Predicting Bike Renting

Sarang Madhukar Kapse

26 January 2019

Contents

Chapter 1

Introduction

1.1 Problem Statement

The problem statement is to predict the bike rental count on daily based on the enivironmental and seasonal setting.

The aim of this project would be to choose an appropriate model which would predict the count of bike according to variations in seasonal and environmental setting. The seasonal setting would be the 4 different season (1:spring, 2:summer, 3:fall, 4:winter). The environment setting would be weather situation , temperature , humidity and wind speed.

1.2 Dataset

The dataset contains 16 variables and 731 observations.

Dataset Details:

Number of Variables: 16

Variable Information:

instant: Record index
dteday: Date
season: Season (1:springer, 2:summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
holiday: weather day is holiday or not (extracted fromHoliday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted fromFreemeteo)
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
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)
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)
hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)
casual: count of casual users
registered: count of registered users
cnt: count of total rental bikes including both casual and registered

After data preprocessing and calculating the importance the number of variable reduces to 8 out of which 7 are independent and 1 is our target variable the following table shows the dataset for predicting the losses.

Independent Variables	Dependent Variable
Instant	**cnt**
Season	
Mnth	
Atemp	
Hum	
Windspeed	
Casual	

Table 1.2.1 Variables under predictions

Chapter 2

Methodology

2.1 Pre Processing

Any dataset which we want to use for building model is explored and necessary steps are taken after exploring the dataset which involves cleaning the missing values, detecting the outliers, normalizing the data observation and deleting variables which are multi-collinear and running the chi square and regression test to detect and delete variables which don't help us to predict our target variable. The preprocessing steps used on the dataset are shown by the following table.

Steps	Pre processing technique
Step 1	Changing the required data types
Step 2	Detect missing values and impute with KNN
Step 3	Box plot distribution and outlier check
Step 4	Detect outliers and impute with KNN
Step 5	Feature selection with Co-relation plot and Chi square test .
Step 6	Normality check and normalizing the dataset observation

Table 2.1 Preprocessing steps

Step 1: Changing the require Data types

After loading our data set and running the str(data_bike) command in R we can see that most of the variables which are actually factorial is named as integer and some of the variables which are integer need to be changed to numeric. In order to perform KNN imputation on our missing values we need to change the data types as the KNN imputation method works on the concept of replacing values according to the data types.

```
> str(data_bike)
'data.frame':    731 obs. of  16 variables:
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ 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 ..
```

The marked variables need to be changed as these are factorial variables but defined as integer variable in the dataset. While some of the integer variable which are integer need to be converted to numeric to smoothen the further preprocessing technique.

After using this command in R the required data types were changed as follows.

```
data_bike$season = as.factor(data_bike$season)
data_bike$yr = as.factor(data_bike$yr)
data_bike$mnth = as.factor(data_bike$mnth)
data_bike$holiday = as.factor(data_bike$holiday)
data_bike$weekday = as.factor(data_bike$weekday)
data_bike$workingday = as.factor(data_bike$workingday)
data_bike$weathersit = as.factor(data_bike$weathersit)
data_bike$casual = as.numeric(data_bike$casual)
data_bike$registered = as.numeric(data_bike$registered)
data_bike$cnt = as.numeric(data_bike$cnt)
data_bike$windspeed = as.numeric(data_bike$windspeed)
```

Step 2 : Detect missing values and impute with KNN

After changing the required data types, we proceed to our next step which is detecting the missing values . In our data set there are no missing values.
This is shown by the following R command.

missing_val

	Columns	Missing_percentage
1	instant	0
2	dteday	0
3	season	0
4	yr	0
5	mnth	0
6	holiday	0
7	weekday	0
8	workingday	0
9	weathersit	0
10	temp	0
11	atemp	0
12	hum	0
13	windspeed	0
14	casual	0
15	registered	0
16	cnt	0

.

Step 3: Box plot distribution and outlier check

A dataset containing outliers can greatly affect the model developement process. The outlier observations in our dataset is detected graphically using box plot method the following figure shows the box plot distribution for the continuous variables.

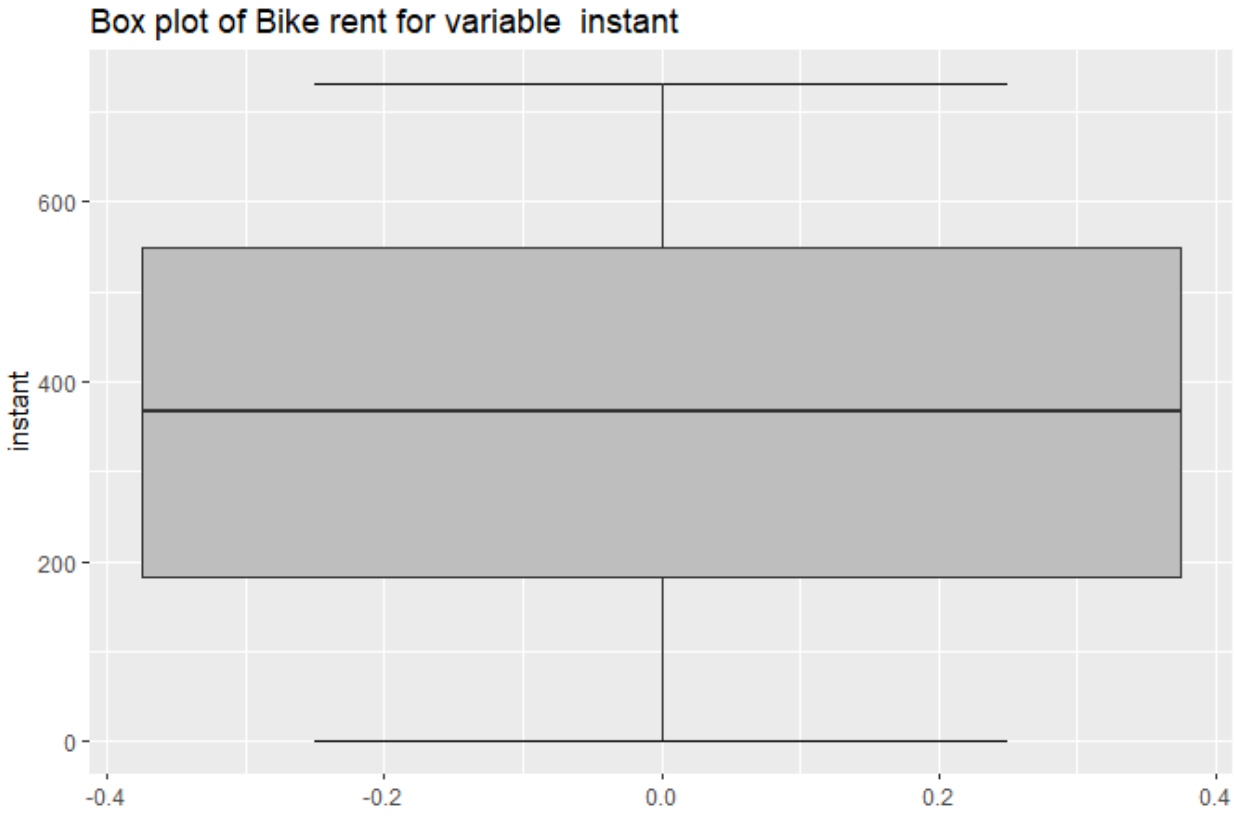

Figure 2.1.2 Box plot for Variable Instant

The above box plot shows that there are no outliers.

