PROJECT

Predication of Bike Rental count based On the Environmental and Seasonal settings.

Submitted By:

Mosouwer Jamil

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

1.1 PROBLEM STATEMENT

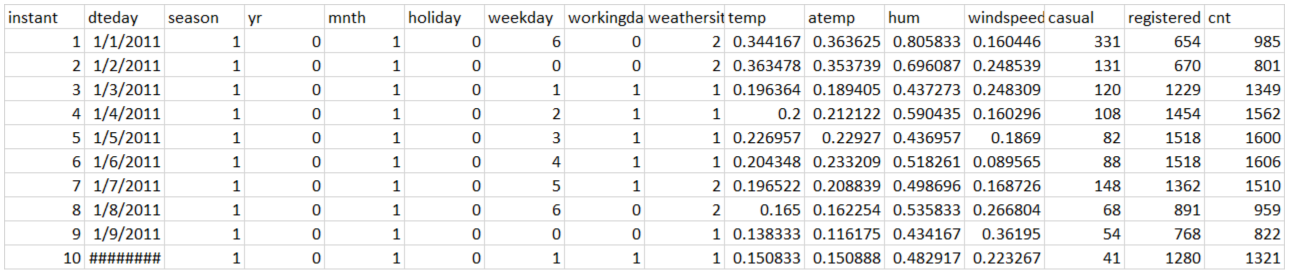
The project is about a bike rental company who has its historical data, and now our objective of this Project is to predict the bike rental count on daily basis, considering the environmental and seasonal settings. These predicted values will help the business to meet the demand on those particular days by maintain the amount of supply.

Nowadays there are number of bike renting companies like, Ola Bikes, Rapido etc. And these bike renting companies deliver services to lakhs of customers daily. Now it becomes really important to manage their data properly to come up with new business ideas to get best results. In this case we have to identify in which days there can be most demand, such that we have enough strategies met to deal with such demand.

1.2 DATA

The given dataset contains 16 variables and 731 observations. The “cnt” is the target variable and remaining all other variables are the independent variables.

Our objective is to develop a model that can determine the count for future test cases. And this model can be developed by the help of given data. A snapshot of the data is mentioned following.



instant	dteday	season	yr	mnth	holiday	weekday	workingda	weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
1	1/1/2011	1	0	1	0	6	0	2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	1/2/2011	1	0	1	0	0	0	2	0.363478	0.353739	0.696087	0.248539	131	670	801
3	1/3/2011	1	0	1	0	1	1	1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
4	1/4/2011	1	0	1	0	2	1	1	0.2	0.212122	0.590435	0.160296	108	1454	1562
5	1/5/2011	1	0	1	0	3	1	1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
6	1/6/2011	1	0	1	0	4	1	1	0.204348	0.233209	0.518261	0.089565	88	1518	1606
7	1/7/2011	1	0	1	0	5	1	2	0.196522	0.208839	0.498696	0.168726	148	1362	1510
8	1/8/2011	1	0	1	0	6	0	2	0.165	0.162254	0.535833	0.266804	68	891	959
9	1/9/2011	1	0	1	0	0	0	1	0.138333	0.116175	0.434167	0.36195	54	768	822
10	########	1	0	1	0	1	1	1	0.150833	0.150888	0.482917	0.223267	41	1280	1321

Table: Data

CHAPTER 2: METHODOLOGY

After going through the dataset in detail and pre-understanding the data the next step is, Methodology that will help achieve our goal.

In Methodology following processes are followed:

- Pre-processing:

  It includes missing value analysis, outlier analysis, feature selection and feature scaling.

- Model development:

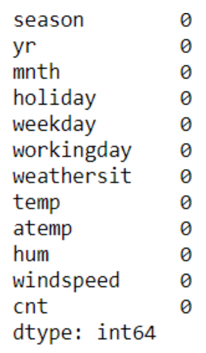
  It includes identifying suitable Machine learning Algorithms and applying those algorithms in our given dataset.

2.1 Pre-processing

Here, we will use techniques like missing value analysis, outlier analysis, feature selection, feature scaling. This techniques are used to structure our data. Basically, pre-processing is done because and the model asks for structured data and preprocessing is used to structure the data we have got. As, normally the data we get can be messy i.e.: it can include many missing values, inconsistent values etc. And this things needs to be checked prior developing a model.

2.1.1 Missing Value Analysis

Missing value is availability of incomplete observations in the dataset. This is found because of reasons like, incomplete submission, wrong input, manual error etc. These Missing values affect the accuracy of model. So, it becomes important to check missing values in our given data.



```
season        0
yr            0
mnth          0
holiday       0
weekday       0
workingday    0
weathersit    0
temp          0
atemp         0
hum           0
windspeed     0
cnt           0
dtype: int64
```

No missing values found

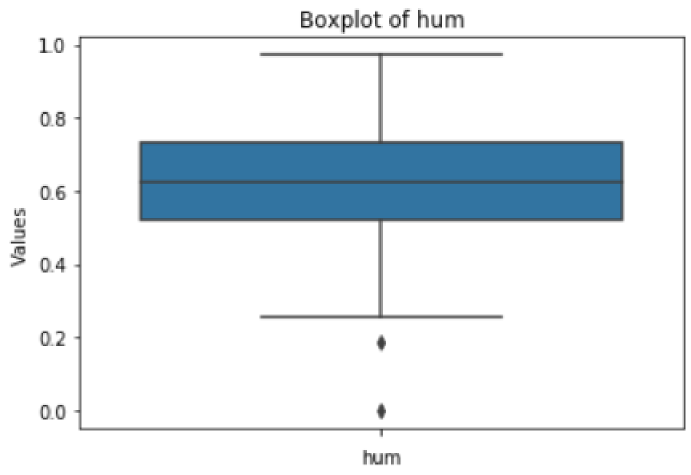
As there is no missing values found in our given data, thus we don't need to follow imputation processes here. So, we can directly move to our next step that is outlier analysis.

2.1.2 Outlier Analysis

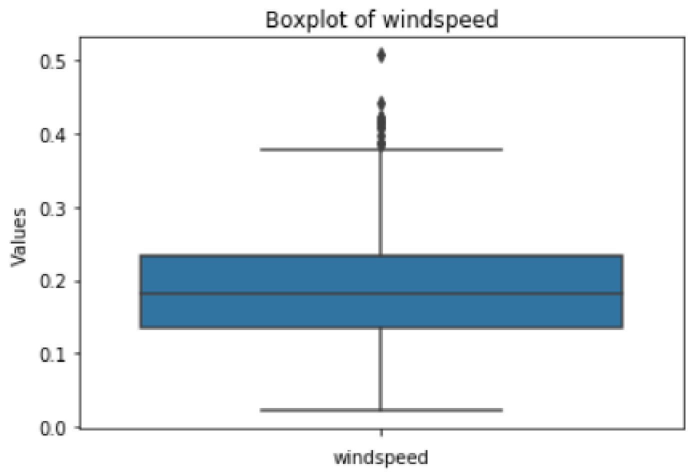
Outlier is an abnormal observation that stands or deviates away from other observations. These happens because of manual error, poor quality of data and it is correct but exceptional data. But, it can cause an error in predicting the target variables. So we have to check for outliers in our data set and also remove or replace the outliers wherever required.

In this project, outliers are found in only two variables this are Humidity and windspeed, following are the box plots for both the variables and dots outside the quartile ranges are outliers.

```
hum
```



```
windspeed
```

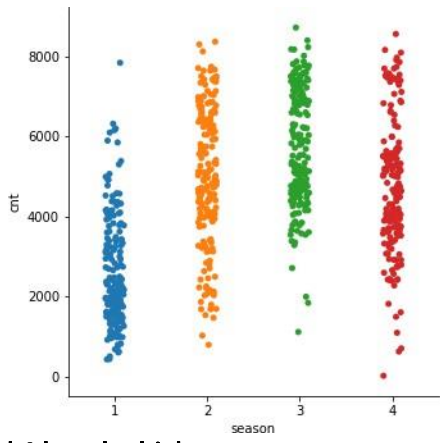


All this outliers mentioned above happened because of manual error, or interchange of data, or may be correct data but exceptional. But all these outliers can hamper our data model. So there is a requirement to eliminate or replace such outliers, and impute with proper methods to get better accuracy of the model. In this project, I used median method to impute the outliers in windspeed and humidity variables.

2.1.3 Data Understanding

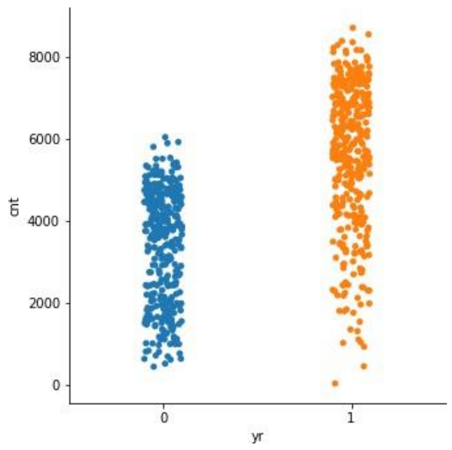
Data Understand is a process where we know our data in a better way by the help of visual representations and come up with initial ideas to develop our model. Here, the specific variables are plotted with respect to the target variable. In some cases two variables are compared, whereas in some cases three variables are plotted together for our better understanding and visualization.

a. Season



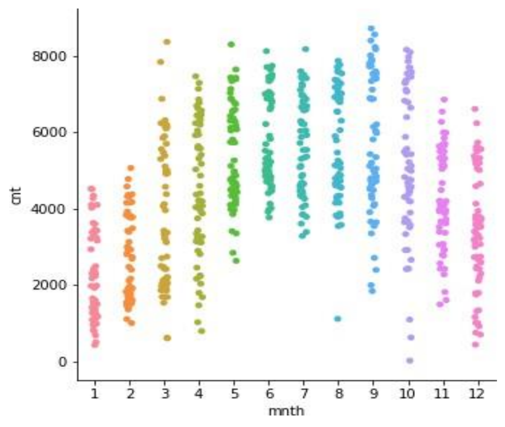
Here, it is found that in Season 2, 3 and 4 has the highest count

b. Year



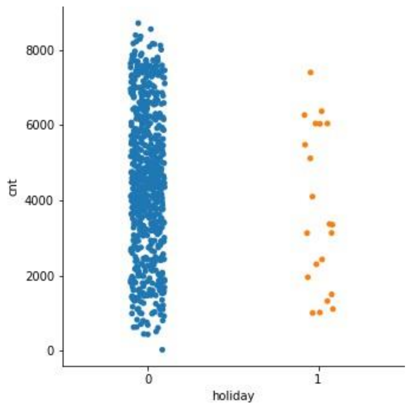
Here, it is found that in Year 1 has high count than 0

c. Month



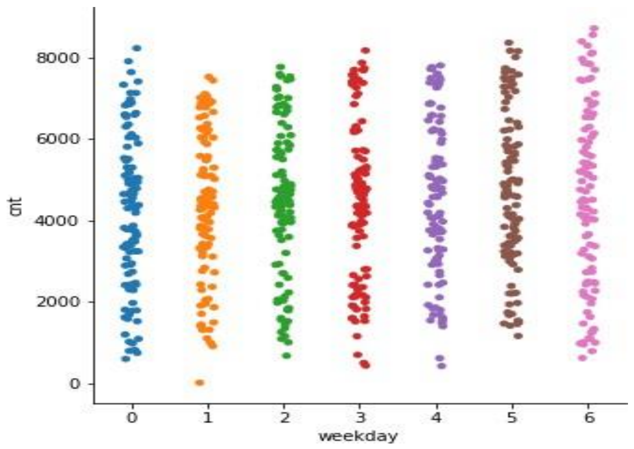
Here, it is observed that in Months 3 to 10 we got a good number of count t

d. Holidays and Non-Holidays



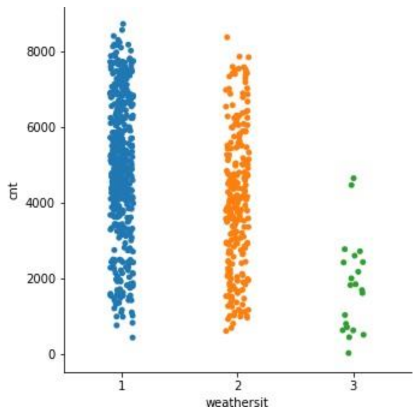
Here, it is found that, on holidays the count is higher when compared non-holidays

e. Weekdays



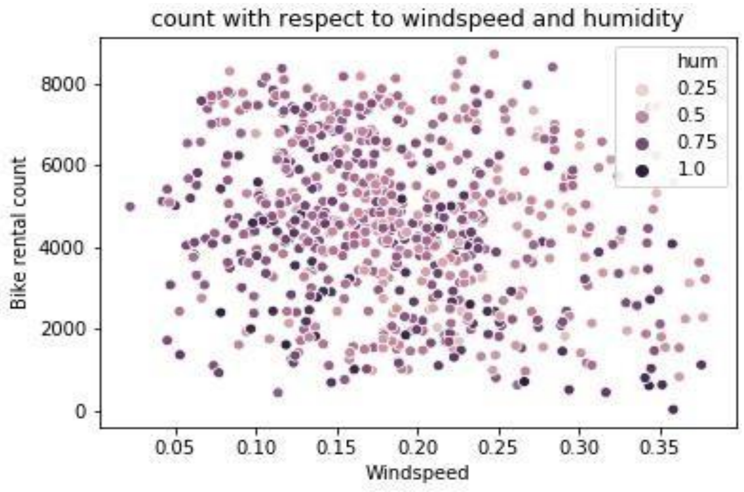
Here, it is observed that in weekdays, 0 and 6 i.e. Monday to Saturday the count is highest.

