


# Prediction of Bike Rental count.



PRUDHVI KUMAR K  
20th September, 2019

# Contents

<b>1 Introduction.....</b>	<b>3</b>
1.1 Problem Statement.....	3
1.2 Data.....	3
<b>2 Methodology.....</b>	<b>4</b>
2.1.1 PreProcessing.....	4
2.1.2 Missing value and outlier analysis.....	5
2.1.4 Feature Selection.....	8
<b>3 Modeling.....</b>	<b>12</b>
3.1.1 Model Selection.....	12
3.1.2 Regression.....	12
3.1.3 Decision tree.....	13
<b>4 Conclusion.....</b>	<b>17</b>
4.1.1 Model Evaluation.....	17
4.1.2 Mean absolute percentage error (MAPE).....	17
4.1.3 Model Selection.....	18
<b>Appendix A - Extra</b>	
Figures.....	20



Complete R File.....	26
Complete python file.....	31

## Chapter-1 Introduction

### 1.1 Problem Statement

The objective of this project is to prediction of bike rental count, based on the Environmental and seasonal changes.

### 1.2 Data

The details of data attributes in the dataset are as follows.

The sample of data is given below.(Data represented here is random data.)

---

Instant	date	season	year	month	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed	casual	reg	count
1	1/1/2011	1	0	1	0	6	0	2	0.223	0.223	2	0.123	0.123	0.223	2555
2	2/1/2011	2	0	1	0	0	0	2	1.223	0.234	4	1.234	2.222	1.223	2345
3	3/2/2011	3	0	1	0	1	1	1	0.223	0.223	4	2.234	1.234	2.234	33
4	4/1/2011	4	0	1	1	2	1	1	1.223	0.445	3	3.234	2.345	0.234	3345
5	5/1/2011	4	0	1	1	3	1	1	0.345	0.335	5	0.234	0.223	0.345	3345

---

As you can see below we have the following 15 variables, using which we have to correctly predict the bike rental count.

The predictor variables.

**Instant date season year month holiday weekday workingday  
weathersit temp atemp hum windspeed casual reg**

---

## Chapter 2

### Methodology

#### 2.1 Pre Processing

The work starts with data preprocessing, means looking at the data to get insights. 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 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 distributions of the variable by QQ-normality graph or histogram. Histogram is the best chart to represent the data distribution, later the data is normalized or standardized, according to the model needs.

Data preprocessing is the tedious task, we need to focus more on this part to reduce the model complexity. Before feeding the data to model, we must preprocess the data, which has various stages like missing value analysis, impute the missing values, outlier check, normality check, sampling. Then the data is subjected to model training and prediction.

After completing the model prediction, the machine learning prediction system is ready for deployment. Project deployment refers to , preparing the machine learning environment to the front end users.

Preprocessing is done in both R and python programming.

## 2.1.1 Missing value analysis.

Missing value analysis helps address several concerns caused by incomplete data. If cases with missing values are systematically different from cases without missing values, the results can be misleading. Also, missing data may reduce the precision of calculated statistics because there is less information than originally planned. Another concern is that the assumptions behind many statistical procedures are based on complete cases, and missing values can complicate the theory required.

The Missing Value Analysis procedure performs three primary functions:

Describes the pattern of missing data. Where are the missing values located? How extensive are they? Do pairs of variables tend to have values missing in multiple cases? Are data values extreme? Are values missing randomly?

Estimates means, standard deviations, covariances, and correlations for different missing value methods: listwise, pairwise, regression, or EM (expectation-maximization). The pairwise method also displays counts of pairwise complete cases.

Fills in (imputes) missing values with estimated values using regression or EM methods; however, multiple imputation is generally considered to provide more accurate results.

In our case, for bike rental the data seems to be correct without missing values, so there is no missing values in the data . so looking forward to perform an outlier check.

## 2.1.2 outlier analysis

The outlier analysis play an important role in the preprocessing, outliers are the data points , which are located above or below the mean of the dataset. Actually they are good informational data points, but not good in statistical point of view.

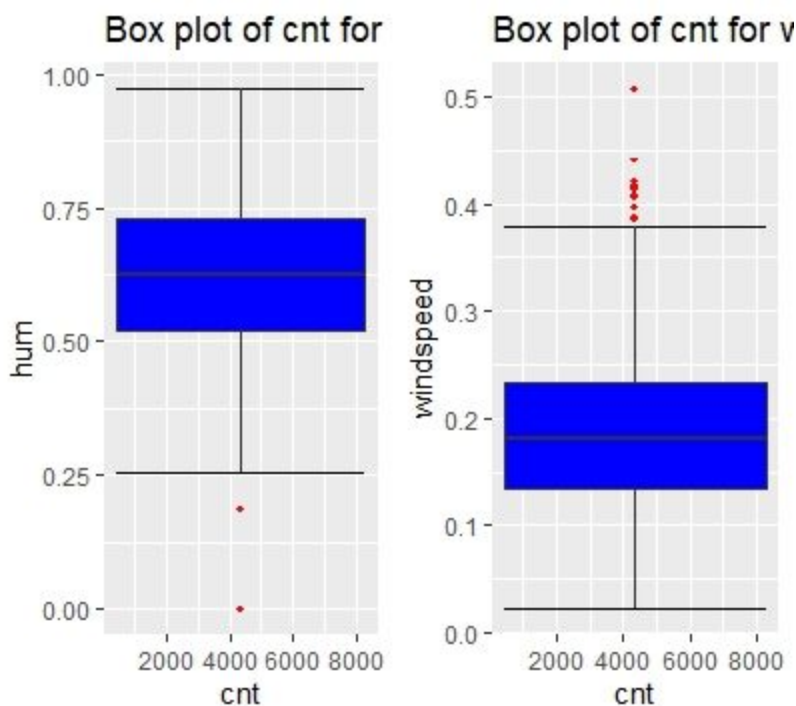
At high end the effective way to analyze the outliers is the graphical method. Which is known as box plot method.