.

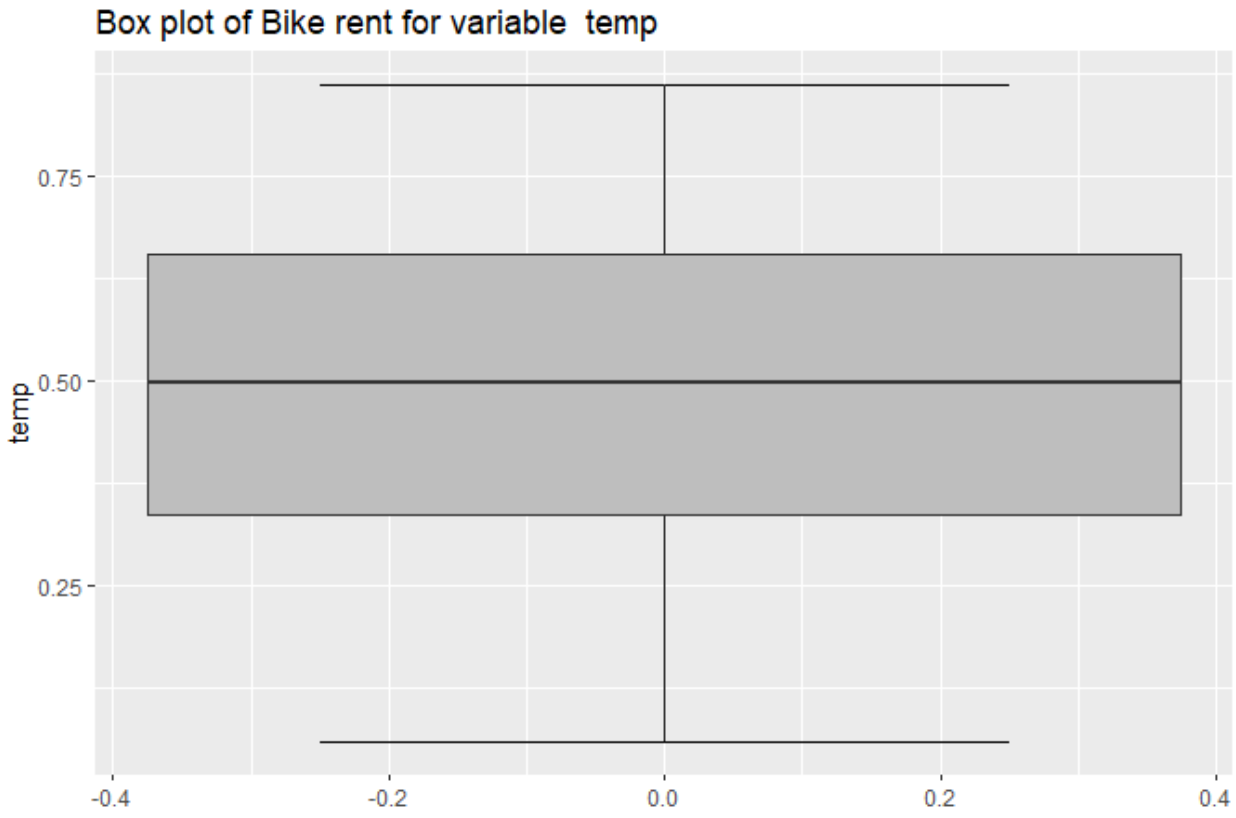

Figure 2.1.3 Box Plot Variable temp

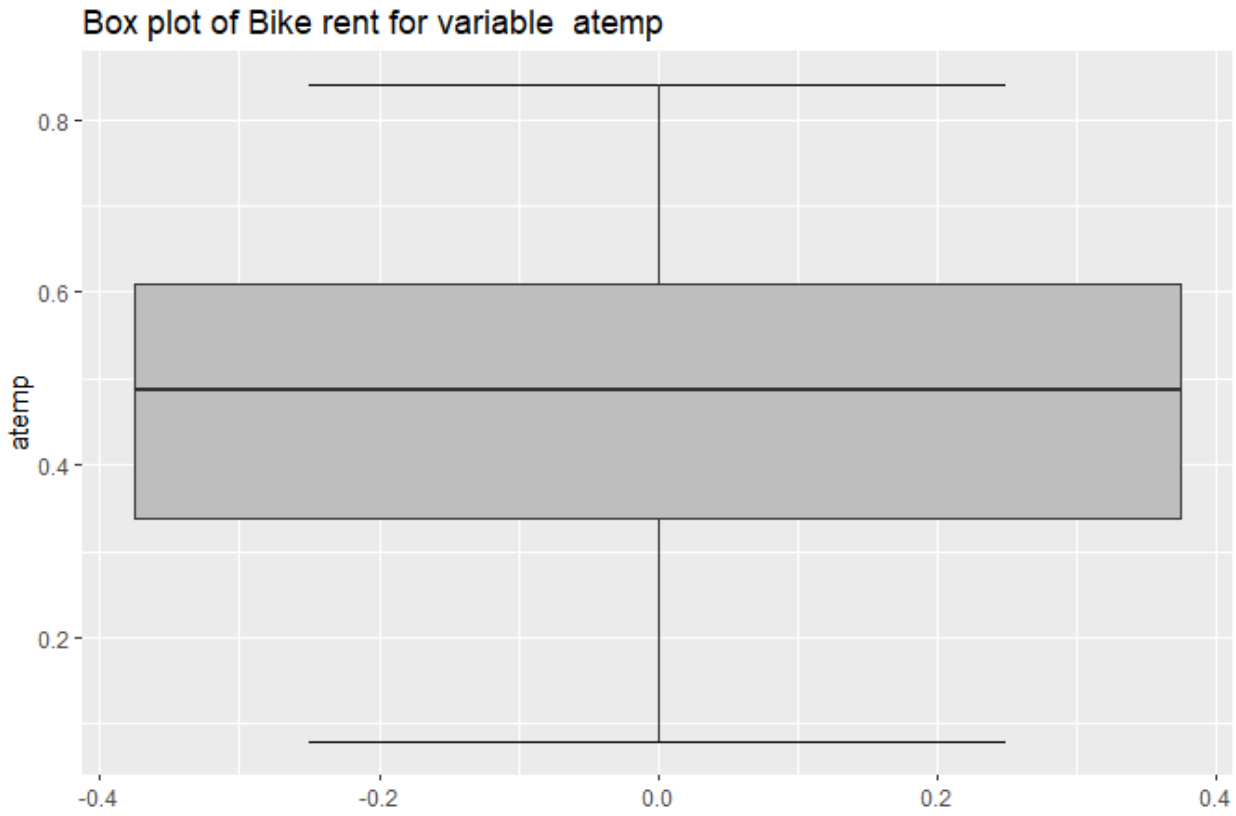

Figure 2.1.4 Box Plot for variable atemp (atmospheric temp)