Weather



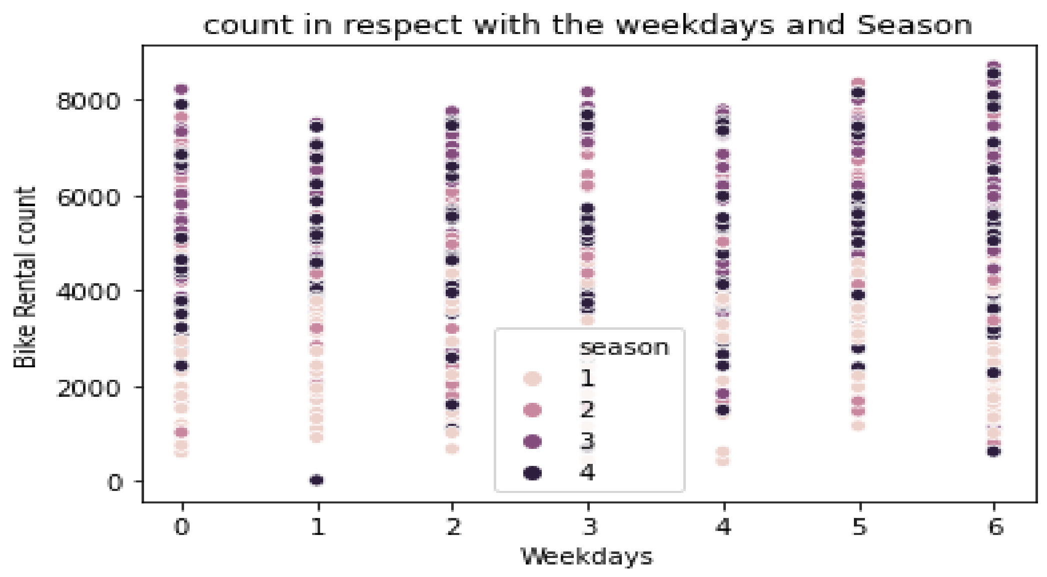
Here, in weather it is observed that, weather 1 has the highest count

f. Windspeed and Humidity vs count



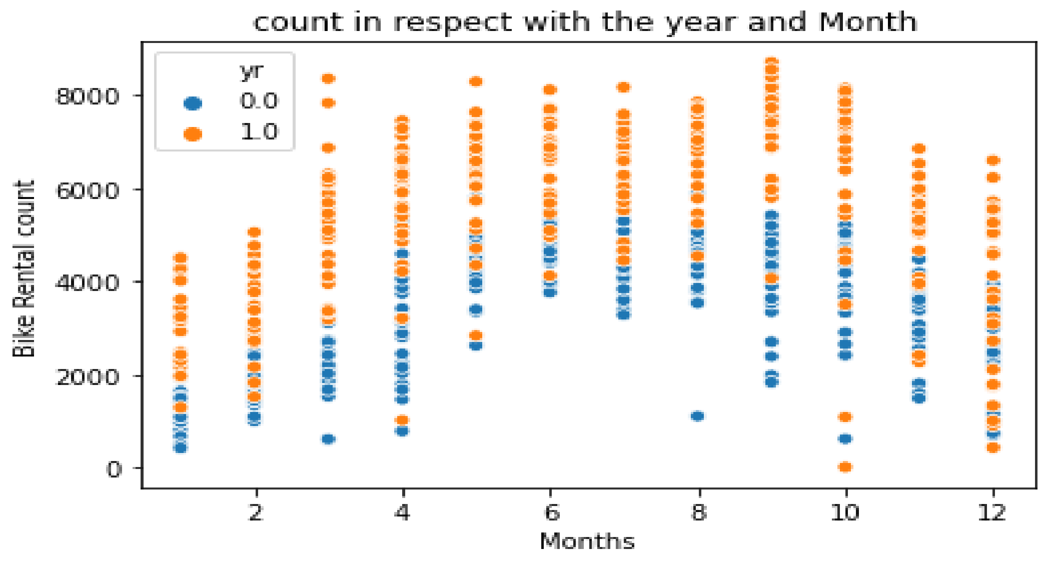
Here, it is found that in count vs windspeed and humidity, Count is High in ranges of windspeed 0.10 to 0.25 and humidity 0.5 to 0.75

g. Weekdays and Season vs count



Here, it is observed that in count vs weekdays and season, Count is high in 4th season and 1st and 6th of weekdays

h. Year and month vs count

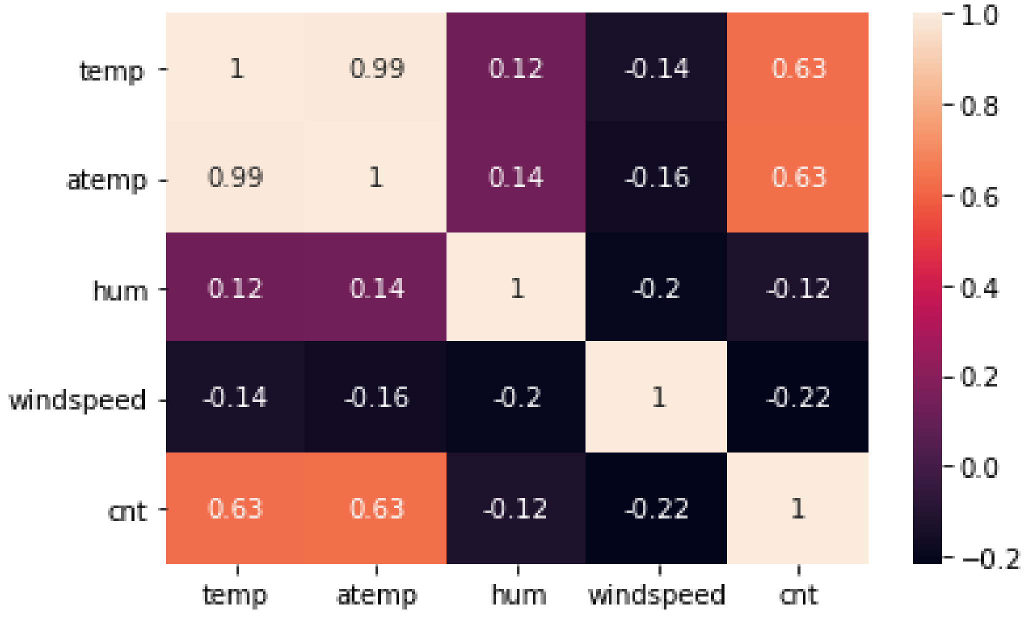


Here, it is found that count vs respect to year and month, count is high in year 1, particularly from season 3 to 12 excluding 9th.

Feature Selection

Sometimes it happens that, all the variables in our data may not be accurate enough to predict the target variable, in such cases we need to analyze our data, understand our data and select the dataset variables that can be most useful for our model. In such cases we follow feature selection. Feature selection helps by reducing time for computation of model and also reduces the complexity of the model.

a. Correlation Analysis for Numerical Variables.



Observing here, it is found that temperature and atemp are highly correlated with each other. So,in further processes we can drop atemp as it is similar to temperature.

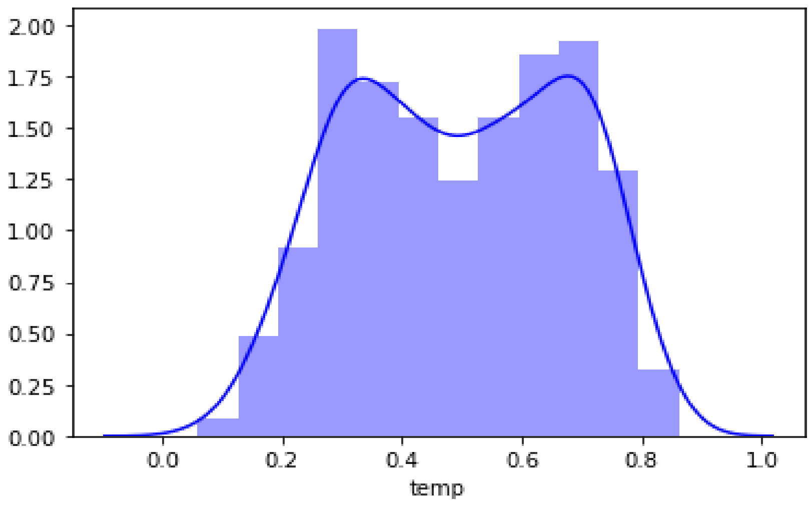
b. ANOVA Test for Categorical Variables

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

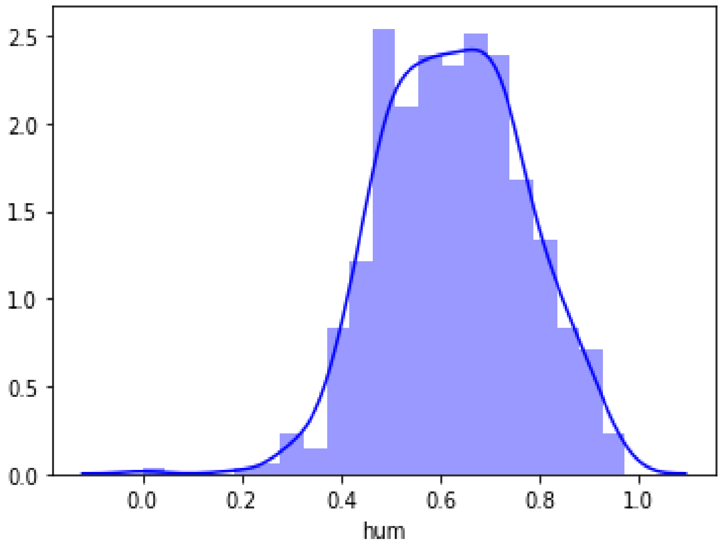
From the observations, it is found that the variables holiday, weekday, and working day has p value >0.05. Here, null hypothesis is accepted. I.e. this variables has no dependency over target variable. So, in further processes this variables can be dropped before modeling. And this process of deducting the variables is also called as dimension reduction.

2.1.5 Feature Scaling

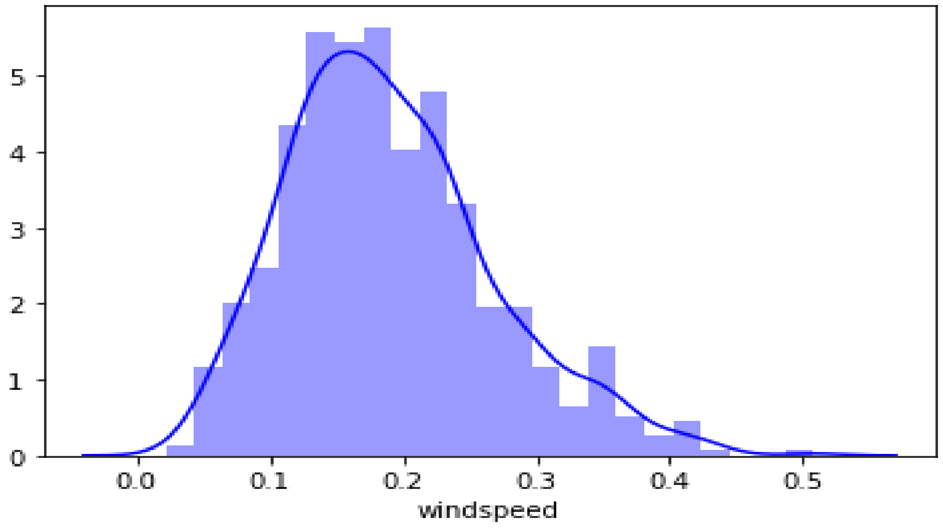
Here, In Feature Scaling ranges of variables are normalized or standardized, such that variables can be compared with same range. This is done for an unbiased and accurate model.
In this project, as the data are found as approximately symmetric. The feature scaling is not required.
Following are the plots of approximately symmetric data visuals.



