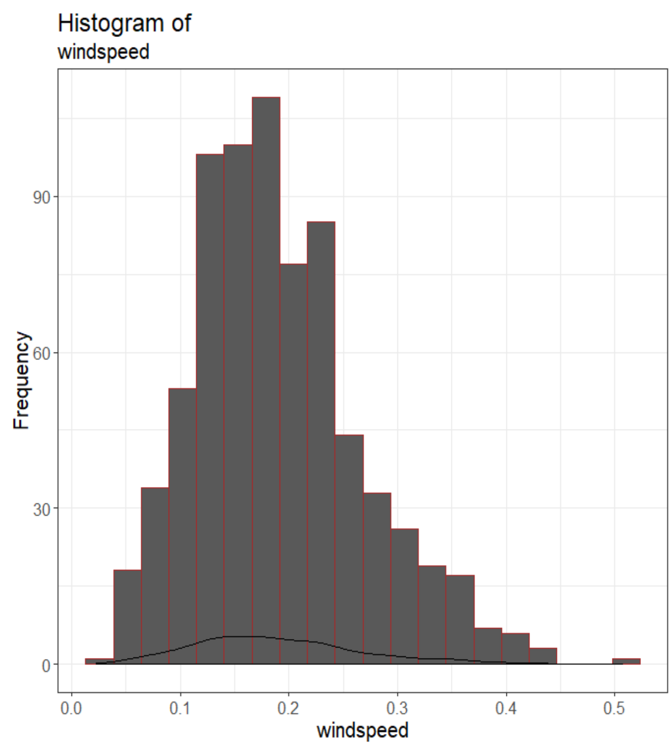




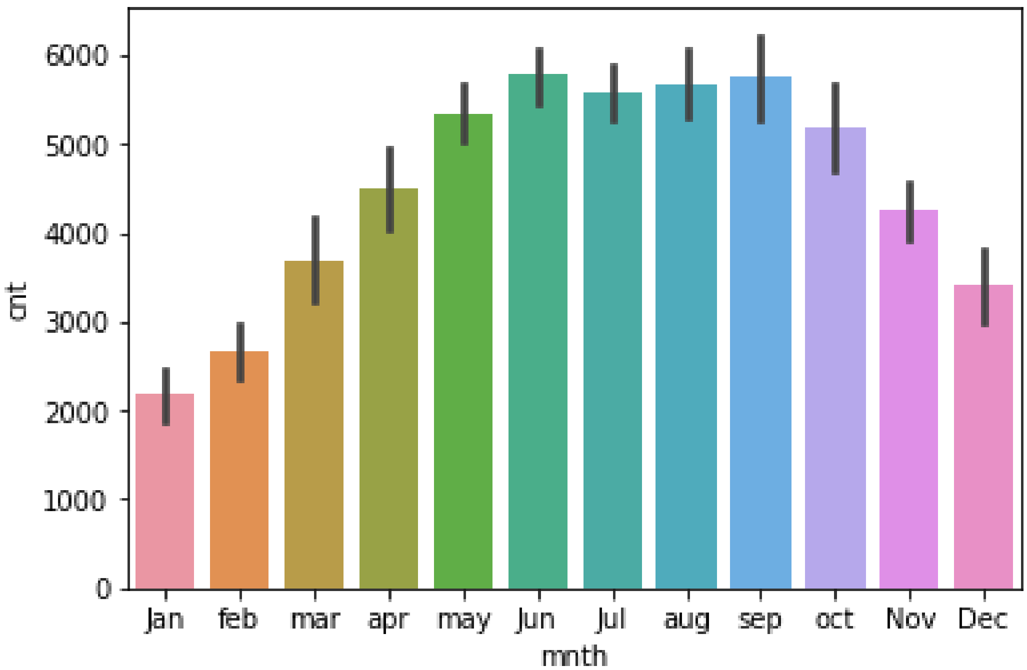

scatterplot of
hum
Bike count
hum


scatterplot of
windspeed
Bike count
windspeed