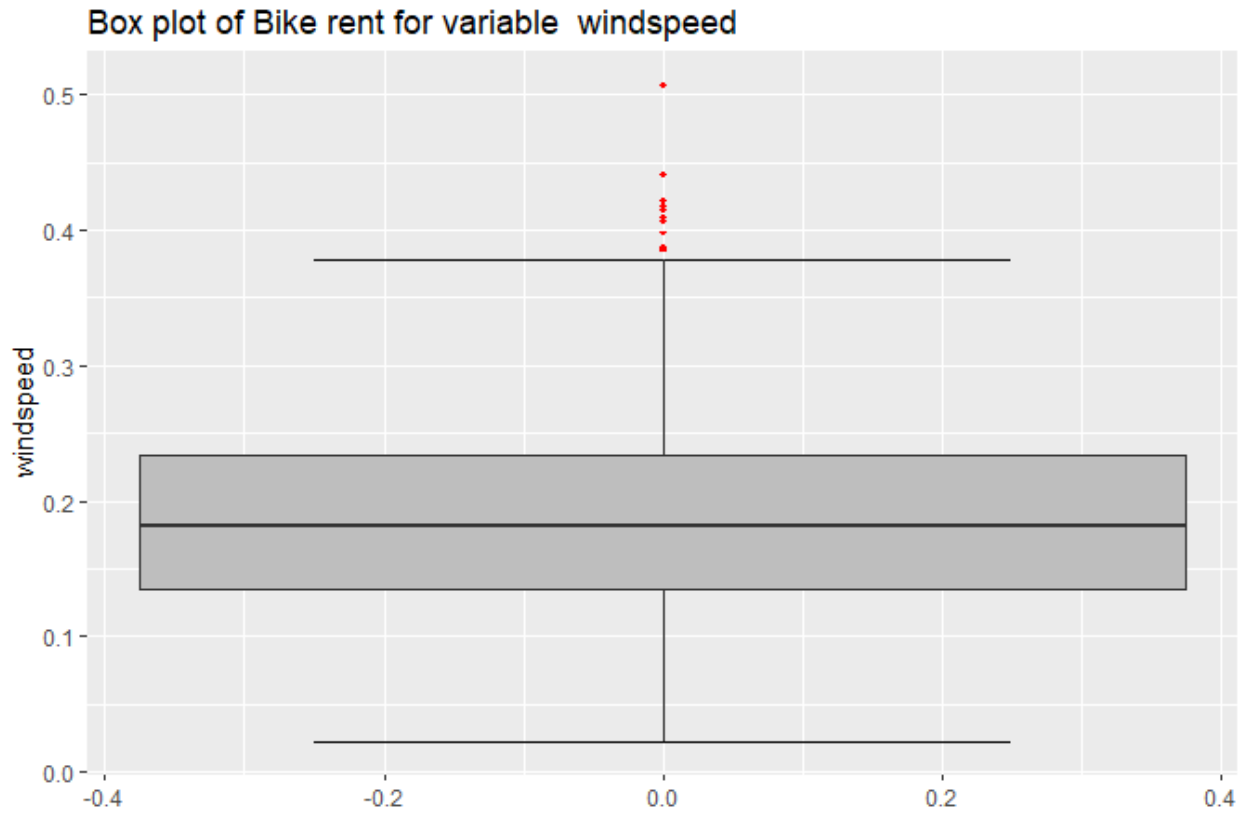

Figure 2.1.5 Box plot for Variable windspeed

The Boxplot for variable windspeed shows that variable windspeed has outliers shown by red dots.