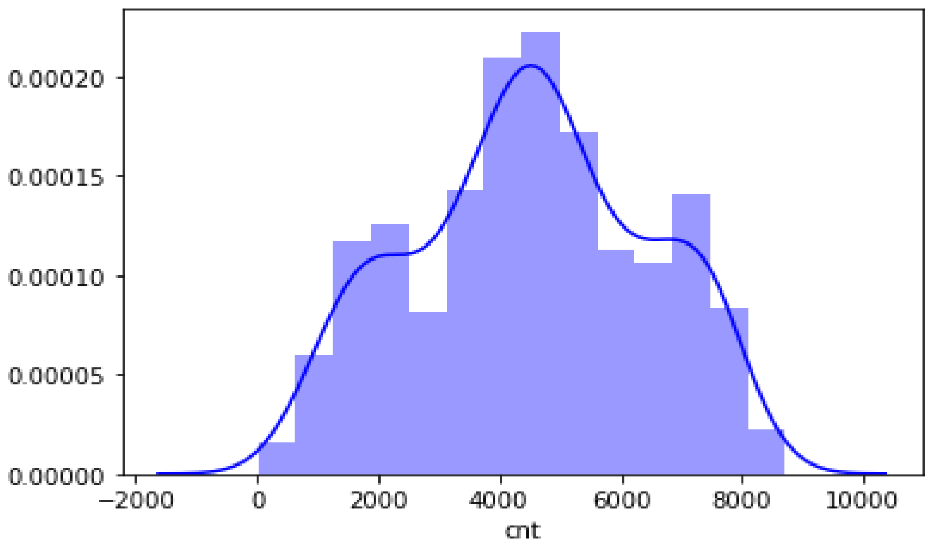
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.0
0.2
0.4
0.6
0.8
1.0
temp

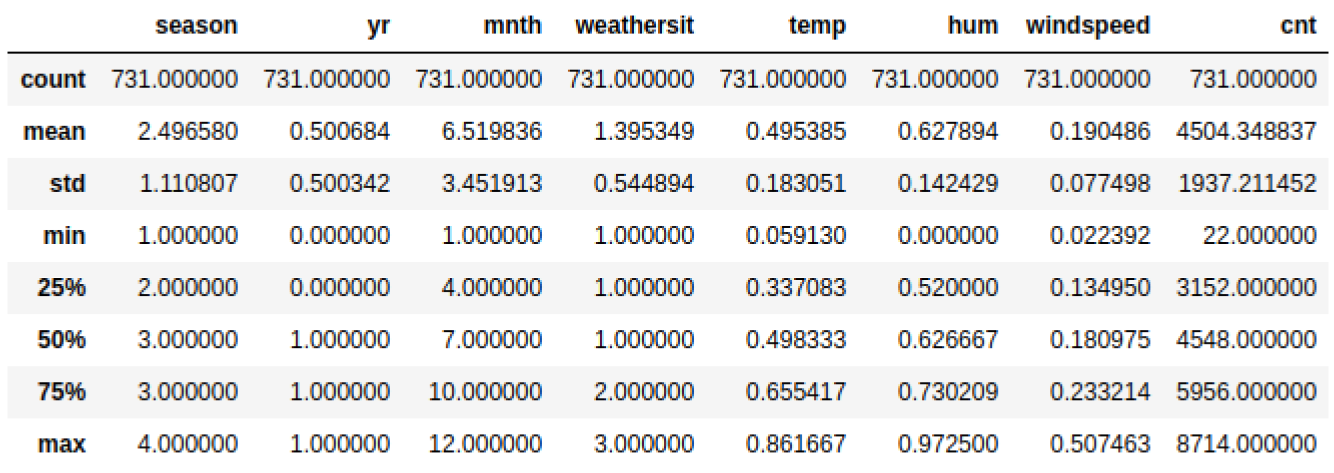


2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0
hum



5
4
3
2
1
0
0.0
0.1
0.2
0.3
0.4
0.5
windspeed





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

data is normalized, No need of scaling

2.2 Model Development

The next step after Exploratory Data Analysis and Data Pre-Processing is Model Development. Now we have our data ready to be implemented to develop a model. There are number of models and Machine learning algorithms that are used to develop model, some are like decision tree, random forest, SVM, KNN, Naïve Bayes, Linear regression, Logistic Regression etc. So, before implementing any model we have to choose precisely our model. So, the first step in Model Development is selection of model.

2.2.1 Model Selection

As per industry standards, there are four categories of models that are derived by classifying problem statement and goal of the project. These categories are:

- Forecasting
- Classification
- Optimization
- Unsupervised Learning

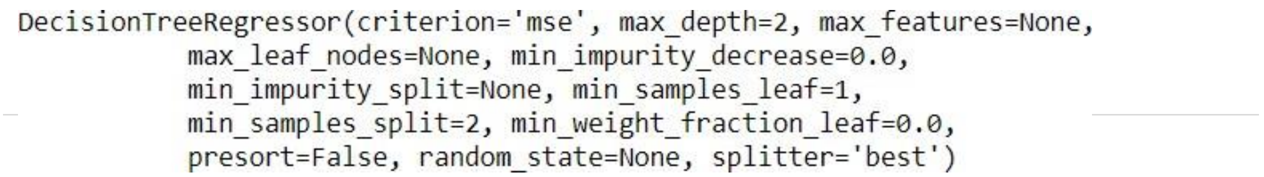
The process of selecting precise model depends on our goal and the problem statement. In this project the problem statement is to predict the bike rental count on daily basis, considering the environmental and seasonal settings. Thus, the problem statement is an identified as regression problem and falls under the category of forecasting, where we have to forecast a numeric data or continuous variable for the target.
Basis of understanding the criteria and given data's problem statement. In this project Decision Tree, Random Forest and Linear Regression are models selected for Model Development.

2.2.2 Decision Tree

Decision Tree is a supervised learning predictive model that uses a set of binary rules to calculate the target value/dependent variable.
Decision trees are divided into three main parts this are:

- **Root Node** : performs the first split
- **Terminal Nodes** : that predicts the outcome, these are also called leaf nodes
- **Branches** : arrows connecting nodes , showing the flow from root to other leaves

a. Decision Tree in Python



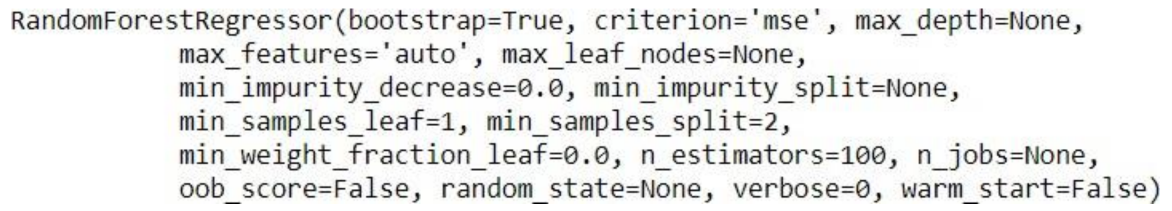
```
DecisionTreeRegressor(criterion='mse', max_depth=2, max_features=None,
           max_leaf_nodes=None, min_impurity_decrease=0.0,
           min_impurity_split=None, min_samples_leaf=1,
           min_samples_split=2, min_weight_fraction_leaf=0.0,
           presort=False, random_state=None, splitter='best')
```

The above fit plot shows the criteria that is used in developing the decision tree in Python. To develop themodel in python, during modeling I have kept all the attributes at default, except the depth as 2. Although these attributes can be played around to derive better score of the model, which is called Hyper tuning of the model. After this the fit is used to predict in test data and the error rate, R-Square and accuracy is calculated.

MAPE: 36.948
RSQUARE: 0.654
ACCURACY: 63.051

2.2.3 Random Forest
The next model to be followed in this project is Random forest. It is a process where the machine follows an ensemble learning method for classification and regression that operates by developing a number of decision trees at training time and giving output as the class that is the mode of the classes of all the individual decision trees.

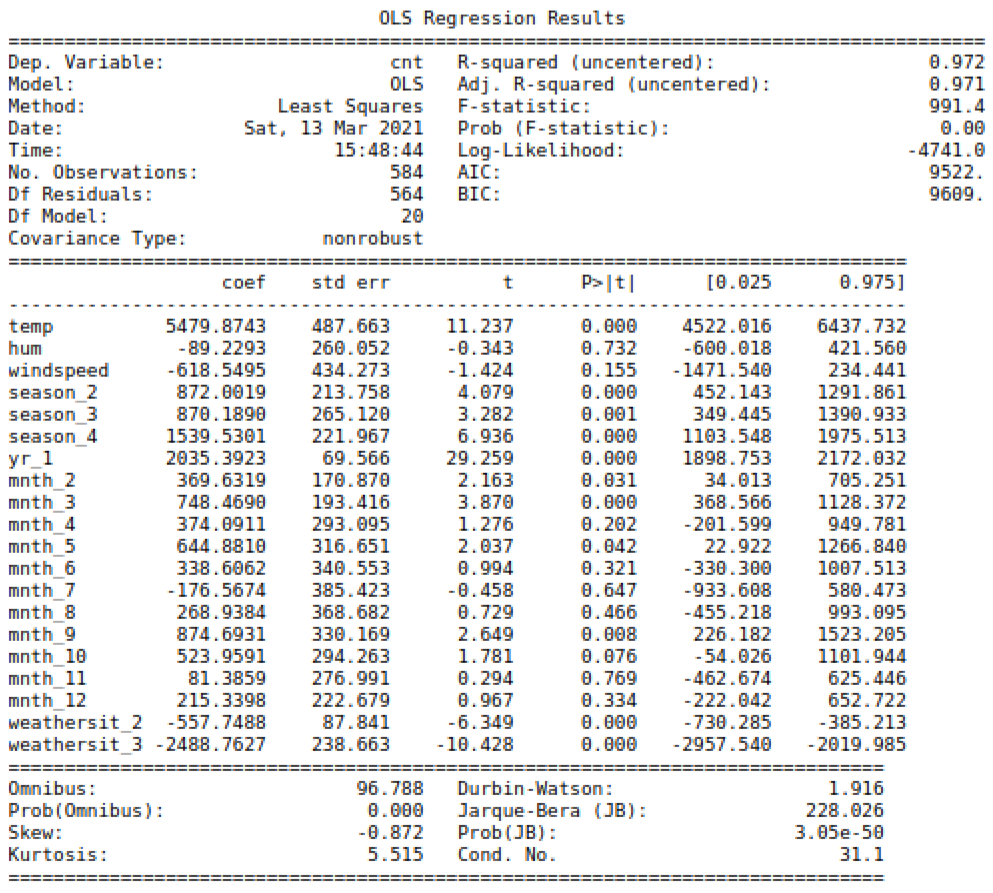
Like the Decision tree above are all the criteria values that are used to develop the Random Forest model in



```
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=100, n_jobs=None,
           oob_score=False, random_state=None, verbose=0, warm_start=False)
```

python. Everything is kept default only except n_estimators, which is tree numbers. Although this attributes can be altered to get a model with a better score. After this the error rate, R Square and
accuracy of the model is noted.
MAPE: 21.586
RSQUARE: 0.878
ACCURACY: 78.413