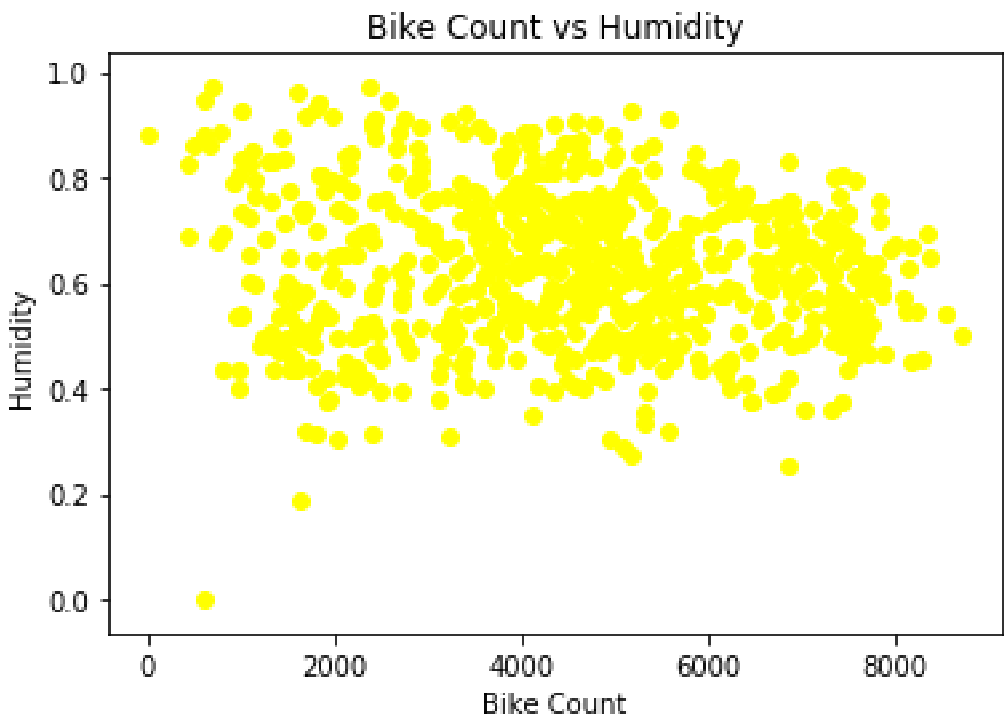

Histogram of
hum
Frequency
hum


Histogram of
windspeed
Frequency
windspeed

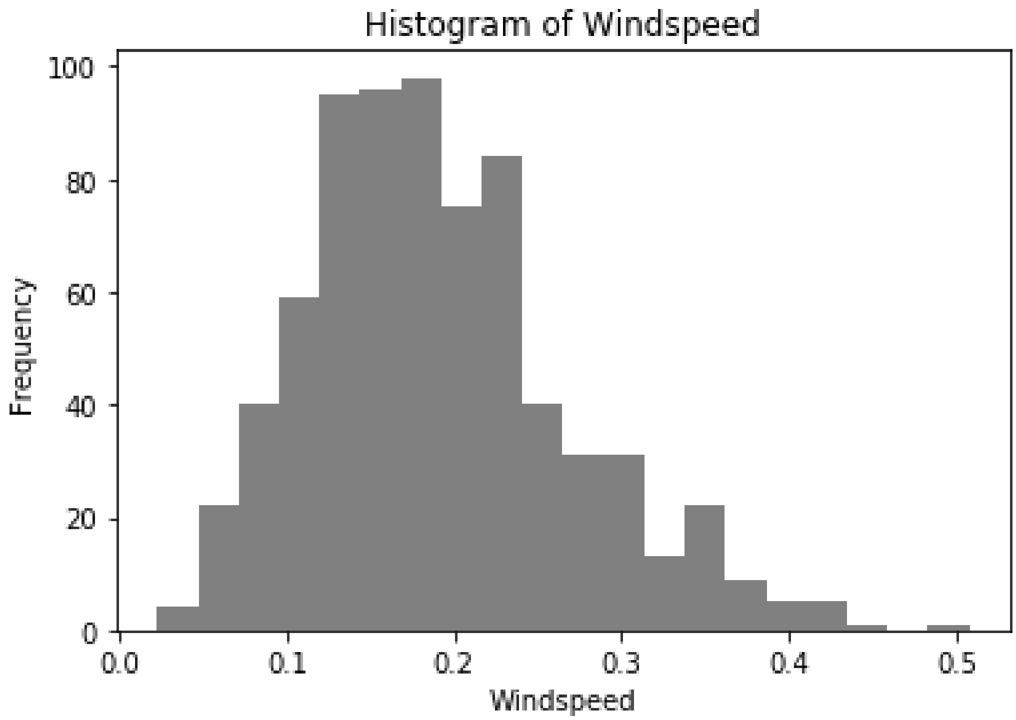
In PYTHON:



6000
5000
4000
3000
2000
1000
0
cnt
Jan
feb
mar
apr
may
Jun
Jul
aug
sep
oct
Nov
Dec
mnth



Bike Count vs Humidity
1.0
0.8
0.6
0.4
0.2
0.0
Humidity
0
2000
4000
6000
8000
Bike Count


Histogram of Windspeed
Frequency
0
20
40
60
80
100
0.0
0.1
0.2
0.3
0.4
0.5
Windspeed

Model Building:

SAMPLING:

For the pupose of checking the model we need to divide the dataset into test and train .In which one set is used for training the model and the trained model is applied to the test data further it get compared with actual vs predicted values. I has used 80% of is training data 20% of it is testing data.In R I has used create data partition function in caret library.In python I has used train_test_split function in the sklearn library

Linear Regression Model:

In this model,in R I has used lm function ,further I has introduced stepAIC function which was give the important variable. variable is reduced step by step by seeing P-value and vif .I had removed more than 10 variable which is of more significance.Finally I got R^2 value of 0.83 and 3star p_value.

Coefficients:

	Estimate	Std.Error	t value	Pr(>|t|)	
(Intercept)	1382.83	231.88	5.964	3.91e-09	***
yr1	1994.14	59.38	33.584	< 2e-16	***
workingday1	410.54	79.13	5.188	2.78e-07	***
temp	6117.63	179.97	33.993	< 2e-16	***
hum	-1824.20	299.74	-6.086	1.90e-09	***
windspeed	-2617.80	441.42	-5.930	4.74e-09	***
season2	788.21	73.67	10.700	< 2e-16	***
season4	1083.75	87.59	12.373	< 2e-16	***
mnth9	961.12	113.14	8.495	< 2e-16	***
mnth10	510.95	128.04	3.991	7.28e-05	***
weekday6	534.63	104.92	5.096	4.47e-07	***
weathersit2	-416.72	79.57	-5.237	2.16e-07	***
weathersit3	-1678.78	212.51	-7.900	1.08e-14	***

In Python,I had separated the dependent and independent variable with y and x respectively.Further it can sent into the model .Linear model is available in the library sklearn.statmodels.Finally it produce the R^2 value of around 0.86

Decision tree:

In R there is a function rpart in library (rpart) which is used to build the decision tree in the training dataset ,which inturn deployed to the x-test data to predict the result

In Python there is a function DecisionTreeRegressor in library (sklearn) which is used to build the decision tree in the training dataset ,which inturn deployed to the x-test data to predict the result

Random Forest:

In R there is a function RandomForest in library (Randomforest) which is used to build the random Forest in the training dataset ,which inturn deployed to the x-test data to predict the result

In Python there is a function RandomForest Regressor in library (sklearn.ensemble) which is used to build the RandomForest in the training dataset ,which inturn deployed to the x-test data to predict the result

XGBoost:

In R there is a function in library (caret) which is used to build the Xgbtree in the training dataset with control and tuning parameter ,which inturn deployed to the x-test data to predict the result