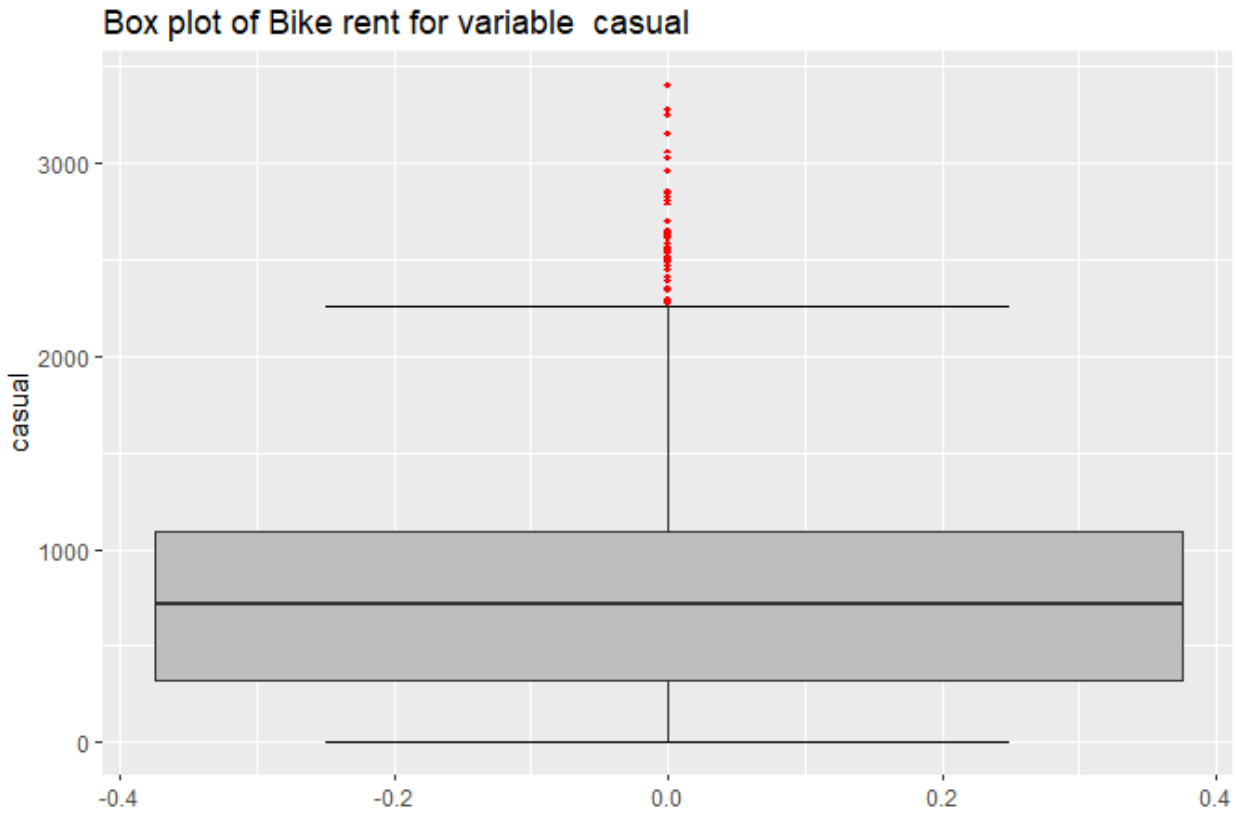

Figure 2.1.6 Box Plot for Variable Casual

The boxplot shows that there are outliers.

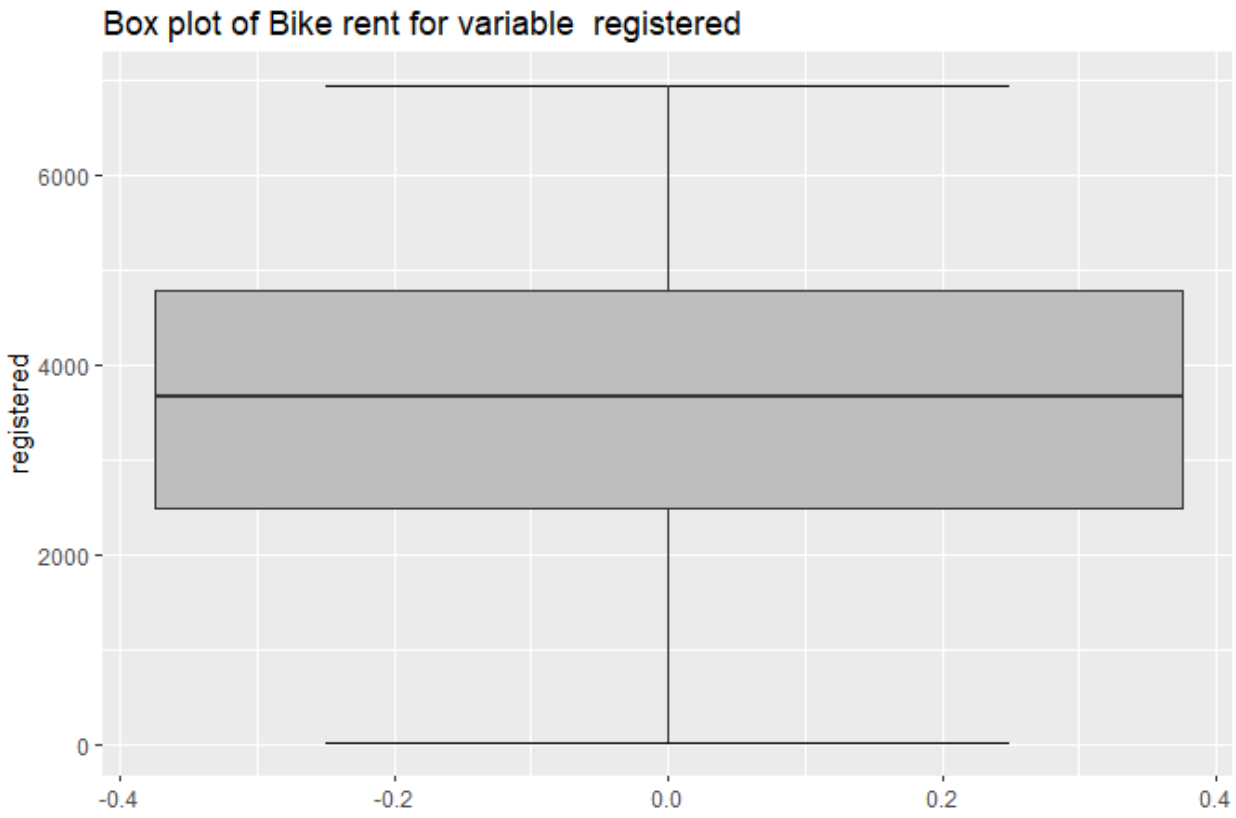

Figure 2.1.7 Box plot for Variable registered

The boxplot for the variable registered has no outliers.