2.2.3 Linear Regression
The next method in the process is linear regression. It is used to predict the value of variable Y based on
one or more input predictor variables X. The goal of this method is to establish a linear relationship
between the predictor variables and the response variable. Such that, we can use this formula to estimatethe
value of the response Y, when only the predictors (X- Values) are known.



```
                            OLS Regression Results
=======================================================================================
Dep. Variable:                    cnt   R-squared (uncentered):                   0.972
Model:                            OLS   Adj. R-squared (uncentered):              0.971
Method:                 Least Squares   F-statistic:                              991.4
Date:                Sat, 13 Mar 2021   Prob (F-statistic):                        0.00
Time:                        15:48:44   Log-Likelihood:                         -4741.0
No. Observations:                 584   AIC:                                      9522.
Df Residuals:                     564   BIC:                                      9609.
Df Model:                          20
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
temp          5479.8743    487.663     11.237      0.000    4522.016    6437.732
hum            -89.2293    260.052     -0.343      0.732    -600.018     421.560
windspeed     -618.5495    434.273     -1.424      0.155   -1471.540     234.441
season_2       872.0019    213.758      4.079      0.000     452.143    1291.861
season_3       870.1890    265.120      3.282      0.001     349.445    1390.933
season_4      1539.5301    221.967      6.936      0.000    1103.548    1975.513
yr_1          2035.3923     69.566     29.259      0.000    1898.753    2172.032
mnth_2         369.6319    170.870      2.163      0.031      34.013     705.251
mnth_3         748.4690    193.416      3.870      0.000     368.566    1128.372
mnth_4         374.0911    293.095      1.276      0.202    -201.599     949.781
mnth_5         644.8810    316.651      2.037      0.042      22.922    1266.840
mnth_6         338.6062    340.553      0.994      0.321    -330.300    1007.513
mnth_7        -176.5674    385.423     -0.458      0.647    -933.608     580.473
mnth_8         268.9384    368.682      0.729      0.466    -455.218     993.095
mnth_9         874.6931    330.169      2.649      0.008     226.182    1523.205
mnth_10        523.9591    294.263      1.781      0.076     -54.026    1101.944
mnth_11         81.3859    276.991      0.294      0.769    -462.674     625.446
mnth_12        215.3398    222.679      0.967      0.334    -222.042     652.722
weathersit_2  -557.7488     87.841     -6.349      0.000    -730.285    -385.213
weathersit_3 -2488.7627    238.663    -10.428      0.000   -2957.540   -2019.985
==============================================================================
Omnibus:                       96.788   Durbin-Watson:                   1.916
Prob(Omnibus):                  0.000   Jarque-Bera (JB):              228.026
Skew:                          -0.872   Prob(JB):                     3.05e-50
Kurtosis:                       5.515   Cond. No.                         31.1
==============================================================================
```

Plot: Linear regression Python

Here, F-Statistic explains about the quality of the model. AIC is Akkaine information criterion, if we have multiple models with same accuracy then we need to refer this to choose the best model. The table three values containing Omnibus and JB test are mostly required for time variance analysis. Here, as we are not using any time values in our project we can ignore this table 3. T-statistic explain how much statistically significant the coefficient is. It is also used to calculate the P –Value. And if P-Value is less than 0.05 we reject null hypothesis and say that the variable is significant. Here, all the variables are less than 0.05 and are significant. The R squared and adjusted R squared values show how much variance of the output variable is explained by the independent or input variables. Here the adjusted r square value is 82.7%, which explains that only 83% of the variance of count is explained by the input variables. This shows that the model is performing well. After this predictions are done and error metrics are calculated.

MAPE: 21.586
RSQUARE: 0.878
ACCURACY: 78.413

Model Summary:
From the above mentioned various models that can be developed for the given data. At first place, The Data is divided into train and test. Then the models are developed on the train data. After that the model is fit into it to test data to predict the target variable. After predicting the target variable in test data, the actual and predicted values of target variable are compare to get the error and accuracy. And looking over the error and accuracy rates, the best model for the data is identified and it is kept for future usage.

CHAPTER 3: EVALUATION OF THE MODEL

So, now we have developed few models for predicting the target variable, now the next step is evaluate the models and identify which one to choose for deployment. To decide these, error metrics are used. In this project MAPE, R Square and Accuracy are used. And addition to these error metrics K Fold Cross validation is also applied to identify the best model of all.

3.1 Mean Absolute Error (MAE)

MAE or Mean Absolute Error, it is one of the error measures that is used to calculate the predictive performance of the model. It is the sum of calculated errors. In this project we will apply this measure to our models.

In Python :

Method	Mape Error(in Percentage)
Decision Tree	36.9480
Random Forest	20.9466
Linear Regression	18.8006

Table: Mape in Python

If we observe the above tables, we choose the model with lowest MAPE as a suitable Model. Here, from R we get Random Forest as a better model, whereas from Python we get Linear Regression as a better model. So following this we can conclude that Both Random Forest and Linear Regression can be used as model for this data, if you evaluate on the basis of MAPE. But we need more error metrics to cross check this. So, we go for R Square which is a better error metric.

3.1 Accuracy

The second matric to identify or compare for better model is Accuracy. It is the ratio of number of correct predictions to the total number of predictions made.

Accuracy= number of correct predictions / Total predictions made

a. In Python

Method	Accuracy (in Percentage)
Decision Tree	63.051
Random Forest	79.053
Linear Regression	81.199

Table: Accuracy in Python Models

As, Accuracy derives from MAE/MAPE its observations also suggest same models as better models as suggested by MAPE. Here, the models with highest accuracy are chosen, and from the observations it is found that both Random Forest and Linear Regression are good models for the given data set.

3.2 R Square

R Square is another metric that helps us to know about the Correlation between original and predicted values.

In Python

Method	R – Square (in Percentage)
Decision Tree	65.44
Random Forest	88.43
Linear Regression	84.36

Table: Accuracy in Python Models

R Square is identified as a better error metric to evaluate models. If we observe the above tables, we choose the model with highest R Square as a suitable Model. Here, from both R and Python it is found that Random Forest is a best fit model for the given data.

APPENDIX A
R Code

```
rm(list=ls())

 #Set Working Directory
 setwd("C:/Users/Lenovo/Documents/LM/EdWisor/Projects/Project 2")

 getwd()


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


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

 ############Load Data########################################

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


 #Exploratory Data Analysis

 class(Data_Day)
 dim(Data_Day)
 head(Data_Day)
 names(Data_Day)
 str(Data_Day)
 summary(Data_Day)


 #From the above observations

 #Droping few columns
 Data_Day = subset(Data_Day, select = -c(instant, dteday, casual, registered))

 dim(Data_Day)
 names(Data_Day)


 #separate numeric and categorical variables

 numeric_var = c('temp', 'atemp', 'hum', 'windspeed', 'cnt')

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

```
######################## Missing Value analysis ##############################

summary(is.na(Data_Day))
sum(is.na(Data_Day))

#there is no missing values


###############Outlier Analysis ###################################

df = Data_Day
Data_Day =  df

# BoxPlots - Distribution and Outlier Check


library(ggplot2)

for (i in 1:length(numeric_var))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (numeric_var[i]), x = "cnt"), data =
subset(Data_Day))+
           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=numeric_var[i],x="count")+
           ggtitle(paste("Box plot of count for",numeric_var[i])))
}

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


# outliers found in windspeed and humidity variables.


#replacing outliers with NA

for(i in numeric_var){
  print(i)
  outlier = Data_Day[,i][Data_Day[,i] %in% boxplot.stats(Data_Day[,i])$out]
  print(length(outlier))
  Data_Day[,i][Data_Day[,i] %in% outlier] = NA
}

sum(is.na(Data_Day))
```

```
#Impute NA values with KNN

library(DMwR)
library(rpart)

Data_Day = knnImputation(Data_Day, k = 5)

sum(is.na(Data_Day))


#################### Data Understanding ###################


# Time to plot some graphs, so let's install few libraries

library(ggplot2)
library(scales)
library(psych)
library(gplots)


# Barplot with x axis as season and y axis as count

ggplot(Data_Day, aes(x = Data_Day$season, y =  Data_Day$cnt))+
  geom_bar(stat = "identity", fill = "blue")+
  labs(title = "Number of bikes rented with respect to season", x = "Seasons", y =
"cnt")+
  theme(panel.background =element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

#It is found that season 3, has the highest count of bikes and season 1 has lowest
count of bikes


# Barplot with x axis as year and y axis as count

ggplot(Data_Day, aes(x = Data_Day$yr, y = Data_Day$cnt))+
  geom_bar(stat = "identity", fill = "red")+
  labs(title = "Number of bikes rented with respect to year", x = "yr", y = "cnt")+

  theme(panel.background =element_rect("white"))+
  theme(plot.title = element_text(face = "bold"))

# It is found that Year 1 has the highest count while year 0 has lowest count.


# Barplot with x axis as weekday and y axis as count

ggplot(Data_Day, aes(x = Data_Day$weekday, y = Data_Day$cnt))+
   geom_bar(stat = "identity", fill = "navyblue")+
   labs(title = "Number of bikes rented with respect to days", x = "Days of the
week", y = "count")+
   theme(panel.background =element_rect("white"))+
   theme(plot.title = element_text(face = "bold"))

#It is found that on day 5 there is highest count and on day 0 its lowest count of
bikes rented


#Count with respect to temperature and humidity together

ggplot(Data_Day,aes(temp,cnt)) +
   geom_point(aes(color=hum),alpha=0.5) +
   labs(title = "Bikes count vs temperature and humidity", x = "Normalized
temperature", y = "Count")+
   scale_color_gradientn(colors=c('blue','light blue','dark blue','light
green','yellow','dark orange','black')) +
   theme_bw()

#it is found that when normalized temperature is between 0.5 to 0.75 and humidity
is between 0.50 to 0.75, count is high.


# Count with respect to windspeed and weather together

ggplot(Data_Day, aes(x = windspeed, y = cnt))+
   geom_point(aes(color= weathersit ), alpha=0.5) +
   labs(title = "Bikes count vs windspeed and weather", x = "Windspeed", y =
"Count")+
   scale_color_gradientn(colors=c('blue','light blue','dark blue','light
green','yellow','dark orange','black')) +
   theme_bw()

# It is found that count is at peak, when windspeed is from 0.1 to 0.3 and weather
is from 1.0 to 1.5.


# Count with respect to temperature and season together

ggplot(Data_Day, aes(x = temp, y = cnt))+
   geom_point(aes(color=season),alpha=0.5) +
   labs(title = "Bikes count vs temperature and season", x = "Normalized
temperature", y = "Count")+
   scale_color_gradientn(colors=c('blue','light blue','dark blue','light
green','yellow','dark orange','black')) +
   theme_bw()

# it is found that count is maximum when temperature is 0.50 to 0.75  &  season  3  to
season  4.
```

```
################Feature Selection  ############################################

df2 = Data_Day
Data_Day  =  df2

#Correlation Analysis and Anova test is done identify if variables can be reduced
or notis perfo

# Correlation Analysis for numeric variable

library(corrgram)

corrgram(Data_Day[,numeric_var],order=FALSE,upper.panel = panel.pie,
          text.panel = panel.txt,
          main= "Correlation Analysis between numeric variables")

#it is found that temperature and atemp are highly correlated with each other.


# Anova Test for categorical variables

for(i in categorical_var){
  print(i)
  Anova_test_result = summary(aov(formula = cnt~Data_Day[,i],Data_Day))
  print(Anova_test_result)
}

#it is found that holiday, weekday and workingday has p value > 0.05. null
hypothesis accepted


# Dimension redusction , removing variables that ar not required

Data_Day = subset(Data_Day, select=-c(atemp,holiday,weekday,workingday))


############Feature Scaling ########################################


numeric_var = c("temp","hum","windspeed","cnt")
catergorical_var = c("season", "yr", "mnth", "weathersit")


# Skewness test

library(propagate)
```

```
for(i in numeric_var){
  print(i)
  skew = skewness(Data_Day[,i])
  print(skew)
}

#dataset is approximately symmetric. values are found ranging between -0.5 to +0.5.