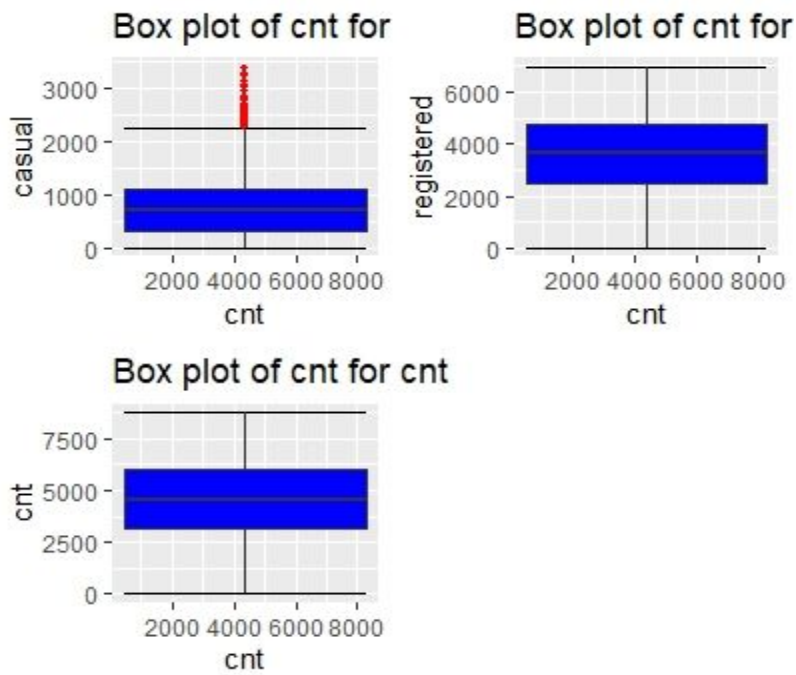
We can plot box plot in both R and Python, both have various options to build a boxplot. Like we have SEABORN, MATPLOTLIB PYPLOT in python and GGLOT and BOXPLOT. STATS in R.

After detecting outliers , at high end there are two levels of operation either delete the outliers or impute them. For imputing check missing value analysis , it got many methods like MEAN, MEDIAN, KNN...etc.

For the bike rental data,

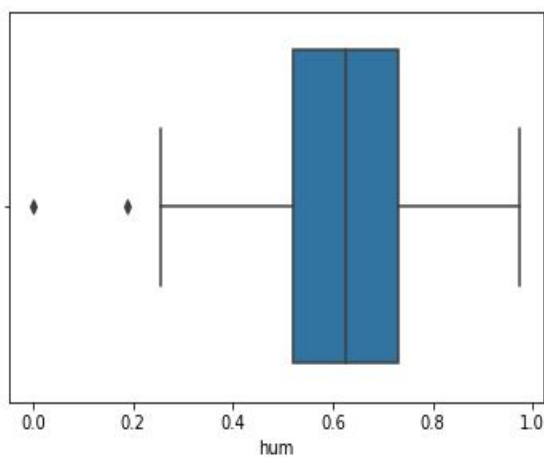
The box plot analysis for R is shown in below figure. Only for numeric variables.



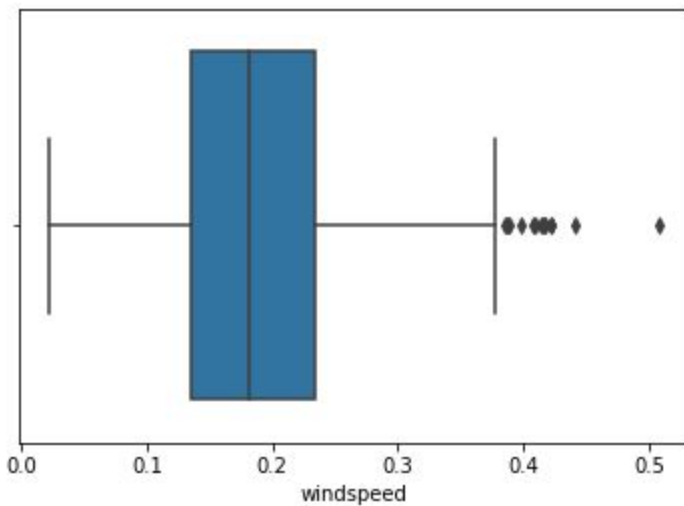


From the above analysis, we have found the outliers , which is spotted red in the box plot chart. Found negative outlier in hum and positive outlier in casual and wind speed.

Few plots form python, SEABORN.

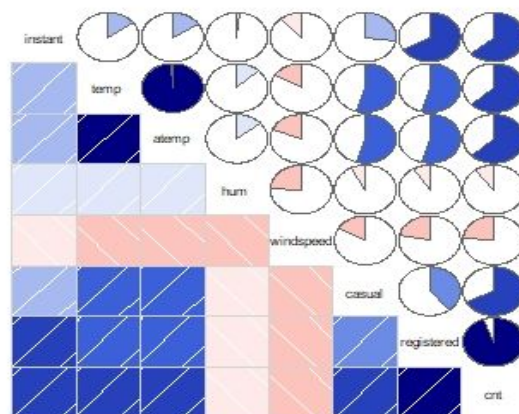






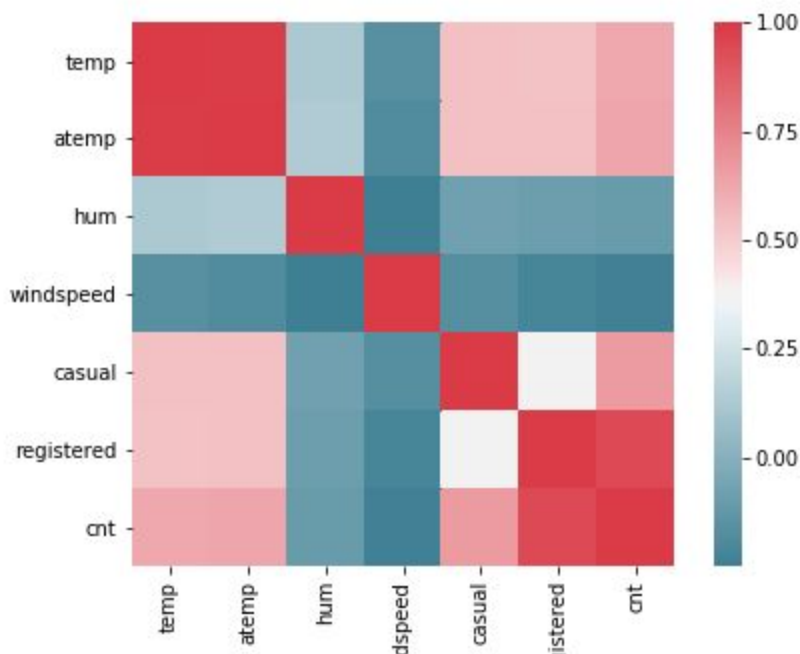
### 2.1.3 Feature Selection

Before performing any type of modelling we need to assess the importance of each predictor variable in our analysis. 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 doing that. Below we have used the **correlation plot** for numerical features and **chi<sup>2</sup> test** for categorical variables. This method is implemented in both R and Python languages.