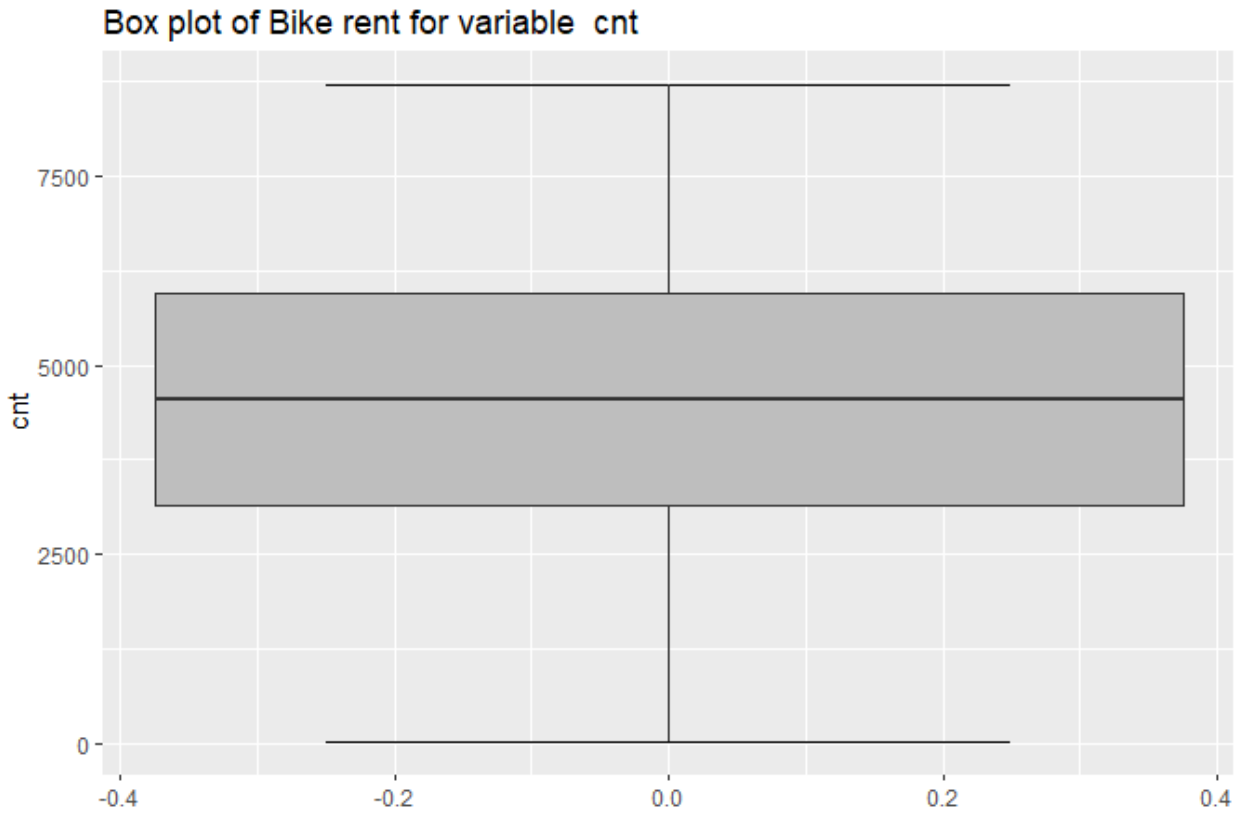

Figure 2.1.8 Box Plot for Variable cnt (total count of users)

The boxplot for total count variable has no outliers.

The above box plot detection method shows that there are some points in the variables of our dataset which need to be removed or impute as they lie above the upper fence. As the deletion process would delete the entire rows containing outliers in our dataset the method that was used to treat this outliers was KNN imputation method .

############## loop to remove outlier and impute using KNN############

```
for(i in cnames_bike)
{val = data_bike[,i][data_bike[,i] %in% boxplot.stats(data_bike[,i])$out]
#print(length(val))
data_bike[,i][data_bike[,i] %in% val] = NA
}

data_bike = knnImputation(data_bike, k = 5)
```

Step 5 : Feature selection with Correlation plot and Chi square test .

In order to start our model development process we also need to see co linearity issues in our dataset for that we need to see the correlation plot and check the variables which are collinear. Collinear variables are those variables which carry the same information and don't add any extra information but the same information as their partner variables we need to detect these variables and remove them as during our model building and predicting the test data these consume unnecessary space and affect our predictions. The following figure shows the correlation plot.

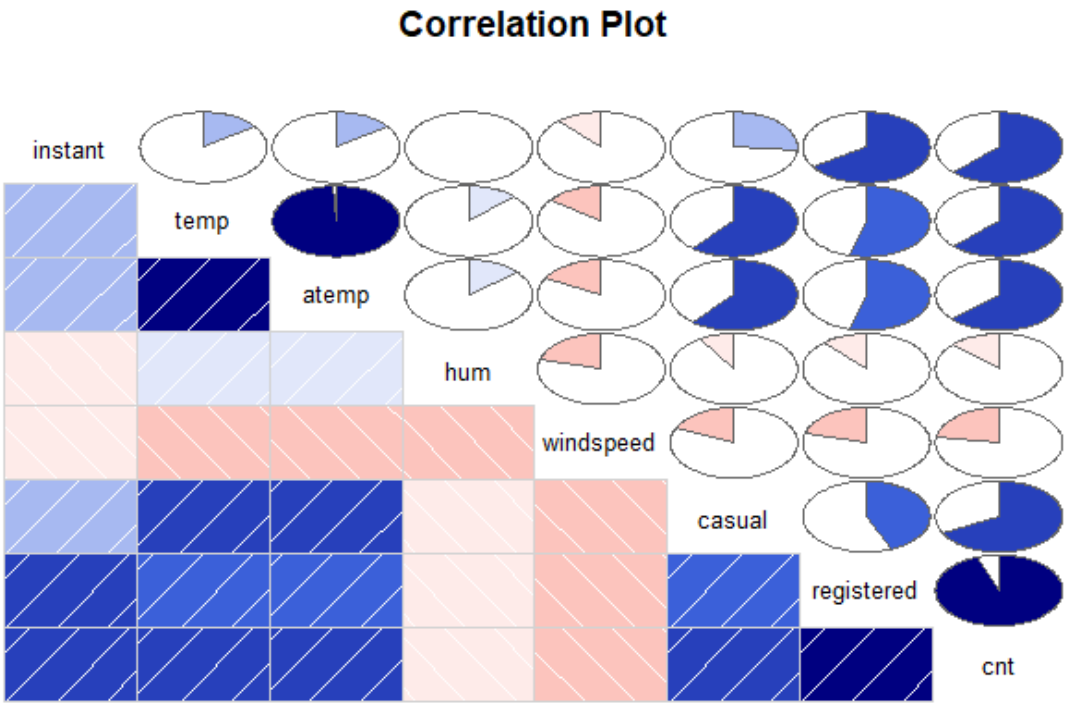

Figure 2.1.11 Correlation Plot

From the correlation plot it can be seen that the four variables are having high association or relationship , which is shown by the circle in dark blue colour. So we can remove one of the variables from temp and atemp . Similarly we can remove one of the variable from registered and cnt. In our case we have removed temp and registered user from our dataset. Note that the correlation plot is only used for continuous variable.

Chi square Test of Independence :
The correlation plot is used to check the relationship between two continuous variable but what about the variables which are factors in such cases we need to use the chi square test of Independence. The chi square test is based on two assumptions Null and Alternate hypothesis after running this test we need to select one of our assumptions. In our case we are selecting Seasons as our target variable and the rest of the variables as our Independent or predictor variable. The reason for selecting season as our target variable is the fact that problem statement wants our prediction to be based on seasonal setting.

Two assumptions:

1. Null Hypothesis:

Target variable and the Predictor variables are independent.