# Identify range and check min max of the variables to check noramility

for(i in numeric_var){
  print(summary(Data_Day[,i]))
}

#dat is found as normalized, scaling not required


# visualizing normality check

hist(Data_Day$temp, col="Navyblue", xlab="Temperature", ylab="Frequency",
      main="Temperature Distribution")

hist(Data_Day$hum, col="Blue", xlab="Humidity", ylab="Frequency",
      main="Humidity Distribution")

hist(Data_Day$windspeed,col="Dark green",xlab="Windspeed",ylab="Frequency",
      main="Windspeed Distribution")

# the distribution is approximately symmetric



###################MODELING #########


library(DataCombine)
rmExcept("Data_Day")


df3 = Data_Day
Data_Day =  df3

#Develop error metrics

#R Square

Rsquare = function(y,y1){
  cor(y,y1)^2
}
```

```
#MAPE

MAPE = function(y,y1){
  mean(abs((y-y1)/y))*100
}



#######Dummy creation ############

categorical_var = c("season","yr","mnth","weathersit")

library(dummies)

Data_Day = dummy.data.frame(Data_Day, categorical_var)

#Save Data for KFold CV
KFData = Data_Day


#divide data

set.seed(123)
train_index = sample(1:nrow(Data_Day),0.8*nrow(Data_Day))
train=  Data_Day[train_index,]
test= Data_Day[-train_index,]


##############check multicollinearity ##########################

numeric_var = c("temp","hum","windspeed", "cnt")

numeric_var2 = Data_Day[,numeric_var]

library(usdm)

vifcor(numeric_var2, th = 0.7)

#No collinearity  problem.


############DECISION TREE ###################

library(rpart)

DTModel = rpart(cnt~., train, method = "anova" , minsplit=5)


# Predictions
```

```
DTTest = predict(DTModel, test[-25])

summary(DTModel)

#MAPE

DTMape_Test = MAPE(test[,25], DTTest)
DTMape_Test    #26.4225


#RSquare

DT_RSquare = Rsquare(test[,25], DTTest)
DT_RSquare   #0.7612102


############RANDOM  FOREST###################

library(randomForest)
set.seed(123)

RFModel = randomForest(cnt~., train, ntree = 500, importance = TRUE)

#  Predictions

RFTest = predict(RFModel, test[-25])


#  MAPE

RFMape_Test = MAPE(test[,25], RFTest)
RFMape_Test   #   19.32104

#RSquare

RF_RSquare = Rsquare(test[,25], RFTest)
RF_RSquare     #  0.8685008

################LINEAR REGRESSION###################

LRModel = lm(cnt~., train)

summary(LRModel)


# Predictions on test

LRTest = predict(LRModel, test[-25])
```

```
#MAPE

LRMape_Test = MAPE(test[,25], LRTest)
LRMape_Test #   21.56792


#RSquare

LR_RSquare = Rsquare(test[,25], LRTest)
LR_RSquare   #   0.8191175


###########################Model Selection & Evaluation ###########################

print("MAPE  Statistics")
print(DTMape_Test)
print(RFMape_Test)
print(LRMape_Test)

print("Accuracy")
print(100 - DTMape_Test)
print(100 - RFMape_Test)
print(100 - LRMape_Test)


print("R Square Statistics")
print(DT_RSquare)
print(RF_RSquare)
print(LR_RSquare)


########################Cross Validation ############################

#Load Data
library(caret)

KFData

#divide data

set.seed(123)
train_index2 = sample(1:nrow(KFData),0.8*nrow(KFData))
train_KF =  KFData[train_index,]
test_KF  =  KFData[-train_index,]


#Random Forest Cross Validation

RF_KF = train(cnt~.,
                data = train_KF,
                method = "rf",
                tuneGrid = expand.grid(mtry = c(2,3,4)),
                trControl = trainControl(method = "cv",
                                         number =  5,
                                         verboseIter = FALSE,))


print(RF_KF)


knitr::kable(head(RF_KF$results), digits = 3)

print(RF_KF$bestTune)



RFpreds = predict(RF_KF, test_KF[-25])

RFpreds_MAPE = MAPE(test_KF[,25], RFpreds)
RFpreds_MAPE

RFPreds_RSquare = Rsquare(test[,25], RFpreds)
RFPreds_RSquare





#Decision Tree Cross Validation


DT_KF = train(cnt~.,
                   data = train_KF,
                   method = "gbm",
                   tuneGrid = expand.grid(n.trees = 200,
                                          interaction.depth = c(1,2,3),
                                          shrinkage = 0.1,
                                          n.minobsinnode = 10  ),
                   trControl = trainControl(method =  "cv",
                                            number = 5,
                                            verboseIter =  FALSE))

print(DT_KF)


knitr::kable(head(DT_KF$results), digits = 3)

print(DT_KF$bestTune)
```

```
DTpreds = predict(DT_KF, test_KF[-25])

DTpreds_MAPE = MAPE(test_KF[,25], DTpreds)
DTpreds_MAPE

DTPreds_RSquare = Rsquare(test[,25], DTpreds)
DTPreds_RSquare





#Linear Regression CV


LR_KF = train(cnt~.,
              data = train_KF,
              method = "lm",
              tuneGrid = expand.grid(intercept = TRUE),
              trControl = trainControl(method = "cv",
                                       number = 5,
                                       verboseIter =  FALSE))

print(LR_KF)

knitr::kable(head(LR_KF$results), digits = 3)

print(LR_KF$bestTune)

LRpreds = predict(LR_KF, test_KF[-25])

LRpreds_MAPE = MAPE(test_KF[,25], LRpreds)
LRpreds_MAPE

LRPreds_RSquare = Rsquare(test[,25], LRpreds)
LRPreds_RSquare
```

APPENDIX B
Python Code

In [59]:

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from random import randrange,uniform
from sklearn.metrics import r2_score
from scipy import stats
```

In [60]:

```
os.chdir("/home/mosouwer/Downloads")
```

In [61]:

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

In [62]:

```
data.head()
```

Out[62]:

	instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit	temp
0	1	2011-01-01	1	0	1	0	6	0	2	0.344167
1	2	2011-01-02	1	0	1	0	0	0	2	0.363478
2	3	2011-01-03	1	0	1	0	1	1	1	0.196364
3	4	2011-01-04	1	0	1	0	2	1	1	0.200000
4	5	2011-01-05	1	0	1	0	3	1	1	0.226957

In [63]:

```
#data types of variable
data.dtypes
```

Out[63]:

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

In [64]:

```
#shape of the data
data.shape
```

Out[64]:

```
(731, 16)
```

In [65]:

```
#columns
data.columns
```

Out[65]:

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

In [66]:

```
#unique value present in each variable
data.nunique()
```

Out[66]:

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

In [67]:

```
#Defining numeric and categorical variables and saving in specific array

num_var = ['temp', 'atemp', 'hum', 'windspeed', 'cnt']

cat_var = ['season', 'yr', 'mnth', 'holiday', 'weekday', 'workingday', 'weathers
it']
```

DATA PRE PROCESSING

MISSING VALUE ANALYSIS

In [68]:

```
#sum of missing values
data.isnull().sum()
```

Out[68]:

```
instant       0
dteday        0
season        0
yr            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
dtype: int64
```

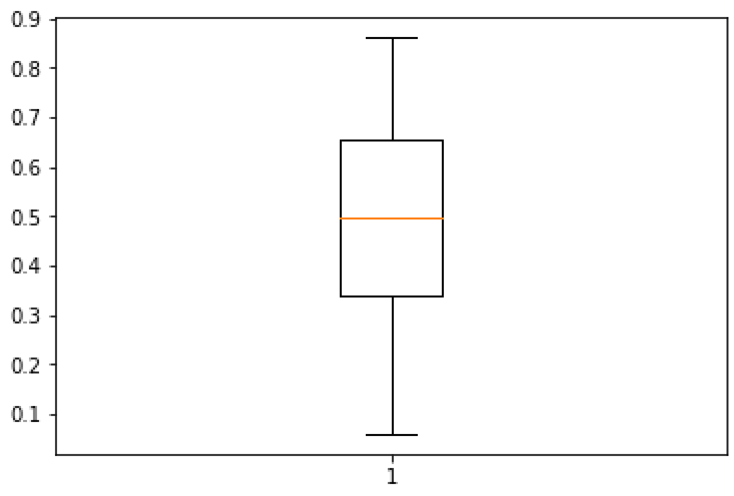
No Missing Value Found

Outliear Analysis

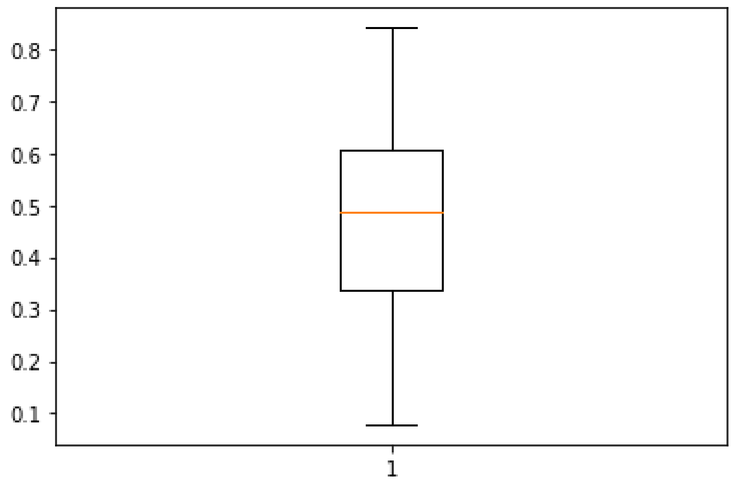
In [69]:

```
for i in num_var:
    print(i)
    plt.boxplot(data[i])
    plt.show()
```

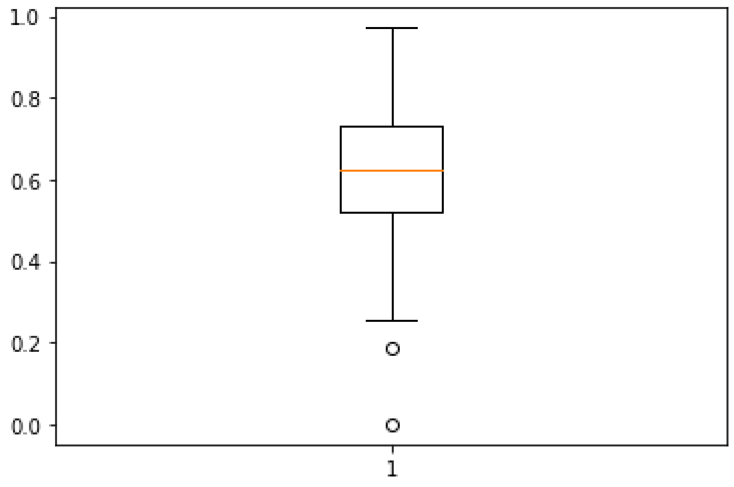
```
temp
```



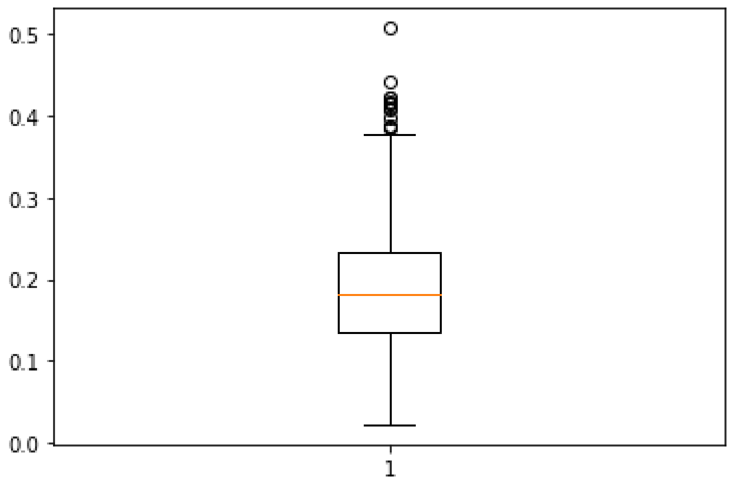
```
atemp
```



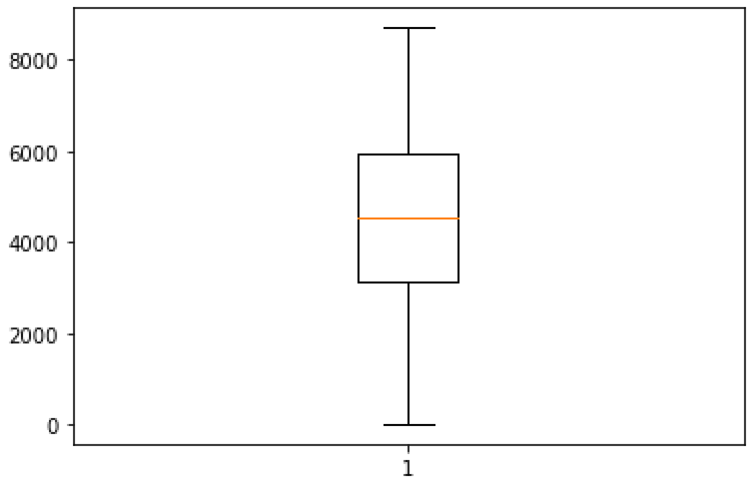
```
hum
```



```
windspeed
```



```
cnt
```