R correlation plot.

## Python correlation plot



From the above plots, “atemp ” is highly correlated with temp and hence it must be removed.

## Chi^2 test for categorical features.

A chi-squared test, also written as  $\chi^2$  test, is any statistical hypothesis test where the sampling distribution of the test statistic is a chi-squared distribution when the null hypothesis is true. Without other qualification, 'chi-squared test' is often used as short for Pearson's chi-squared test. The chi-squared test is used to determine whether there is a significant difference between the expected frequencies and the observed frequencies in one or more categories.

It is purely depend on the p-value it has, if the variable is less than 0.05 p, then it is considered as the important variable ,which is contributing to predict the target variable.

Seperate the categorical features.

### R code for variable importance.

**# chi^2 test for categorical variables.**

```
for(i in 1:8 ){  
  print(names(factor_data)[i])  
  print(chisq.test(table(factor_data[,i])))  
}
```

### OUTPUT

```
[1] "dteday"  
Chi-squared test for given probabilities  
data:  table(factor_data[, i])    X-squared = 0, df = 730,  
p-value = 1  
[1] "season"  
Chi-squared test for given probabilities  
data:  table(factor_data[, i])  
X-squared = 0.29959, df = 3, p-value = 0.9601  
[1] "yr"  
Chi-squared test for given probabilities  
data:  table(factor_data[, i])    X-squared = 0.001368, df =  
1, p-value = 0.9705  
[1] "mnth"  
Chi-squared test for given probabilities  
data:  table(factor_data[, i])  
X-squared = 0.44186, df = 11, p-value = 1  
[1] "holiday"
```

```

Chi-squared test for given probabilities
data: table(factor_data[, i]) X-squared = 649.41, df = 1,
p-value < 2.2e-16
[1] "weekday"
Chi-squared test for given probabilities
data: table(factor_data[, i]) X-squared = 0.016416, df =
6, p-value = 1
[1] "workingday"
Chi-squared test for given probabilities
data: table(factor_data[, i])
X-squared = 98.989, df = 1, p-value < 2.2e-16
[1] "weathersit"
Chi-squared test for given probabilities
data: table(factor_data[, i]) X-squared = 400.95, df = 2,
p-value < 2.2e-16

```

By this , the conclusion is “weathersit” “workingday” “holiday ” are the features contributing a lot to target variable.

By this , excluding the variables or features which are not wanting.

## Python code for chi^2

### # chi^2 test

```

catnames=["season","mnth","yr","holiday","weekday","working
day","weathersit"]
from scipy.stats import chi2_contingency
for i in catnames:
    print(i)

chi2,p,dof,ex=chi2_contingency(pd.crosstab(bikedata['cnt'],
bikedata[i]))          (print p)

```

---

## 2.1.4 Feature scaling

Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

We can visualize the data in both R and Python to have a quick distribution at a glance.

The bike rental data is a normalized one.

## Chapter 3

### 3.1 Data Modelling

#### 3.1.1 Model selection

Based on the target variable or feature we select the model. Here in our case in bike rental data, the target variable is continuous, so the model will be the Regression model.

#### 3.1.2 Regression

Use regression analysis to describe the relationships between a set of independent variables and the dependent variable. Regression analysis produces a regression equation where the coefficients represent the relationship between each independent variable and the dependent variable.

Regression Coefficient is the numerical or constant quantity in a regression equation which attempts to model the relationship between two or more

variables and a response variable by fitting a linear equation to observe the data.

Here in our case, I have applied both the DECISION TREE REGRESSOR and RANDOM FOREST REGRESSOR for R.

For python RANDOM FOREST REGRESSOR is implemented.

### 3.1.3 Decision trees

A predictive model based on a branching series of Boolean tests Can be used for classification and regression. There are a number of different types of decision trees that can be used in Machine learning algorithms. This is easy to understand by the business user.

#### R (Decision tree regression model )

Output for trained model.

```
node), split, n, deviance, yval
  * denotes terminal node
1) root 584 2093759000 4398.342
2) registered< 2914.5 184 173752100 2264.255
4) registered< 2178 118 42997320 1692.153
8) registered< 1347.5 43 6347475 1082.814 *
9) registered>=1347.5 75 11530600 2041.507 *
5) registered>=2178 66 23082440 3287.106 *
3) registered>=2914.5 400 696532800 5380.022
6) registered< 4336 221 102046500 4451.285
12) casual< 693 111 20040700 3978.090 *
13) casual>=693 110 32071030 4928.782 *
  7) registered>=4336 179 168510300 6526.676
14) casual< 839.5 63 17231440 5547.048 *
15) casual>=839.5 116 57983860 7058.716 *
```

## > summary of trained model

Call:

```
rpart(formula = cnt ~ ., data = train, method = "anova")
n= 584
```

	CP	nsplit	rel error	xerror	xstd
1	0.58434342	0	1.00000000	1.00515431	0.047470361
2	0.20345031	1	0.41565658	0.47764421	0.021104101
3	0.05142537	2	0.21220627	0.23940663	0.012479904
4	0.04455860	3	0.16078090	0.17862950	0.010861623
5	0.02384932	4	0.11622230	0.15955632	0.010534008
6	0.01199720	5	0.09237298	0.11623032	0.007126761
7	0.01000000	6	0.08037578	0.09851006	0.006629585