2. Alternate Hypothesis:

Target Variable and the Predictor variables are not independent i.e. they are dependent.

After running the test we get the results in form of X-squared , df , p-value . From these values we need to see the parameter p-value . If the p-value is less than 0.05 then we need to reject null hypothesis. In short if P value is greater than 0.05 then we need to drop that variable from our data set.

The results of chi square test are as follows:

[1] "dteday"

Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 2193, df = 2190, p-value = 0.4779

[2] "weathersit"

Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 14.884, df = 6, p-value = 0.02118

[3] "yr"

Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 0.0041569, df = 3, p-value = 0.9999

[4] "mnth"

Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 1765.1, df = 33, p-value < 2.2e-16

[5] "holiday"

	Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 1.4961, df = 3, p-value = 0.6832

[6] "weekday"

	Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 0.39925, df = 18, p-value = 1

[7] "workingday"

	Pearson's Chi-squared test

data: table(factor_data_2$season, factor_data_2[, i])
X-squared = 0.64285, df = 3, p-value = 0.8866

From the correlation plot and chi square test we can select variables which are necessary for model selection and development and reduce the dimension of our dataset. The following command is used for dimension reduction.

################## Dimension Reduction###############

```
data_bike = subset(data_bike,
                   select = -c(dteday, weathersit, yr, holiday, weekday, workingday, temp, registered ))
```

After the dimension reduction we get the dataset with 8 variables and 731 observation.

Following are the variables

1. Instant
2. Seasons
3. mnth
4.atemp
5. hum
6. windspeed
7. Casual
8.Cnt (total count)

Step 6 : Normality check and normalizing the dataset observation

A normalized data is very useful for model development process. It is the process of reducing unwanted variation either within or between variables. In our data set let us see if there is variation within our variable. For our atemp (atmospheric temperature) variable the following bar graph shows that data is normally distributed.

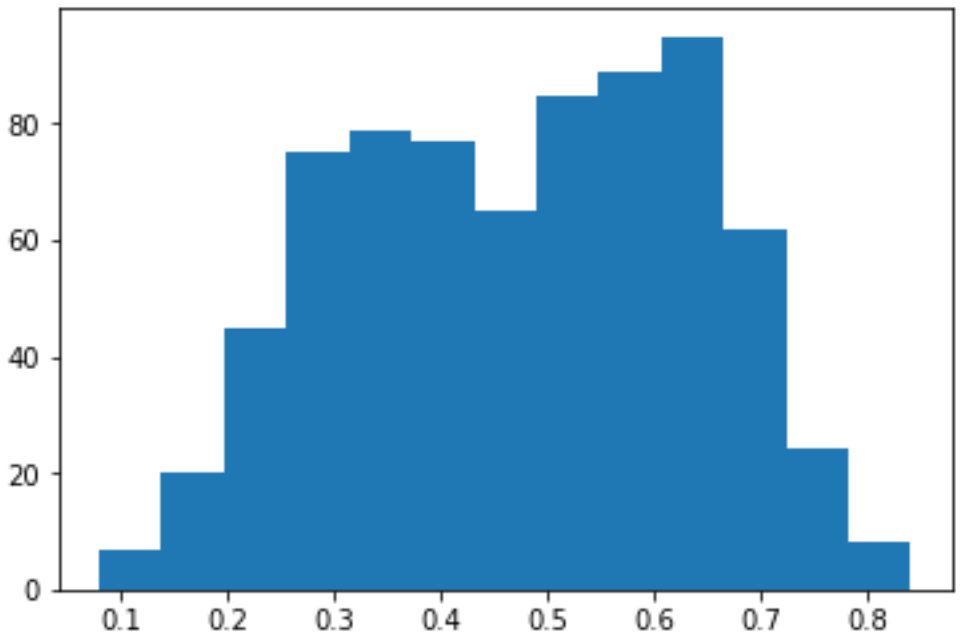

Figure 2.1.12 Histogram for atemp

The above histogram for transportation atemp shows that the data is distribution is normalized.

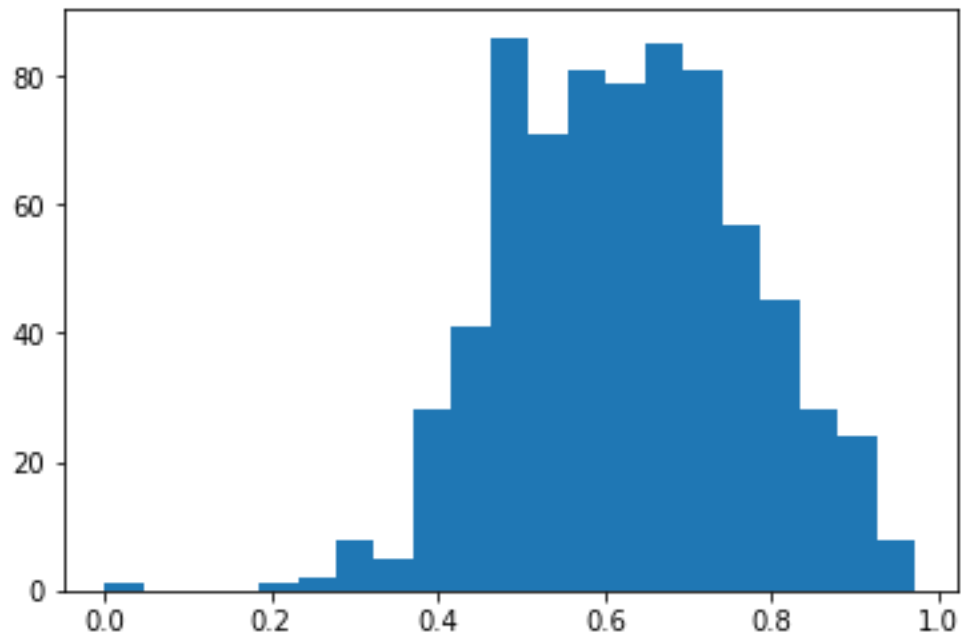

Figure 2.1.13 Histogram for Humidity

The histogram shows that the datapoint are right skewed and needs to be normalized.