In Python there is a function XGBRegressor in library (xgboost) which is used to build the xgboost in the training dataset ,which inturn deployed to the x-test data to predict the result

SVR:

In R there is a function in library (e1071) which is used to build the SVR in the training dataset with control and tuning parameter ,which inturn deployed to the x-test data to predict the result

Error-Metrics:

For the purpose of checking the accuracy of the model we are going for error metrics there are various error metric for regression model such as Rootmean square error,mean absolute error,mean absolute percentage error.In this model I has taken two error metrics such as RMSE,MAPE.MAPE is used to calculate the percentage of error between the actual and predicted value.It will the error in term of percentage.RMSE is used to calculate the standard deviation of the prediction error.It is mainly employed in time series data so we can employ RMSE error_metrics in addition to that we can go for MAPE for the calculation of percentage of error

In R I had employed error metrics by function Regr.eval

In Python MAPE is computed manually,RMSE is calculated by using function mean squared error in the library sklearn.metrics

Prediction for test dataset:

In R ,the test dataset is enter into the SVR model, to predict the fare amount because its RMSE is lesser compared to other model

In Python, test dataset is enter into the XGBoost model, to predict the fare amount because its RMSE is lesser compared to other model

Finding &Conclusion:

- Humidity and windspeed weakens the bike count ,summer and winter season plays an vital role in determining bike count
- Weather play an important role in count as weakens in weathers Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist , Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
- Temperature play an important role in determining the Bike count
- Working day play an important role as more bike count as there will be more people using in regular basics
- 2012 Year also play an important role in determining the fare
- RMSE and MAPE is used to select the model,lower the value better the model
- Hence ,the freezed model(SVR model in R and Random Forest model in Python) is used to predict the Bike count of the test dataset
- The model is 88.89% accuracy in R, the model is 85.17% accuracy in python
- Thus,this model is good in predicting the bike Rental count for Bike rental business