Variable importance

registered	casual	temp	hum	weathersit
windspeed	workingday			
55	20	14	4	
4	2	1		

Few predicted test cases of decision tree.

```
> predictions_tree
```

	4	5	8	11	18	19
30	42	49	53	54	60	
2041.507	2041.507	1082.814	1082.814	1082.814	2041.507	
1082.814	2041.507	3287.106	2041.507	2041.507	2041.507	
	66	68	77	81	88	90
91	94	96	114	131	134	
2041.507	2041.507	3287.106	3287.106	2041.507	2041.507	
2041.507	3287.106	3287.106	3287.106	3978.090	3287.106	

## R( Random forest)

> #Visualize some rules

```
> rules[1:2,]  
[1] "X[,3] %in% c('2','3') & X[,4]<=0.27125 &  
X[,4]<=0.214402 & X[,5]<=0.789855 & X[,7]<=269.5"  
[2] "X[,3] %in% c('2','3') & X[,4]<=0.27125 &  
X[,4]>0.214402 & X[,5]<=0.789855 & X[,7]<=269.5 &  
X[,8]<=1855.5"  
> #Make rules more readable:  
> readrules = presentRules(rules, colnames(train))  
> readrules[1:2,]  
[1] "weathersit %in% c('2','3') & temp<=0.27125 &  
temp<=0.214402 & hum<=0.789855 & casual<=269.5"  
[2] "weathersit %in% c('2','3') & temp<=0.27125 &  
temp>0.214402 & hum<=0.789855 & casual<=269.5 &  
registered<=185
```

## Python (Random forest)

Random forest is an ensemble that consists of many decision trees

The method combines Breiman's "bagging" idea and the random selection of features. Outputs the class that is the mode of the class's output by individual trees. Mean for regression. Can be used for classification and regression.

### Python model output.

```
RandomForestRegressor(bootstrap=True, criterion='mse',  
max_depth=None,  
max_features='auto',
```



```
max_leaf_nodes=None,  
min_impurity_decrease=0.0,  
min_impurity_split=None,  
min_samples_leaf=1,  
min_samples_split=2,  
min_weight_fraction_leaf=0.0,  
n_estimators=20,  
n_jobs=None, oob_score=False,  
random_state=None,  
verbose=0, warm_start=False)
```

### Predictions of random forest.

```
4045.6 6990.9 4456.15 4191. 5365.15 1996.75 1860.7 5784.1 4284.6  
4551.3 1625.2 4565.85 6951.35 3704.95 6156.25 1472.8 6285. 2259.7  
4572.6 3304.75 2133.95 7963.4 6656.25 5506.15 2769.35 740.6 4396.8
```

---

## Chapter 4

### Conclusion

#### 4.1.1 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. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

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

Evaluating a model is a core part of building an effective machine learning model. There are several evaluation metrics, like confusion matrix, cross-validation, AUC-ROC curve, etc. Different evaluation metrics are used for different kinds of problems.

#### 4.1.2 MAPE

Here in regression model the evaluation is done by MAPE.

The mean absolute percentage error (MAPE), also known as mean absolute percentage deviation (MAPD), is a measure of prediction accuracy of a forecasting method in statistics, for example in trend estimation, also used as a loss function for regression problems in machine learning. It usually

expresses accuracy as a percentage, by Multiplying 100 to percentage error.

### For R

```
mape=function(av,pv){  
  mean(abs((av-pv)/av))*100 #av=actual value and pv=  
  predicted value  
}
```

### For python

```
#mape                                #av= actual value and pv= predicted  
value  
def mape(av, pv):  
    mape = np.mean(np.abs((av - pv) / av))*100  
    return mape
```

From above we can define the mape function in both R and Python.

The internal mape operation is

$$\frac{\sum_{t=1}^n \left| \frac{(Y_t - \hat{Y}_t)}{Y_t} (100) \right|}{n}$$

## 4.1.3 Model Selection

R (Decision tree regression) mape

```
> mape(test[,9],predictions_tree)
```

```
[1] 13.00985
```



```
> #error rate=13.80%
```

```
> #accuracy =86.00 %
```

R (Random forest regression) mape

```
> mape((test[,9]),RF_Predictions)
```

```
[1] 7.36288
```

```
> #error=7.5 %
```

```
> #accuracy=93.5
```

Python (Random forest regression )

```
mape = np.mean(np.abs((av - pv) / av))*100
```

```
3.175962395781774
```

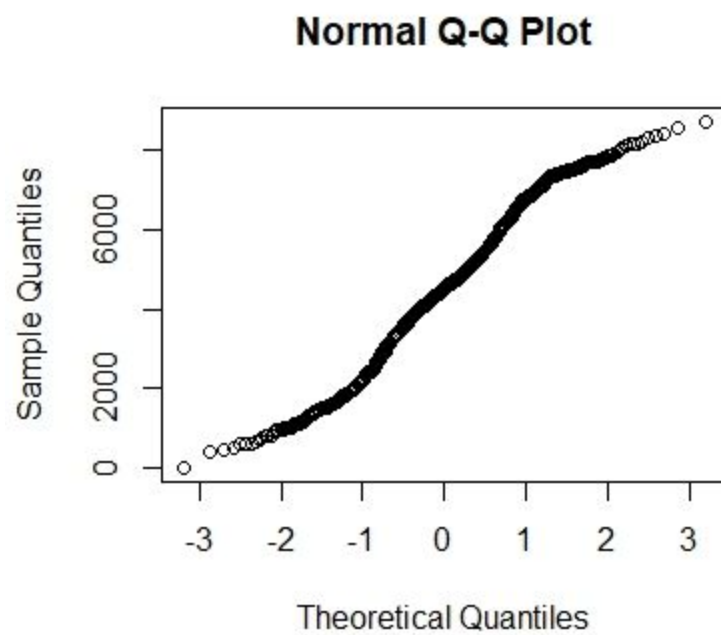
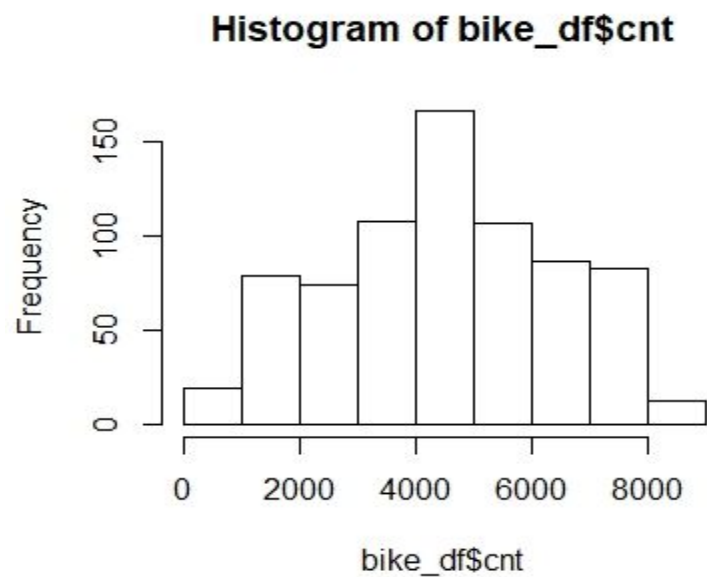
```
> # error= 3.1%    ># accuracy= 97%
```