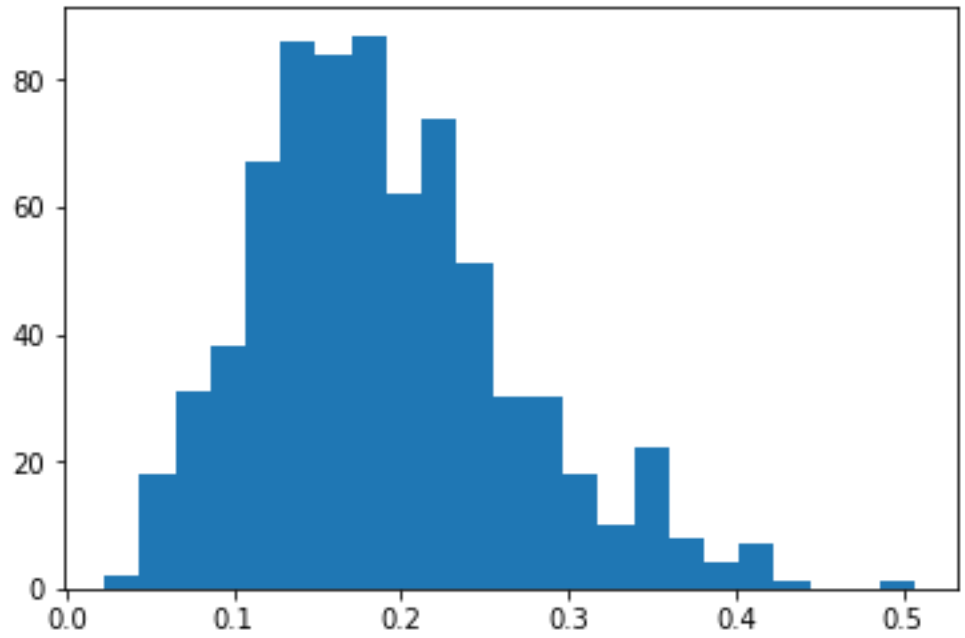

Figure 2.1.14 Histogram for Windspeed
The histogram shows data points are left skewed.

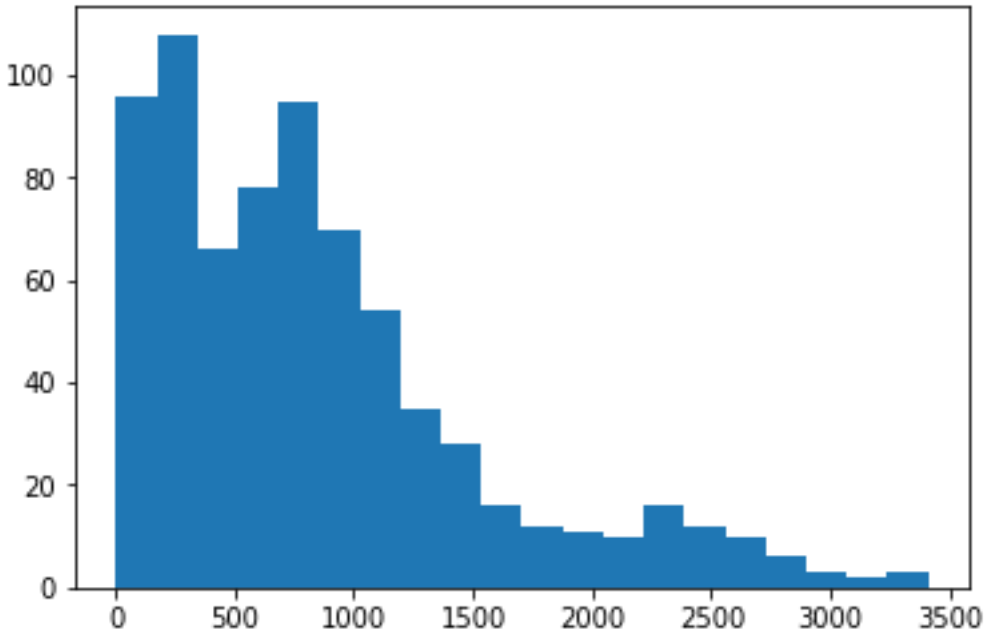

Figure 2.1.15 Histogram for casual

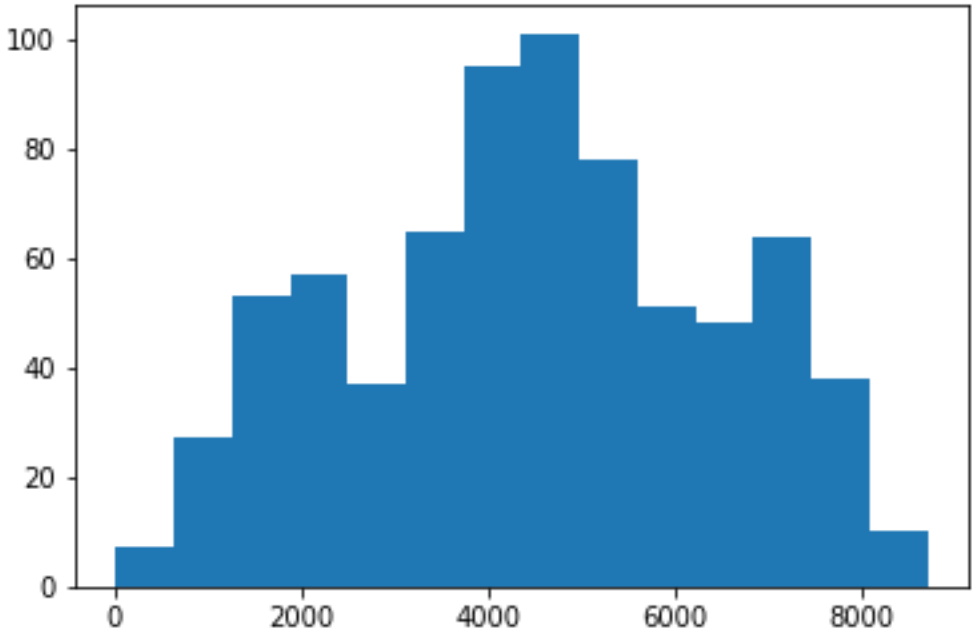

Figure 2.1.16 Histogram for Total Count

For normalizing the dataset we used the following R code

```
numeric_index_norm = sapply(data_bike,is.numeric) #selecting only numeric

numeric_data_norm = data_bike[,numeric_index_norm]

cnames_absent_norm = colnames(numeric_data_norm)

cnames_norm = c("atemp","hum","windspeed","casual","cnt")

for(i in cnames_norm){
  print(i)
  data_bike[,i] = (data_bike[,i] - min(data_bike[,i]))/
    (max(data_bike[,i] - min(data_bike[,i])))
}
```

The range of normalized data observation is from 0 to 1.

2.2 Modeling

2.2.1 Model Selection

During our preprocessing steps we realized that our dataset has both Categorical and Continuous variable . But our target variable Total Bike count is continuous variable and we need to predict this variable after selecting our model. In our case Decision tree for regression and Linear regression model were used. From the two models we will compare the predictions and select the better one between these models.