Outliears are found in humidity and windspeed variables

In [70]:

```
#calculate outliears
#calculate innerfence ,Outerfence and IQR

for i in num_var:
    print(i)
    q75, q25=np.percentile(data.loc[:,i],[75,25])
    iqr=q75-q25
    innerfence=q25 - (iqr*1.5)
    upperfence=q75 + (iqr*1.5)
    print("Innerfence : "+str(innerfence))
    print("upperfence : "+str(upperfence))
    print("IQR : ",str(iqr))



# replace outliers with NA

    data.loc[data[i]<innerfence, i] = np.nan
    data.loc[data[i]>upperfence, i] = np.nan
```

```
temp
Innerfence : -0.14041600000000015
upperfence : 1.1329160000000003
IQR :  0.3183330000000001
atemp
Innerfence : -0.06829675000000018
upperfence : 1.0147412500000002
IQR :  0.2707595000000001
hum
Innerfence : 0.20468725
upperfence : 1.0455212500000002
IQR :  0.21020850000000002
windspeed
Innerfence : -0.012446750000000034
upperfence : 0.38061125
IQR :  0.0982645
cnt
Innerfence : -1054.0
upperfence : 10162.0
IQR :  2804.0
```

In [71]:

```
data.isnull().sum()
```

Out[71]:

```
instant        0
dteday         0
season         0
yr             0
mnth           0
holiday        0
weekday        0
workingday     0
weathersit     0
temp           0
atemp          0
hum            2
windspeed     13
casual         0
registered     0
cnt            0
dtype: int64
```

15 Outliears Found

In [72]:

```
#impute NA with median
data.hum=data.hum.fillna(data.hum.median())
data.windspeed=data.windspeed.fillna(data.windspeed.median())
```

In [73]:

```
data.isnull().sum()
```

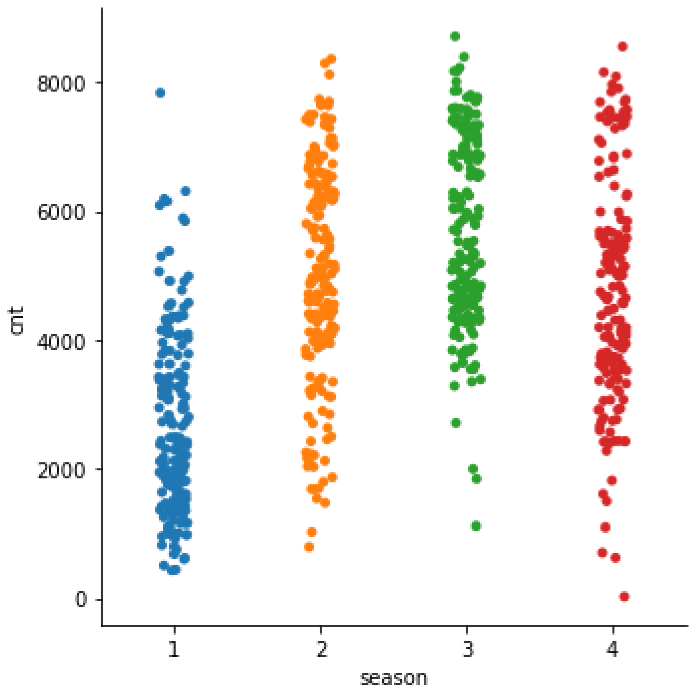
Out[73]:

```
instant       0
dteday        0
season        0
yr            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
dtype: int64
```

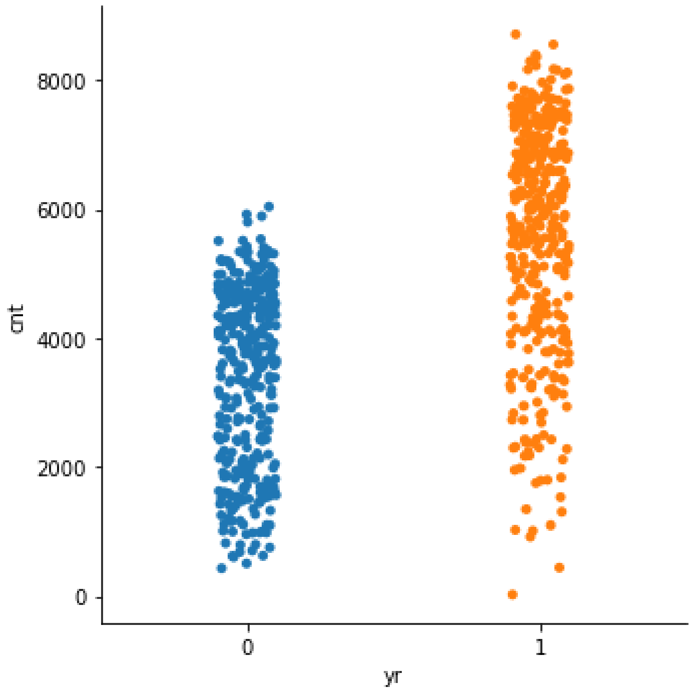
Data Analysis

In [74]:

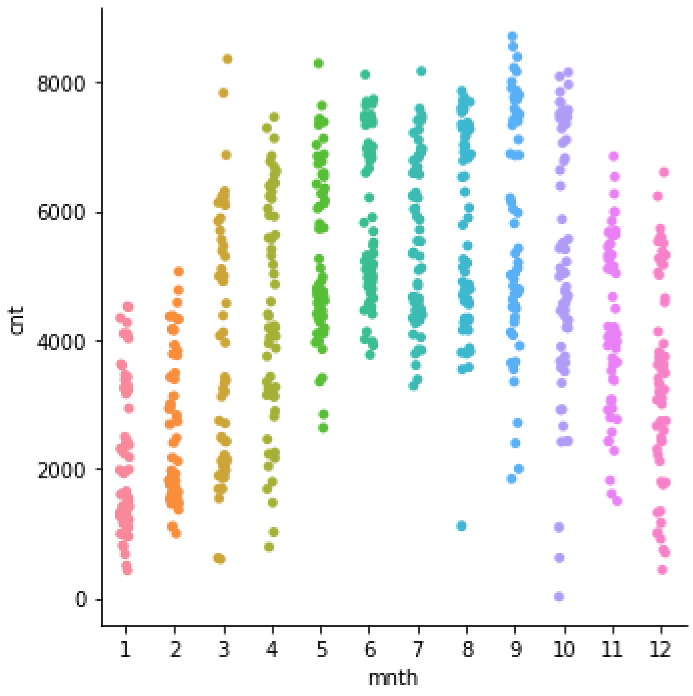
```
for i in cat_var:
    sns.catplot(x=i,y="cnt",data=data)
```

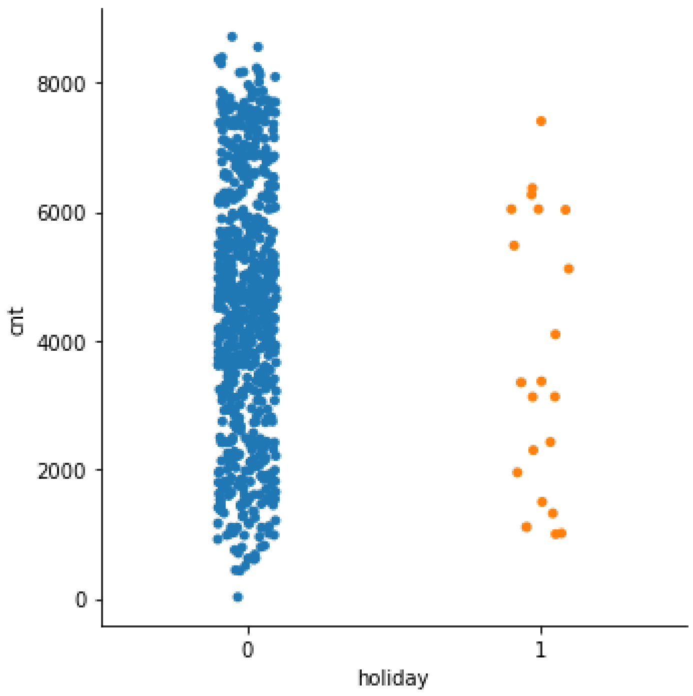


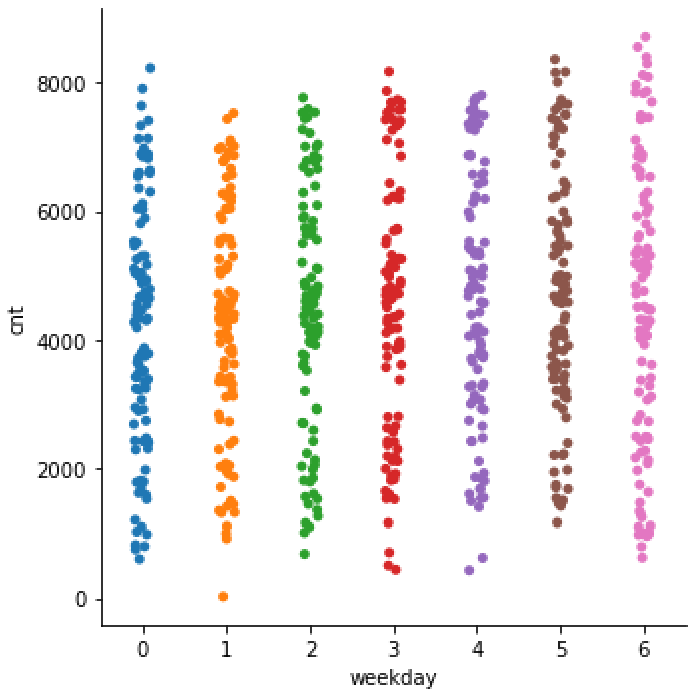
cnt
0
2000
4000
6000
8000
1
2
3
4
season