We can see that both models perform comparatively on average and therefore we can select either of the two models without any loss of information.

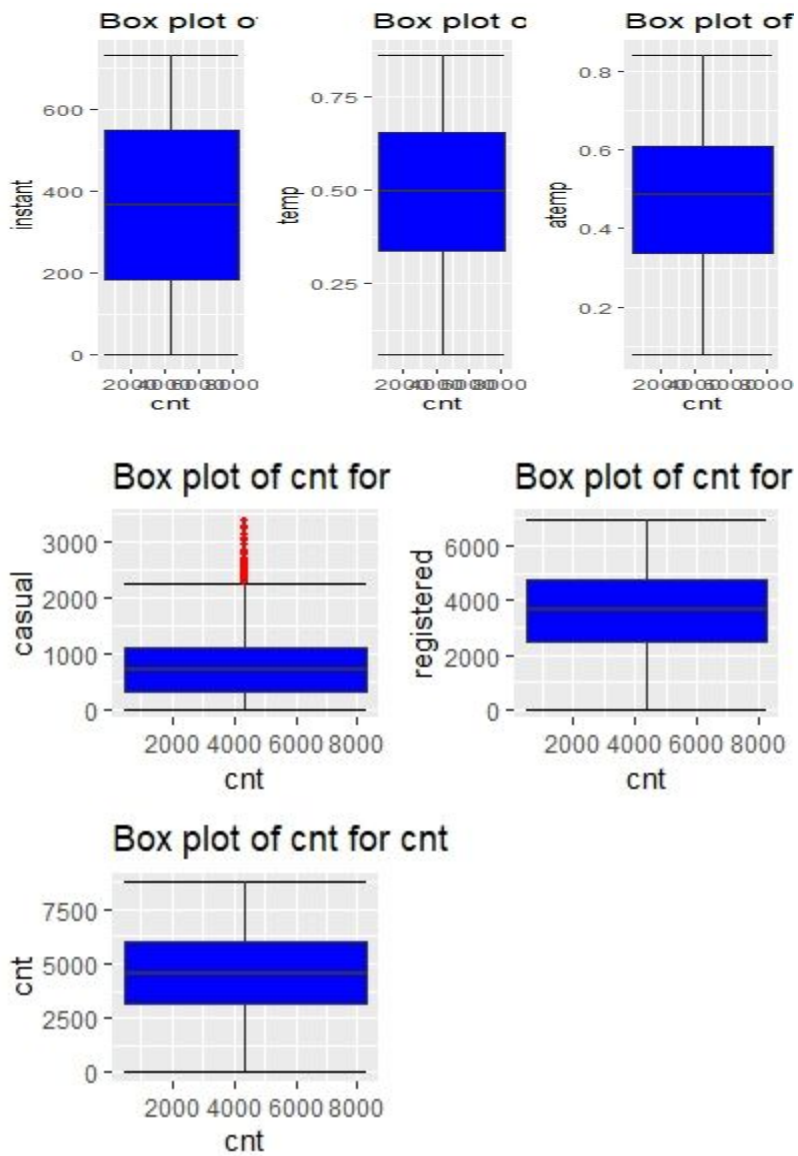
Here the random forest regressor is performing well. [So freezing the random forest regressor model for bike rental data set in both python and R.](#)

## Appendix A - Extra Figures

Normality and distribution plots.

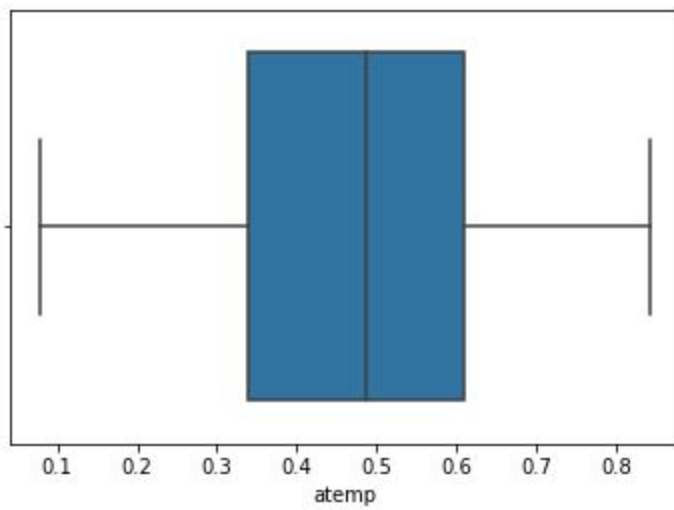
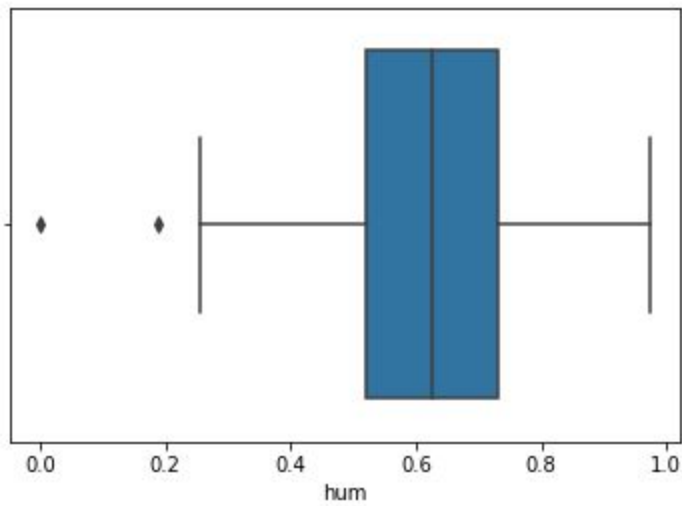