Appendix

R

```
#######Bike count prediction##########

rm(list=ls())

getwd()

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

dim(data)

str(data)

names(data)

sum(is.na(data))

sum(duplicated(data$instant))

str(data)

############Changing variable to factor#######

names(data)

num=c('season','yr','mnth','holiday','weekday','workingday','weathersit')

for(i in num){

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

}

str(data)

################Visualisation#############

library(ggplot2)

##############factor data########

fac=c('season','yr','mnth','holiday','weekday','workingday','weathersit')

for(i in (1:length(fac))){
```

```
assign(paste0('f',i),ggplot(data,aes_string(x=fac[i],y='cnt'))+geom_bar(stat='identity',fill='darksl
ateblue')+xlab(fac[i])+ylab("Frequency")+
   ggtitle("Bar plot of",fac[i])+theme_bw()+theme(text=element_text(size = 15)))
}
gridExtra::grid.arrange(f1,f2,ncol=2)
gridExtra::grid.arrange(f3,f4,ncol=2)
gridExtra::grid.arrange(f5,f6,f7,ncol=3)

#########Scatter plot########
no=c('temp','atemp','hum','windspeed')
for(i in (1:length(no))){
  assign(paste0('n',i),
ggplot(data,aes_string(x=no[i],y='cnt'))+geom_point(color='green')+xlab(no[i])+ylab("Bike
count")+
   ggtitle("scatterplot of",no[i])+theme_bw()+theme(text=element_text(size = 15)))
}
gridExtra::grid.arrange(n1,n2,ncol=2)
gridExtra::grid.arrange(n3,n4,ncol=2)



###########Histogram###########
no=c('temp','atemp','hum','windspeed')
for(i in (1:length(no))){
  assign(paste0('h',i),
ggplot(data,aes_string(x=no[i]))+geom_histogram(color='Brown',bins=20)+xlab(no[i])+ylab("Fr
equency")+
        ggtitle("Histogram of",no[i])+theme_bw()+theme(text=element_text(size =
15))+geom_density())
```

```
}
gridExtra::grid.arrange(h1,h2,ncol=2)
gridExtra::grid.arrange(h3,h4,ncol=2)


###################Boxplot#############
no=c('temp','atemp','hum','windspeed')
for(i in (1:length(no))){
  assign(paste0('b',i),
ggplot(data,aes_string(y=no[i]))+geom_boxplot(color='red',fill='orange')+xlab(no[i])+ylab("Fre
quency")+
        ggtitle("boxplot of",no[i])+theme_bw()+theme(text=element_text(size = 15)))
}
gridExtra::grid.arrange(b1,b2,ncol=2)
gridExtra::grid.arrange(b3,b4,ncol=2)


######Outlier Detection and Removal############
no=c('temp','atemp','hum','windspeed')
for (i in no)
{
  box=data[,i][data[,i]%in% boxplot.stats(data[,i])$out]
  data[,i][data[,i]%in% box]=NA
}
data$cnt=ifelse(data$cnt>100,data$cnt,NA)
View(data)
sum(is.na(data))
data=na.omit(data)
#######correlation#############
library(corrgram)
```

```
cor(data[,no])

corrgram(data[,no],order=F,upper.panel = panel.pie,text.panel = panel.text,main="correlation
plot")

names(data)

data=data[,-11]

##############Creating Dummies###########

str(data)

d1=data.frame(model.matrix(~season,data))

d1=d1[,-1]

d2=data.frame(model.matrix(~mnth,data))

d2=d2[,-1]

d3=data.frame(model.matrix(~weekday,data))

d3=d3[,-1]

d4=data.frame(model.matrix(~weathersit,data))

d4=d4[,-1]

dim(data)

str(data)

data=data[,-c(1:2)]

str(data)

names(data)

data=data[,-c(11,12)]

dim(data)

data=cbind(data,d1,d2,d3,d4)

str(data)

data=data[,-c(1,3,5,7)]

##############Sample #############

library(caret)
```

```
sam=createDataPartition(data$cnt,p=0.80,list=F)
train=data[sam,]
test=data[-sam,]
names(test)