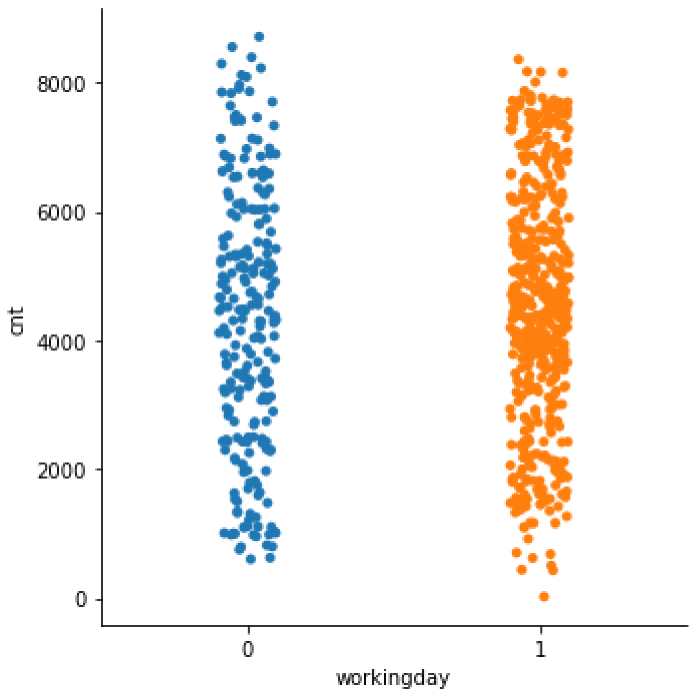


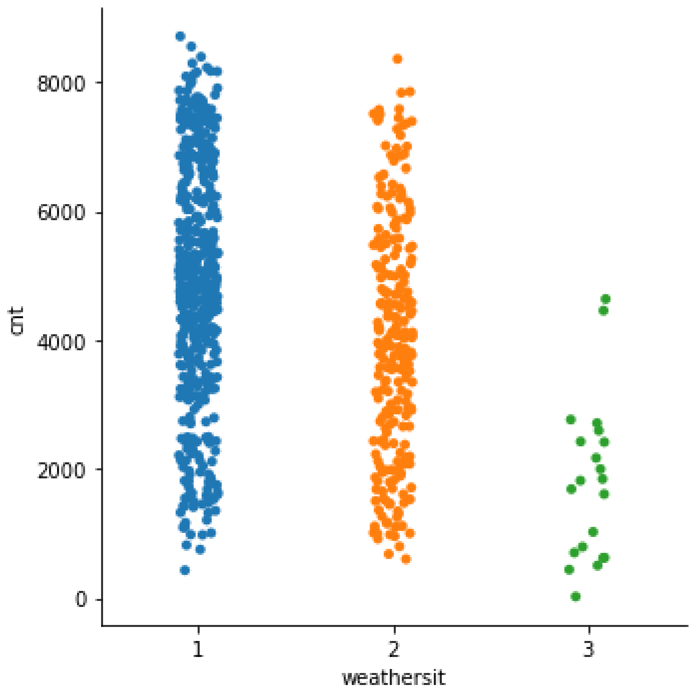
cnt
0
2000
4000
6000
8000
0
1
yr


cnt
0
2000
4000
6000
8000
1
2
3
4
5
6
7
8
9
10
11
12
mnth


cnt
0
2000
4000
6000
8000
0
1
holiday


cnt
0
2000
4000
6000
8000
0
1
2
3
4
5
6
weekday


cnt
0
2000
4000
6000
8000
0
1
workingday



It is found that

In Season 2, 3 and 4 has the highest count

In Year 1 has high count than 0

In Months 4 to 10 has got pretty good count

On holidays the count is higher compared non-holidays

In weekdays, 0 and 6 has the highest count

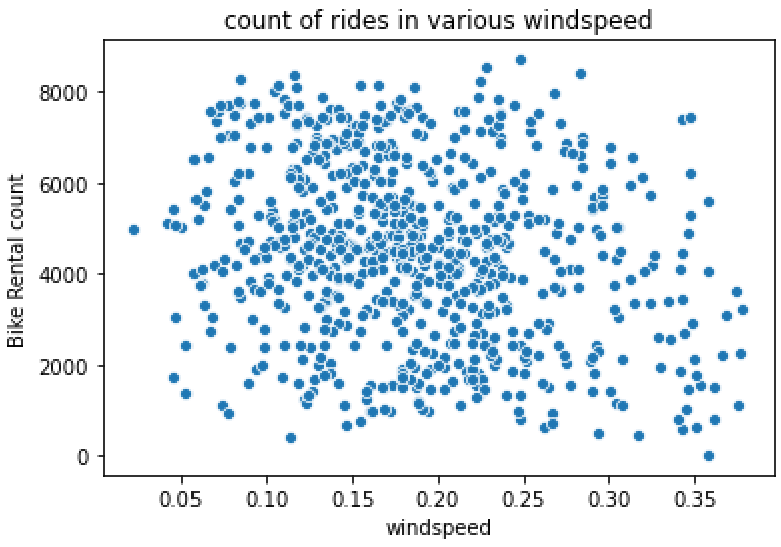
In weather, 1 has the highest count

In [75]:

```
#count of rides in various windspeed

sns.scatterplot(x="windspeed",y="cnt",data=data)
plt.title("count of rides in various windspeed")
plt.ylabel("Bike Rental count")
```

Out[75]:

```
Text(0, 0.5, 'Bike Rental count')
```

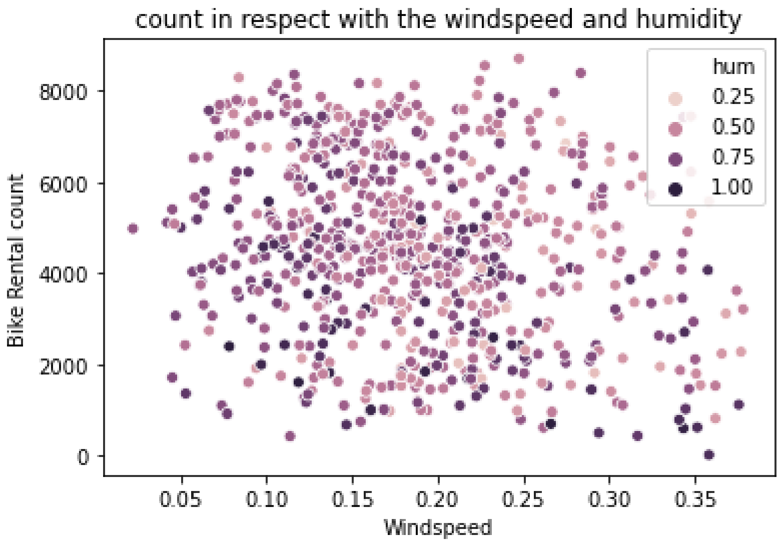


In [76]:

```
# count in respect with the windspeed and humidity

sns.scatterplot(x="windspeed",y="cnt",hue="hum",data=data)
plt.title("count in respect with the windspeed and humidity")
plt.xlabel("Windspeed")
plt.ylabel("Bike Rental count")
```

Out[76]:

```
Text(0, 0.5, 'Bike Rental count')
```



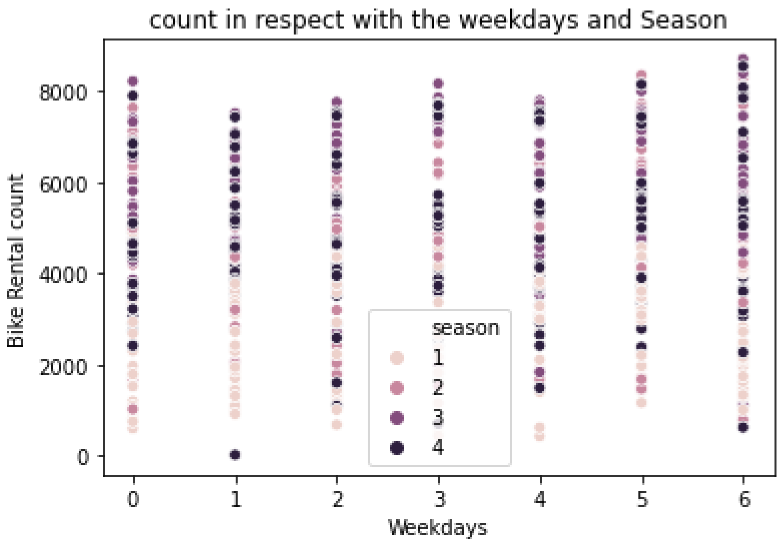
Count is high when windspeed is between 0.10 to 0.25 and humidity 0.50 to 0.75

In [77]:

```
#count in respect with the weekdays and season

sns.scatterplot(x="weekday",y="cnt",hue="season",data=data)
plt.title("count in respect with the weekdays and Season")
plt.xlabel("Weekdays")
plt.ylabel("Bike Rental count")
```

Out[77]:

```
Text(0, 0.5, 'Bike Rental count')
```



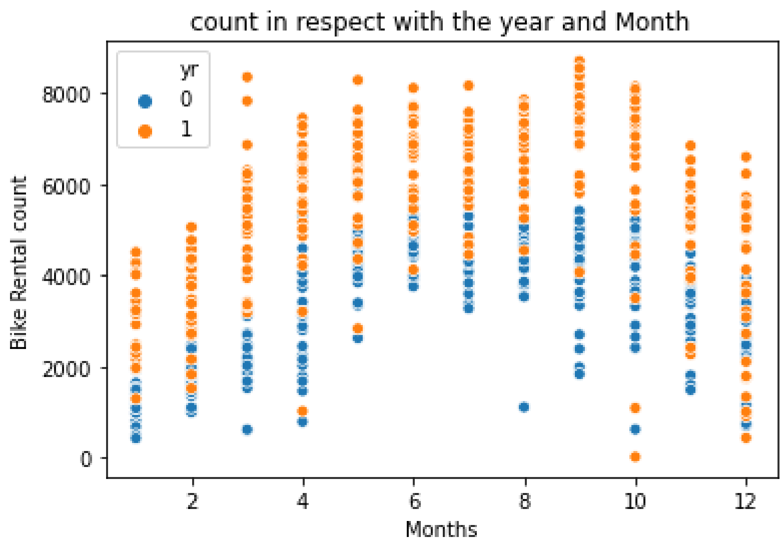
Count is high in weekday 0 , 6 and season 4 has a highest count

In [78]:

```
#count in respect with the year and month

sns.scatterplot(x="mnth",y="cnt",hue="yr",data=data)
plt.title("count in respect with the year and Month")
plt.xlabel("Months")
plt.ylabel("Bike Rental count")
```

Out[78]:

```
Text(0, 0.5, 'Bike Rental count')
```



count is high in year 1, particularly from season 3 to 12 excluding 9

Feature Selection

In [79]:

```
# Correlation Analysis and Anova test to find varaibles which can be excluded

data_cor=data.loc[:,num_var]
print(data_cor.corr())
```

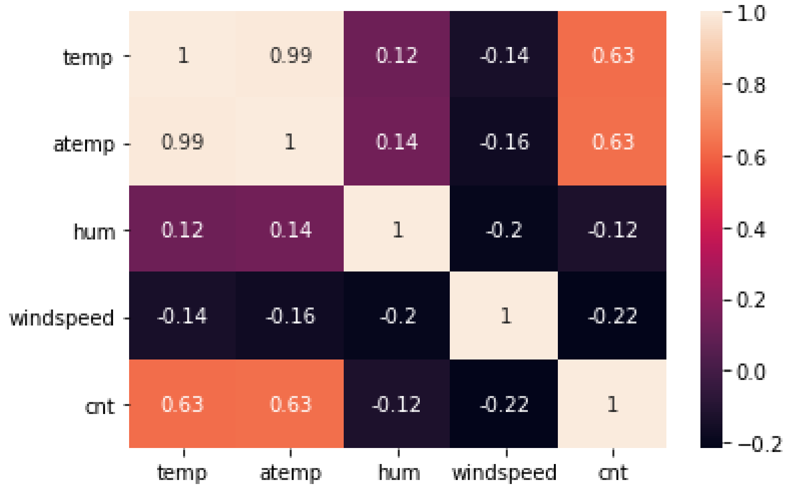
```
               temp     atemp       hum  windspeed       cnt
temp       1.000000  0.991702  0.123723  -0.138937  0.627494
atemp      0.991702  1.000000  0.137312  -0.164157  0.631066
hum        0.123723  0.137312  1.000000  -0.200237 -0.121454
windspeed -0.138937 -0.164157 -0.200237   1.000000 -0.215203
cnt        0.627494  0.631066 -0.121454  -0.215203  1.000000
```

In [80]:

```
sns.heatmap(data_cor.corr(),annot=True)
```

Out[80]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f55572e87f0>
```



From the heatmap we can see temp and a temp are highly co-related with each other

In [81]:

```
# Anova Test for checking redundant categorical variable

import statsmodels.api as sm
from statsmodels.formula.api import ols

for i in cat_var:
    mod = ols('cnt' + '~' + i, data = data).fit()
    anova_table = sm.stats.anova_lm(mod, typ = 2)
    print(anova_table)
```

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

Holiday , Weekday and Workingday has the p-value >0.05 which means we will accept Null hypothesis

In [82]:

```
#Dimension Reduction

data = data.drop(['atemp', 'holiday', 'weekday', 'workingday'],axis=1)
print(data.shape)
```

```
(731, 12)
```

In [83]:

```
# variable "instant" can be dropped as it simply represents the index
# Variable "dteday" can be ignored as output is not based on time series analysi
s
# casual and registered variables can be removed, as these two sums to dependent
variable count

data=data.drop(['instant','dteday','registered','casual'],axis=1)
```

In [84]:

```
data.head()
```

Out[84]:

	season	yr	mnth	weathersit	temp	hum	windspeed	cnt
0	1	0	1	2	0.344167	0.805833	0.160446	985.0
1	1	0	1	2	0.363478	0.696087	0.248539	801.0
2	1	0	1	1	0.196364	0.437273	0.248309	1349.0
3	1	0	1	1	0.200000	0.590435	0.160296	1562.0
4	1	0	1	1	0.226957	0.436957	0.186900	1600.0

In [85]:

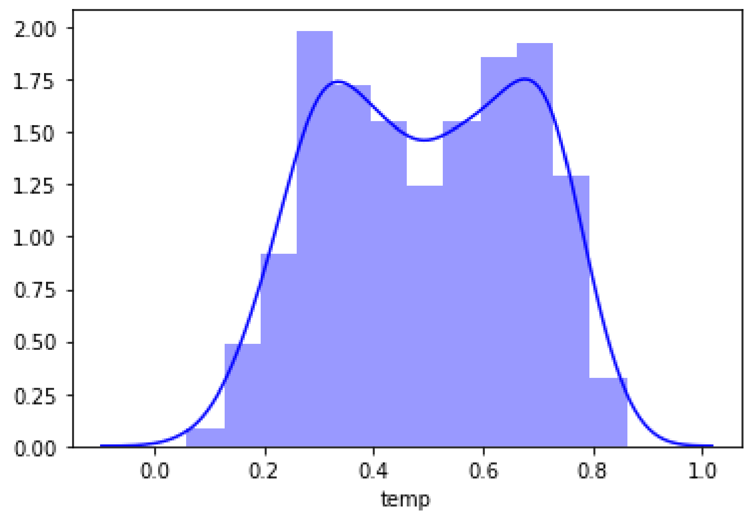
```
#updating var
num_var = ["temp","hum","windspeed","cnt"]    # numeric variables

cat_var = ["season", "yr", "mnth", "weathersit"]    # categorical variables
```

Feature Scaling

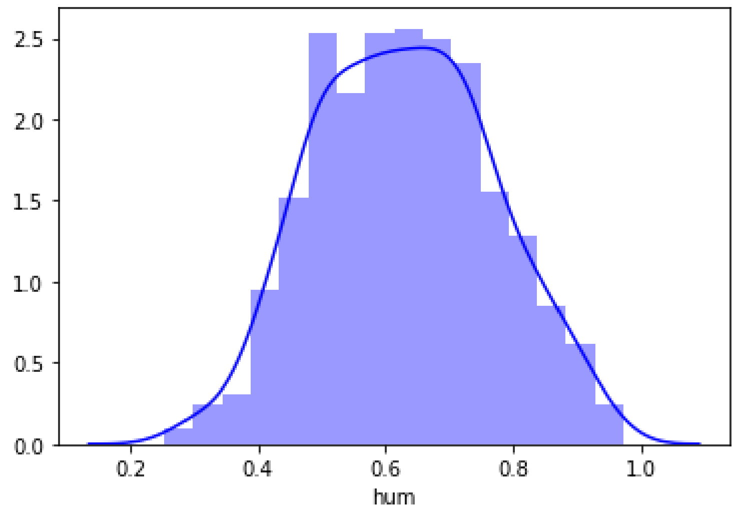
In [86]:

```
#check normality
for i in num_var:
    print(i)
    sns.distplot(data[i], bins = 'auto', color = 'blue')
    plt.show()
```

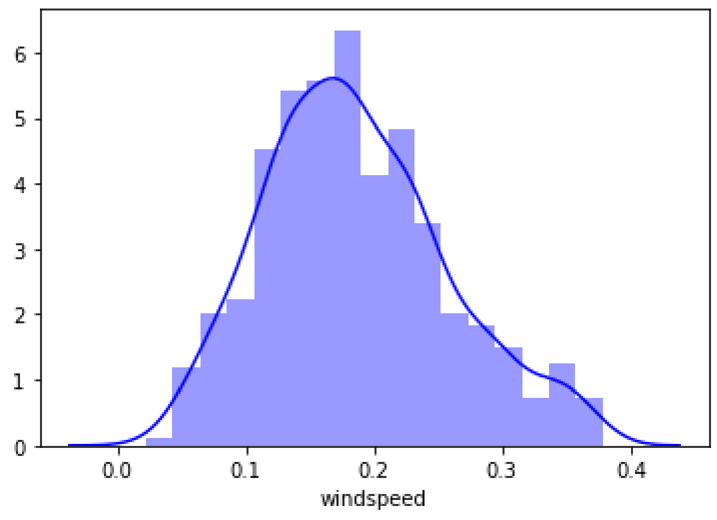
temp



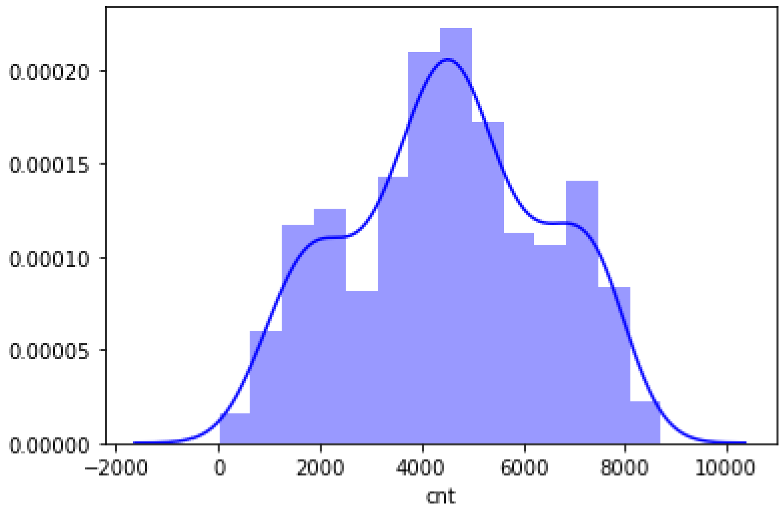
hum



windspeed



cnt



In [87]:

```
#check min max value for normalization
data.describe()
```

Out[87]:

	season	yr	mnth	weathersit	temp	hum	windspeed
count	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000	731.000000
mean	2.496580	0.500684	6.519836	1.395349	0.495385	0.629354	0.186257
std	1.110807	0.500342	3.451913	0.544894	0.183051	0.139566	0.071156
min	1.000000	0.000000	1.000000	1.000000	0.059130	0.254167	0.022392
25%	2.000000	0.000000	4.000000	1.000000	0.337083	0.522291	0.134950
50%	3.000000	1.000000	7.000000	1.000000	0.498333	0.627500	0.178802
75%	3.000000	1.000000	10.000000	2.000000	0.655417	0.730209	0.229786
max	4.000000	1.000000	12.000000	3.000000	0.861667	0.972500	0.378108

data is normalized, No need of scaling

In [88]:

```
data = pd.get_dummies(data, columns = cat_var,drop_first=True)
```

In [89]:

```
data.head()
```

Out[89]:

	temp	hum	windspeed	cnt	season_2	season_3	season_4	yr_1	mnth_2	m
0	0.344167	0.805833	0.160446	985.0	0	0	0	0	0	
1	0.363478	0.696087	0.248539	801.0	0	0	0	0	0	
2	0.196364	0.437273	0.248309	1349.0	0	0	0	0	0	
3	0.200000	0.590435	0.160296	1562.0	0	0	0	0	0	
4	0.226957	0.436957	0.186900	1600.0	0	0	0	0	0	

5 rows × 21 columns

In [90]:

```
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from scipy.stats.stats import pearsonr
```

In [91]:

```
#predictors and trget var
X=data.drop('cnt',axis=1)
Y=data['cnt']
```

In [92]:

```
#devide the data into test and train
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.20,random_state=0
)
```

In [93]:

```
#define Error Metrics.

def MAPE(y_actual, y_predicted):
    MAPE = np.mean(np.abs(y_actual-y_predicted)/y_actual)*100
    return MAPE

def Rsquare(y_actual, y_predicted):
    Rsquare = np.corrcoef(y_actual,y_predicted)**2
    return Rsquare
```

Desicion Tree

In [95]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(max_depth=2).fit(X_train,Y_train)
```

In [96]:

```
#prediction
pred=dt.predict(X_test)
```

In [100]:

```
#Mean absolute percentage error
mape=MAPE(Y_test,pred)
```

In [105]:

```
#RSquare
rsquare=Rsquare(Y_test,pred)
rs_data = rsquare.ravel()
new_rscore = float(rs_data[1])
```

In [112]:

```
print("Mape: "+str(mape))
print("rsquare: "+str(new_rscore))
print("Accuracy: "+str(100-mape))
```

```
Mape: 36.94809301452646
rsquare: 0.6544606873373328
Accuracy: 63.05190698547354
```

Random Forest

In [125]:

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100).fit(X_train,Y_train)
```

In [127]:

```
#prediction

rf_pred=rf.predict(X_test)
```

In [128]:

```
#Mean absolute percentage error
mape=MAPE(Y_test,rf_pred)
```

In [130]:

```
#RSquare
rsquare=Rsquare(Y_test,rf_pred)
rs_data = rsquare.ravel()
new_rscore = float(rs_data[1])
```

In [131]:

```
print("Mape: "+str(mape))
print("rsquare: "+str(new_rscore))
print("Accuracy: "+str(100-mape))
```

```
Mape: 21.586269848650655
rsquare: 0.8783496338171791
Accuracy: 78.41373015134934
```

LINEAR REGRESSION MODEL

In [118]:

```
import statsmodels.api as sm
lr= sm.OLS(Y_train, X_train).fit()
print(lr.summary())
```

```
                                 OLS Regression Results
====================================================================
===================
Dep. Variable:                    cnt   R-squared (uncentered):
0.972
Model:                            OLS   Adj. R-squared (uncentered):
0.971
Method:                 Least Squares   F-statistic:
991.4
Date:                Sat, 13 Mar 2021   Prob (F-statistic):
0.00
Time:                        15:48:44   Log-Likelihood:
-4741.0
No. Observations:                 584   AIC:
9522.
Df Residuals:                     564   BIC:
9609.
Df Model:                          20
Covariance Type:            nonrobust
====================================================================
============
                   coef    std err          t      P>|t|      [0.025
0.975]
--------------------------------------------------------------------
------------
temp          5479.8743    487.663     11.237      0.000    4522.016
6437.732
hum            -89.2293    260.052     -0.343      0.732    -600.018
421.560
windspeed     -618.5495    434.273     -1.424      0.155   -1471.540
234.441
season_2       872.0019    213.758      4.079      0.000     452.143
1291.861
season_3       870.1890    265.120      3.282      0.001     349.445
1390.933
season_4      1539.5301    221.967      6.936      0.000    1103.548
1975.513
yr_1          2035.3923     69.566     29.259      0.000    1898.753
2172.032
mnth_2         369.6319    170.870      2.163      0.031      34.013
705.251
mnth_3         748.4690    193.416      3.870      0.000     368.566
1128.372
mnth_4         374.0911    293.095      1.276      0.202    -201.599
949.781
mnth_5         644.8810    316.651      2.037      0.042      22.922
1266.840
mnth_6         338.6062    340.553      0.994      0.321    -330.300
1007.513
mnth_7        -176.5674    385.423     -0.458      0.647    -933.608
580.473
mnth_8         268.9384    368.682      0.729      0.466    -455.218
993.095
mnth_9         874.6931    330.169      2.649      0.008     226.182
1523.205
mnth_10        523.9591    294.263      1.781      0.076     -54.026
1101.944
mnth_11         81.3859    276.991      0.294      0.769    -462.674
625.446
mnth_12        215.3398    222.679      0.967      0.334    -222.042
652.722
```

```
weathersit_2  -557.7488     87.841     -6.349      0.000    -730.285
-385.213
weathersit_3 -2488.7627    238.663    -10.428      0.000   -2957.540
-2019.985
======================================================================
==========
Omnibus:                       96.788   Durbin-Watson:
1.916
Prob(Omnibus):                  0.000   Jarque-Bera (JB):
228.026
Skew:                          -0.872   Prob(JB):
3.05e-50
Kurtosis:                       5.515   Cond. No.
31.1
======================================================================
==========

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors
is correctly specified.
```

In [119]:

```python
#Prediction

lr_pred = lr.predict(X_test)
```

In [120]:

```python
#Mean absolute percentage error
mape=MAPE(Y_test,lr_pred)
```

In [121]:

```python
#RSquare
rsquare=Rsquare(Y_test,lr_pred)
rs_data = rsquare.ravel()
new_rscore = float(rs_data[1])
```

In [140]:

```python
print("Mape: "+str(mape))
print("rsquare: "+str(new_rscore))
print("Accuracy: "+str(100-mape))
```

```
Mape: 21.586269848650655
rsquare: 0.8783496338171791
Accuracy: 78.41373015134934
```

In [144]:

```python
#Sample Input
LRModel.predict([[0.5, 0.6, 0.7,2,0,0,0,1,0,1,0,1,0,1,0,0,0,0,0,1]])
```

Out[144]:

```
array([2859.92368684])
```

Putting all the variables humidity, weather, temperature , season, month and year, is found that for those particular input we got above result