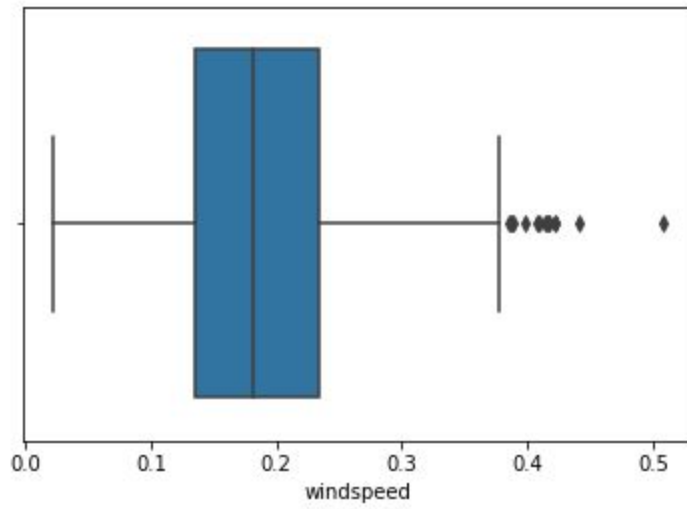
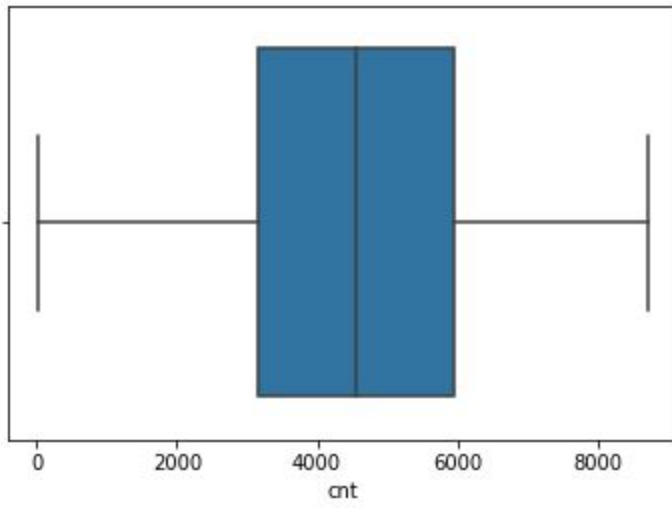
b



BOX PLOTS FOR PREDICTOR VARIABLES.

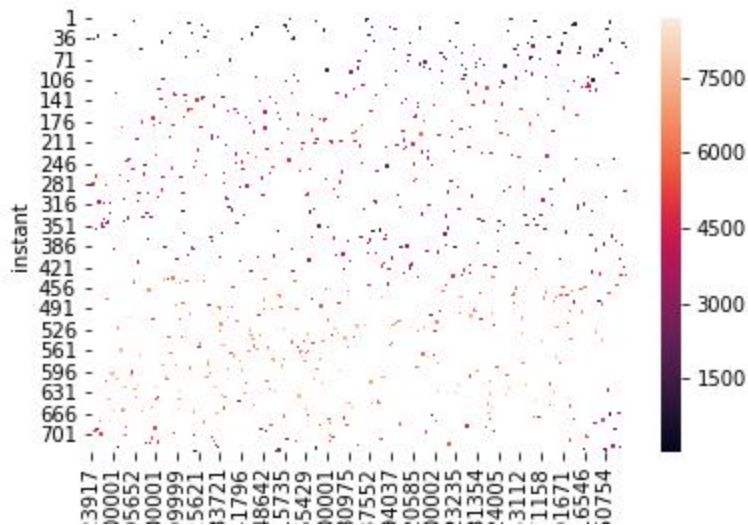
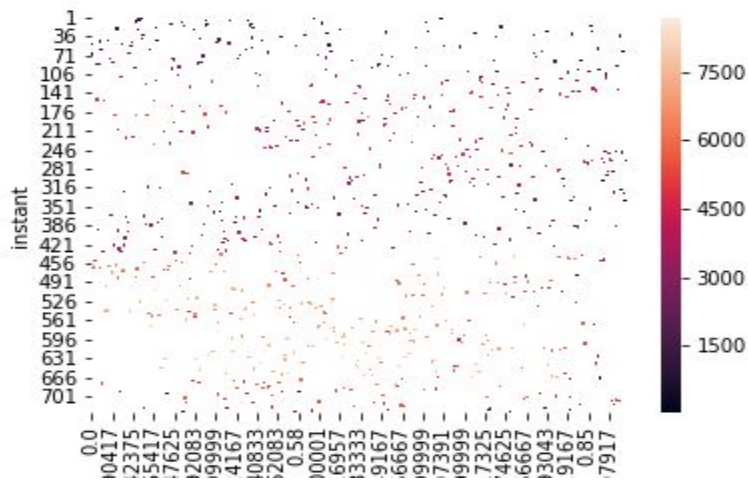
## Python {boxplot using seaborn}