################Linear Regression############
lm1=lm(cnt~.,data)
summary(lm1)
library(car)
vif(lm1)
library(MASS)
step=stepAIC(lm1)
lm2=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 + mnth3 + mnth4 + mnth5 + mnth6 + mnth8 + mnth9 +
        mnth10 + weekday1 + weekday6 + weathersit2 + weathersit3,data)
summary(lm2)
vif(lm2)
##################Month4#################
lm3=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
    season4 + mnth3 + mnth5 + mnth6 + mnth8 + mnth9 +
    mnth10 + weekday1 + weekday6 + weathersit2 + weathersit3,data)
summary(lm3)
vif(lm3)
##################Month6#################
lm4=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 + mnth3 + mnth5 + mnth8 + mnth9 +
        mnth10 + weekday1 + weekday6 + weathersit2 + weathersit3,data)
summary(lm4)
```

```r
vif(lm4)
#################Month5#################
lm5=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 + mnth3 + mnth8 + mnth9 +
        mnth10 + weekday1 + weekday6 + weathersit2 + weathersit3,data)
summary(lm5)
#################Weekday1#################
lm6=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 + mnth3 + mnth8 + mnth9 +
        mnth10 +  weekday6 + weathersit2 + weathersit3,data)
#################Month8#################
lm7=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 + mnth3 + mnth9 +
        mnth10 +  weekday6 + weathersit2 + weathersit3,data)
summary(lm7)
#################Month3#################
lm8=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 + season3 +
        season4 +  mnth9 +
        mnth10 +  weekday6 + weathersit2 + weathersit3,data)
summary(lm8)
#################Season3#################
lm9=lm(cnt ~ yr + workingday + temp + hum + windspeed + season2 +
        season4 + mnth9 +
        mnth10 +  weekday6 + weathersit2 + weathersit3,data)
summary(lm9)
vif(lm9)
names(test)
```

```
pr=predict(lm9,test[,-7])
library(DMwR)
regr.eval(test$cnt,pr,stats = c('rmse','mape'))
####Accuracy=83.35%
####RMSE=735.02
####MAPE=16.65%
################Decision Tree#########
library(rpart)
tree_mod=rpart(cnt~.,train,method="anova")
summary(tree_mod)
pre1=predict(tree_mod,test[,-7])
regr.eval(test$cnt,pre1,stat=c("rmse","mape"))
####Accuracy=78.8%
####RMSE=851.36
####MAPE=21.20%
###############Random Forest############
library(randomForest)
forest_mod=randomForest(cnt~.,train,importance=T,ntree=100)
summary(forest_mod)
pre2=predict(forest_mod,test[,-7])
regr.eval(test$cnt,pre2,stat=c("rmse","mape"))
####Accuracy=83.76%
####RMSE=633.14
####MAPE=16.24%
###################SVR#################
library(e1071)
svr_mod=svm(cnt~.,data,type='eps-regression')
```

```
summary(svr_mod)
pre3=predict(svr_mod,test[,-7])
out=cbind(test[,-7],pre3)
write.csv(out,"output for sample data.csv")
regr.eval(test$cnt,pre3,stat=c("rmse","mape"))
####Accuracy=88.89%
####RMSE=457.67
####MAPE=11.11%
#################XGboost###########
library(caret)
control=trainControl(method='cv',number=5,savePredictions = T,classProbs = T)
paragrid=expand.grid(eta=0.1,gamma=1,max_depth=3,nrounds=100,colsample_bytree=0.7,
              min_child_weight=2,subsample=0.5)
model=train(cnt~.,data=train,method='xgbTree',Control=control,tuneGrid=paragrid)
pre5=predict(model,test[,-7])
regr.eval(test$cnt,pre5,stat=c("rmse","mape"))
####Accuracy=87.0%
####RMSE=594.50
####MAPE=13.02%
```