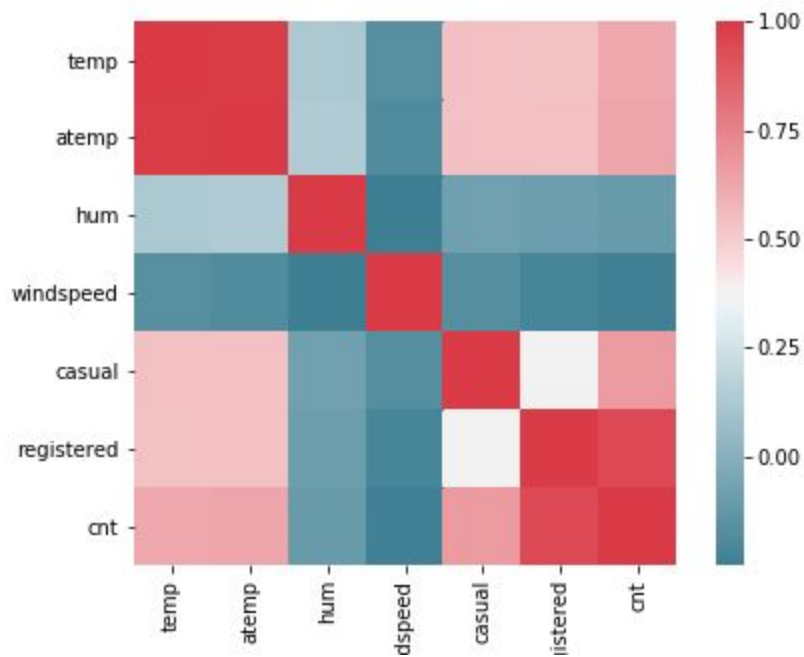
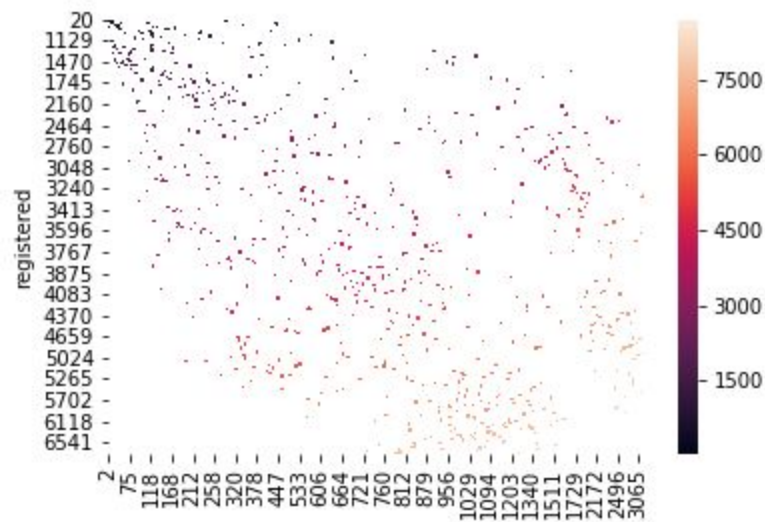






The below graphs are plotted using pivot table in python to know number of bike counts in every seasonal change.





The above graph is correlation plot matrix.

---

## Complete R File

```
# clearing the environment
rm(list=ls())

# setting the working directory
setwd("D:/R and PYTHON files/data set/project 1")

# checking the working directory
getwd()

#Load Libraries
install.packages (c("ggplot2", "corrgram", "DMwR", "caret",
"randomForest", "unbalanced", "C50", "dummies", "e1071",
"Information","MASS", "rpart", "gbm", "ROSE", 'sampling',
'DataCombine', 'inTrees'))

# loading the data in environment (in object bike_df)
bike_df=read.csv("day.csv",header=TRUE)

# missing value analysis
sum(is.na(bike_df))      # there is no missing value in this
dataset.

# exploring the data
str(bike_df)

# converting the variables in their respective datatype.
bike_df$season=as.factor(bike_df$season)
bike_df$yr=as.factor(bike_df$yr)
bike_df$mnth=as.factor(bike_df$mnth)
```

```

bike_df$holiday=as.factor(bike_df$holiday)
bike_df$weekday=as.factor(bike_df$weekday)
bike_df$workingday=as.factor(bike_df$workingday)
bike_df$weathersit=as.factor(bike_df$weathersit)

# seperating the numerical variables and categorical
variables.
numeric_index=sapply(bike_df,is.numeric)
numeric_data=bike_df[,numeric_index]
cnames=colnames(numeric_data) #numerical variables.

factor_index=sapply(bike_df,is.factor)
factor_data=bike_df[,factor_index]
pnames=colnames(factor_data) #categorical variables.

#outlier detection and deletion.
library("scales")
library("psych")
library("ggplot2")
library("gplot")
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]),
x = "cnt"), data = subset(bike_df))+
    stat_boxplot(geom = "errorbar",width = 0.5) +
    geom_boxplot(outlier.colour="red", fill = "blue"
,outlier.shape=18,
              outlier.size=1, notch=FALSE) +
    theme(legend.position="bottom")+
    labs(y=cnames[i],x="cnt")+
    ggtitle(paste("Box plot of cnt for",cnames[i])))
}

```

```

# Plotting plots together
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=3)
gridExtra::grid.arrange(gn4,gn5,ncol=2)
gridExtra::grid.arrange(gn6,gn7,gn8,ncol=2)

# removing outlier
for (i in cnames){
  print(i)

  value=bike_df[,i][bike_df[,i]%in%boxplot.stats(bike_df[,i])$
out]
}
bike_df=bike_df[which(!bike_df[,i]%in%value),]

# correlation plot for numerical variables.
library(corrgram)
corrgram(bike_df[,cnames],order=F,upper.panel=panel.pie,text
.panel=panel.txt,mean="correlation plot")

# chi^2 test for categorical variables.
for(i in 1:8 ){

  print(names(factor_data)[i])
  print(chisq.test(table(factor_data[,i])))
}

# deleting the features which are not wanting.
bike_del=
subset(bike_df,select=-c(atemp,dteday,instant,season,yr,mnth
,weekday))