Python:

```
###########Library###########

import os

import pandas as pd

import matplotlib.pyplot as pt

import numpy as np

os.chdir("C:/Users/welcome/Desktop/Project-2")

os.getcwd()

data=pd.read_csv("day.csv")

data.dtypes

data.isnull().sum()

########Dropping the unimportant variable#####

data=data.drop("casual",axis=1)
```

```
data=data.drop("registered",axis=1)

data=data.drop("instant",axis=1)

data=data.drop("dteday",axis=1)

data.shape

###########Checking for missing values############

data.isnull().sum()

#############Changing into proper datatype#######

num=['season','yr','mnth','holiday','weekday','workingday','weathersit']

for i in num:
   data.loc[:,i]=data.loc[:,i].astype("object")

   data.dtypes

data['yr']=np.where(data['yr']==0,'2011','2012')

data['season']=np.where(data['season']==1,'spring',
   np.where(data['season']==2,'summer',np.where(data['season']==3,'fall',
         'winter')))
data['mnth']=np.where(data['mnth']==1,'Jan',
```

```
np.where(data['mnth']==2,'feb',
  np.where(data['mnth']==3,'mar',
    np.where(data['mnth']==4,'apr',
      np.where(data['mnth']==5,'may',
        np.where(data['mnth']==6,'Jun',
          np.where(data['mnth']==7,'Jul',
            np.where(data['mnth']==8,'aug',
              np.where(data['mnth']==9,'sep',
                np.where(data['mnth']==10,'oct',
                  np.where(data['mnth']==11,'Nov','Dec'))))))))))



###############visualisation############

##########Histogram for numeric attribute###########

pt.hist(data.temp,bins=20,color='red')

pt.xlabel('Temperature')

pt.ylabel('Frequency')

pt.title('Histogram of Temperature')
###########
pt.hist(data.atemp,bins=20,color='blue')
```

```
pt.xlabel('Actual Temperature')

pt.ylabel('Frequency')

pt.title('Histogram of Actual Temperature')
########
pt.hist(data.windspeed,bins=20,color='grey')

pt.xlabel('Windspeed')

pt.ylabel('Frequency')

pt.title('Histogram of Windspeed')
###########
pt.hist(data.hum,bins=20,color='yellow')

pt.xlabel('Humidity')

pt.ylabel('Frequency')

pt.title('Histogram of Humidity')

############Scatterplot for numeric attibute#########

pt.scatter(x=data.cnt,y=data.temp,color='red')

pt.ylabel('Temperature')
```

```
pt.xlabel('Bike Count')

pt.title('Bike Count vs Temperature')
#########
pt.scatter(x=data.cnt,y=data.atemp,color='green')

pt.ylabel('Actual Temperature')

pt.xlabel('Bike Count')

pt.title('Bike Count vs  Actual Temperature')
########
pt.scatter(x=data.cnt,y=data.windspeed,color='blue')

pt.ylabel('Windspeed')

pt.xlabel('Bike Count')

pt.title('Bike Count vs Windspeed')
#########
pt.scatter(x=data.cnt,y=data.hum,color='Yellow')

pt.ylabel('Humidity')

pt.xlabel('Bike Count')
```

```
pt.title('Bike Count vs Humidity')

##############Barplot for Factor Variable########

num=['season','yr','mnth','holiday','weekday','workingday','weathersit']

import seaborn as sn

sn.barplot(x="season",y="cnt",data=data)

sn.barplot(x="yr",y="cnt",data=data)

sn.barplot(x="mnth",y="cnt",data=data)

sn.barplot(x="holiday",y="cnt",data=data)

sn.barplot(x="weekday",y="cnt",data=data)

sn.barplot(x="workingday",y="cnt",data=data)

sn.barplot(x="weathersit",y="cnt",data=data)

##########Boxplot for numeric variable######

sn.boxplot(y=data.temp)

sn.boxplot(y=data.atemp)
```

```
sn.boxplot(y=data.hum)


sn.boxplot(y=data.windspeed)


sn.boxplot(y=data.cnt)



data=data.drop(data[(data.cnt <100)].index,axis=0)


###########Remove outliers######
data.dtypes
no1=['temp','atemp','windspeed','hum','cnt']
for i in no1:
    q75,q25=np.percentile(data.loc[:,i],[75,25])
    iqr=q75-q25
    mi=q25-(1.5*iqr)
    ma=q75+(1.5*iqr)
    data.loc[data.loc[:,i]<mi,:i]=np.nan
    data.loc[data.loc[:,i]>ma,:i]=np.nan

data.isnull().sum()


data=data.dropna()


###############Correlation#############
```

```
cor=data.loc[:,no1]

co_mat=cor.corr().round(2)

data=data.drop('atemp',axis=1)

#########Dummies creation##########

num=['season','yr','mnth','holiday','weekday','workingday','weathersit']

for i in num:
   tem=pd.get_dummies(data[i],prefix=i)
   data=data.join(tem)

data.dtypes

data=data.drop('season',axis=1)

data=data.drop('yr',axis=1)

data=data.drop('mnth',axis=1)

data=data.drop('holiday',axis=1)

data=data.drop('weekday',axis=1)

data=data.drop('workingday',axis=1)
```

```
data=data.drop('weathersit',axis=1)

############Sampling###########

x=data.drop('cnt',axis=1)

y=data.iloc[:,3].values

from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=0)

x_tr=x_train

x_te=x_test

##################Linear model##########

import statsmodels.api as sm

mod_1=sm.OLS(y_train,x_train).fit()

mod_1.summary()

max(mod_1.pvalues)
```

```
x_train=x_train.drop('mnth_Jul',axis=1)

mod_2=sm.OLS(y_train,x_train).fit()

mod_2.summary()

max(mod_2.pvalues)

x_train=x_train.drop('mnth_feb',axis=1)

mod_3=sm.OLS(y_train,x_train).fit()

mod_3.summary()

max(mod_3.pvalues)

x_train=x_train.drop('mnth_Jan',axis=1)

mod_4=sm.OLS(y_train,x_train).fit()

mod_4.summary()

max(mod_4.pvalues)

x_train=x_train.drop('weekday_1.0',axis=1)

mod_5=sm.OLS(y_train,x_train).fit()
```

```
mod_5.summary()

max(mod_5.pvalues)

x_train=x_train.drop('weekday_2.0',axis=1)

mod_6=sm.OLS(y_train,x_train).fit()

mod_6.summary()

max(mod_6.pvalues)

x_train=x_train.drop('weekday_4.0',axis=1)

mod_7=sm.OLS(y_train,x_train).fit()

mod_7.summary()

max(mod_7.pvalues)

x_train=x_train.drop('weekday_3.0',axis=1)

mod_8=sm.OLS(y_train,x_train).fit()

mod_8.summary()
```

```
max(mod_8.pvalues)

x_train=x_train.drop('mnth_Dec',axis=1)

mod_9=sm.OLS(y_train,x_train).fit()

mod_9.summary()

max(mod_9.pvalues)

x_train=x_train.drop('mnth_Nov',axis=1)

mod_10=sm.OLS(y_train,x_train).fit()

mod_10.summary()

max(mod_10.pvalues)

x_train=x_train.drop('weekday_5.0',axis=1)

mod_11=sm.OLS(y_train,x_train).fit()

mod_11.summary()

max(mod_11.pvalues)

x_train.columns
```

```
x_test=x_test.loc[:,['temp', 'hum', 'windspeed', 'season_fall', 'season_spring',
       'season_summer', 'season_winter', 'yr_2011', 'yr_2012', 'mnth_Jun',
       'mnth_apr', 'mnth_aug', 'mnth_mar', 'mnth_may', 'mnth_oct', 'mnth_sep',
       'holiday_0.0', 'holiday_1.0', 'weekday_0.0', 'weekday_6.0',
       'workingday_0.0', 'workingday_1.0', 'weathersit_1.0', 'weathersit_2.0',
       'weathersit_3.0']]

pr1=mod_11.predict(x_test)

from sklearn.metrics import mean_squared_error,r2_score

rmse=np.sqrt(mean_squared_error(y_test,pr1))

r2=(r2_score(y_test,pr1))

def mape_error(acu_val,pred_val):
    mape=np.mean(np.abs((acu_val-pred_val)/acu_val))*100
    return mape

mape_error(y_test,pr1)

######Rmse=738
######MAPE=16.10%
######Accuracy=83.90%
####r2=86.66%
```

```
################Decision tree###########

from sklearn import tree

tree_mod=tree.DecisionTreeRegressor(random_state=0).fit(x_tr,y_train)

pr3=tree_mod.predict(x_te)

rmse1=np.sqrt(mean_squared_error(y_test,pr3))

mape_error(y_test,pr3)

r2_1=(r2_score(y_test,pr3))

######Rmse=941
######MAPE=19.78%
######Accuracy=80.22%
####r2=78.37%

################Random Forest###########

from sklearn.ensemble import RandomForestRegressor

for_mod=RandomForestRegressor().fit(x_tr,y_train)

pr2=for_mod.predict(x_te)
```

```
rmse2=np.sqrt(mean_squared_error(y_test,pr2))

mape_error(y_test,pr2)

r2_2=(r2_score(y_test,pr2))



######Rmse=680
######MAPE=14.83%
######Accuracy=85.17%
##########r2=88.70%



##############XGboost########

import xgboost

xg=xgboost.XGBRegressor(n_estimators=100,learning_rate=0.05,gamma=0,subsample=0.50,
                        colsample_bytree=1,max_depth=4).fit(x_tr,y_train)

pr3=xg.predict(x_te)

rmse3=np.sqrt(mean_squared_error(y_test,pr3))

mape_error(y_test,pr3)
```

```
r2_3=(r2_score(y_test,pr3))


######Rmse=617.93

######MAPE=13.52%

######Accuracy=86.48%

##########r2=90.68%


x_te['cnt']=pr3


x_te.to_csv("output for sample data in python .csv")
```