```

```
# preparing the data for forecast and predict
str(bike_del)
train.index = sample(1:nrow(bike_del), 0.8 * nrow(bike_del))
train = bike_del[ train.index,]
test  = bike_del[-train.index,]
```

```
#defining the function (to find the error percentage)
mape=function(av,pv){
  mean(abs((av-pv)/av))*100 #av=actual value and pv=
predicted value
}
```

```
#decision tree regression model
library(rpart)
data1=rpart(cnt~.,data=train,method="anova")
predictions_tree=predict(data1,test[, -9])
summary(data1)
mape(test[,9],predictions_tree)
#error rate=13.00%
#accuracy =87.00 %
```

```
#random forest
library(randomForest)
random_model = randomForest(cnt~ ., train, importance =
TRUE, ntree = 2000)
```

```

#Extract rules from random forest
#transform rf /object to an inTrees' format
library(inTrees)
treeList = RF2List(random_model)

#Extract rules
rules= extractRules(treeList, train[, -14])

#Visualize some rules
rules[1:2,]
#Make rules more readable:
readrules = presentRules(rules, colnames(train))
readrules[1:2,]

#Predict test data using random forest model
RF_Predictions = predict(random_model, test[, -9])

# accuracy check
mape((test[, 9]), RF_Predictions)
#error=7.5 %
#accuracy=93.5

# EDA
qqnorm(bike_df$cnt)
hist(bike_df$cnt)# visualizing few graphs.

# writing the output to hard disk
write(capture.output(predictions_tree), "DTPR.txt")

#freezing the model random forest with high accuracy.

```

---

## Complete python file

```
import os
os.chdir("D:/R and PYTHON files/data set/project 1")
os.getcwd()
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import datetime
import scipy.stats
# after importing the necessary libraries.
# load the data in environment
bikedata=pd.read_csv("day.csv",sep=',')
# Exploratory data analysis.
bikedata.shape
bikedata.head()
# creating the pivot table to visualize the bike count in
different environmental changes.
bikedata2=bikedata.pivot('instant','temp','cnt')
bikedata3=bikedata.pivot('instant','atemp','cnt')
bikedata4=bikedata.pivot('instant','hum','cnt')
bikedata5=bikedata.pivot('instant','windspeed','cnt')
#visualizing the bike count in different environmental
changes.
sns.heatmap(bikedata2).get_figure().savefig('figheat5.png')

sns.heatmap(bikedata3).get_figure().savefig('figheat6.png')
#
sns.heatmap(bikedata4).get_figure().savefig('figheat7.png')
#
```



```
sns.heatmap(bikedata5).get_figure().savefig('figheat8.png')
```

```
# outlier detection.
```

```
sns.boxplot(bikedata['windspeed']).get_figure().savefig('fig4.png')# found the outlier above the upper fence.
```

```
sns.boxplot(bikedata['atemp']).get_figure().savefig('fig2.png')
```

```
outlier=sns.boxplot(bikedata['cnt']).get_figure().savefig('fig3.png')
```

```
sns.boxplot(bikedata['hum']).get_figure().savefig('fig1.png')# found outlier below the lower fence
```

```
plt.hist(bikedata['cnt'])#distribution of count
```

```
# there are no missing values in this data set, as our dataset is very small.
```

```
# converting the variables into perfect datatypes for further analysis.
```

```
bikedata['season']=bikedata['season'].astype('category')
```

```
bikedata['weekday']=bikedata['weekday'].astype('category')
```

```
bikedata['yr']=bikedata['yr'].astype('category')
```

```
bikedata['mnth']=bikedata['mnth'].astype('category')
```

```
bikedata['holiday']=bikedata['holiday'].astype('category')
```

```
bikedata['workingday']=bikedata['workingday'].astype('category')
```

```
bikedata['weathersit']=bikedata['weathersit'].astype('category')
```

```
bikedata['temp']=bikedata['temp'].astype('float')
```

```
bikedata['hum']=bikedata['hum'].astype('float')
```

```
bikedata['atemp']=bikedata['atemp'].astype('float')
```

```
bikedata['windspeed']=bikedata['windspeed'].astype('float')
```

```
bikedata['cnt']=bikedata['cnt']=bikedata['cnt'].astype('float')
```

```

bikedata['registered']=bikedata['registered'].astype('float
')
bikedata['casual']=bikedata['casual'].astype('float')

#deleting the outliers
outliernames=["hum","windspeed"]
for i in outliernames:
    print(i)
    q75, q25 = np.percentile(bikedata.loc[:,i], [75 ,25])
    iqr = q75 - q25

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

    marketing_train =
bikedata.drop(bikedata[bikedata.loc[:,i] < min].index)
    marketing_train =
bikedata.drop(bikedata[bikedata.loc[:,i] > max].index)
# feature selection/engineering
# saving all the numerical values in the object cnames
cnames=["temp","atemp","hum","windspeed","casual","register
ed","cnt"]
bike_corr=bikedata.loc[:,cnames]
f, ax = plt.subplots(figsize=(7, 5))
correlation_matrix=bike_corr.corr()
#correlation plot
sns.heatmap(correlation_matrix,mask=np.zeros_like(correlati
on_matrix,dtype=np.bool),cmap=sns.diverging_palette(220,10,
as_cmap=True),square=True,ax=ax).get_figure().savefig('pyth
onheat_map.png')

```

```

# saving the categorical index in object catnames
# chi^2 test
catnames=["season","mnth","yr","holiday","weekday","working
day","weathersit"]
from scipy.stats import chi2_contingency
for i in catnames:
    print(i)

chi2,p,dof,ex=chi2_contingency(pd.crosstab(bikedata['cnt'],
bikedata[i]))
    print(p)
# so removing all the less feature variables
bikedata=bikedata.drop(['atemp','dteday','instant','season'
,'yr','mnth','weekday'],axis=1)
bikedata.head()

# data modelling
import sklearn
from sklearn.model_selection import train_test_split
from sklearn import tree
from sklearn.ensemble import RandomForestRegressor
# preparing the data for model.
x=bikedata.values[:,0:8]
y=bikedata.values[:,8]
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.
2)
# random forest
random_forest=RandomForestRegressor(n_estimators =
20).fit(xtrain, ytrain)
# prediction
randomforest_predict=random_forest.predict(xtest)
#mape                                #av= actual value and pv= predicted

```

```
value
def mape(av, pv):
    mape = np.mean(np.abs((av - pv) / av))*100
    return mape

mape(ytest,randomforest_predict)
7
#error percentage =3.17 %
#accuracy of the model =97 %
print(randomforest_predict)
# freezing the model random forest with high accuracy.
```

**END OF REPORT**