2.2.2 Decision tree for regression

Decision tree is a very simple model which is based on a branching series of Boolean test. It can be used for both classification and regression
In our case the target variable cnt (Total count) is continuous variable so we would use decision tree for regression.

Let us see how the decision tree is used for predicting the test cases.

After running the following code of chunk in R ,

```
#########################Decision tree  for regression ####################

#########rpart for regression ####################
fit_dt = rpart(cnt~ ., data = df, method = "anova")

#Predict for new test cases
#Predict for new test cases
predictions_DT = predict(fit_dt, test[,-8])

RMSE(predictions_DT, test$cnt)
```

The RMSE value is found to be 0.08950684

2.2.3 Linear Regression

Before going for the prediction of test cases in regression model, first we would check the variance inflation factor and the multi-co linearity factor for our dataset the VIF . The following result help us to know whether our dataset is suitable for model building.

Multicollinearity:
vifcor(numeric_data[, -6], th = 0.9)
No variable from the 5 input variables has collinearity problem.

The linear correlation coefficients ranges between:
min correlation (hum ~ instant): -0.0007681181
max correlation (casual ~ atemp): 0.6066318

---------- VIFs of the remained variables --------

	Variables	VIF
1	instant	1.079138
2	atemp	1.681812
3	hum	1.126505
4	windspeed	1.100194
5	casual	1.774077

All the VIF values are below 10 so we can safely say that there are no multicollinearity issues.

Now let us move towards our model building process. The result of our linear regression model building on the train data is as follows

Test result of linear regression:

```
regr.eval(test[,'cnt'] , predictions_LR ,stats = c('mae','rmse','mape','mse'))
       mae       rmse       mape        mse
0.067207463 0.089729757 0.202162924 0.008051429
```

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

Residuals:
    Min      1Q  Median      3Q     Max
-0.49955 -0.04609  0.01250  0.05739  0.28740

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.079e-01  2.324e-02   8.948  < 2e-16 ***
instant      5.858e-04  2.149e-05  27.252  < 2e-16 ***
season2      7.020e-02  2.530e-02   2.774  0.00571 **
season3      8.323e-02  2.894e-02   2.876  0.00418 **
season4      1.745e-01  2.394e-02   7.290 1.05e-12 ***
mnth2       -3.988e-03  1.962e-02  -0.203  0.83896
mnth3        1.110e-02  2.173e-02   0.511  0.60950
mnth4       -8.433e-03  3.473e-02  -0.243  0.80821
mnth5        2.066e-02  3.642e-02   0.567  0.57069
mnth6       -2.826e-02  3.758e-02  -0.752  0.45243
mnth7       -1.152e-01  4.115e-02  -2.800  0.00529 **
mnth8       -6.577e-02  3.986e-02  -1.650  0.09952 .
mnth9       -3.713e-02  3.560e-02  -1.043  0.29746
mnth10      -1.318e-01  3.241e-02  -4.066 5.47e-05 ***
mnth11      -2.108e-01  3.150e-02  -6.692 5.32e-11 ***
mnth12      -2.147e-01  2.533e-02  -8.475  < 2e-16 ***
atemp        3.631e-01  4.387e-02   8.276 9.22e-16 ***
hum         -2.326e-01  2.291e-02 -10.155  < 2e-16 ***
windspeed   -1.181e-01  2.050e-02  -5.761 1.37e-08 ***
casual       1.898e-01  2.239e-02   8.474  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.09129 on 564 degrees of freedom
Multiple R-squared:  0.8342,        Adjusted R-squared:  0.8286
F-statistic: 149.4 on 19 and 564 DF,  p-value: < 2.2e-16
```

The model development results such as the significance code and P value both of them are important parameter to tell us about the model selected is going to work on our test data or not. The P value which is very less gives us assurance that the model selected is going to work very well on our test data.

From the linear regression model we get predicted values using the code

```
#Build regression model on train data
lm_model = lm(cnt ~., data = train)

#Summary of the model
summary(lm_model)

#Predict the values of test data by applying the model on test data
predictions_LR = predict(lm_model , test[,1:8])
```

but these predicted values of test cases and actual value of the test cases needs to be evaluated. In order to compare them we have used the following command.

```
#MODEL EVALUATION method
regr.eval(test[,'cnt'] , predictions_LR ,stats = c('mae','rmse','mape','mse'))
```

with this command in R we get the following results.

mae	rmse	mape	mse
0.067207463	0.089729757	0.202162924	0.008051429

To evaluate regression model we used the following criterion table:

STATISTIC	CRITERION
R-Squared	Higher the better *(> 0.70)*
Adj R-Squared	Higher the better
F-Statistic	Higher the better
Std. Error	Closer to zero the better
t-statistic	Should be greater 1.96 for p-value to be less than 0.05
AIC	Lower the better
BIC	Lower the better
Mallows cp	Should be close to the number of predictors in model
MAPE (Mean absolute percentage error)	Lower the better
MSE (Mean squared error)	Lower the better
Min_Max Accuracy => mean(min(actual, predicted)/max(actual, predicted))	Higher the better

From the above results with mean absolute error, root mean square error, mean absolute percentage error and mean square error. The important parameter is root mean square error which gives us the indication of how far our predicted values are away from the actual test values. The result shows that it is 0.089, the lesser it is the better is our model for predicting the values. In our case considering 0 as the perfect value for our prediction and 1 as the worst case our model is (100 – 8.9 = 91.9 %) accurately predicting the values.

Also from the table comparing other values like
MAPE which is lower the value better is our model we could say that from our value which is 0.2021 is lower and our model is good enough for prediction.

Similarly our MSE (Mean squared error value) is 0.00805 is a lower value and the lower the value better is our model for prediction.

Chapter 3

Conclusion

3.1 Model Evaluation

Now that we have used decision tree for regression and Linear regtression model, 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 both the model have the same RMSE(0.08950684 for decision tree and 0.089729757 for regression) value so we can say that both model are suitable for predicting the future test cases. But the problem statement objective is to know the seasonal and environmental settings on which our daily based count depends. The solution to this problem would be to interpret the summary of our regression model. The summary of regression model is obtained by following R code.

#Summary of the model

summary(lm_model)

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

Residuals:
     Min       1Q   Median       3Q      Max
-0.49955 -0.04609  0.01250  0.05739  0.28740

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  2.079e-01  2.324e-02   8.948  < 2e-16 ***
instant      5.858e-04  2.149e-05  27.252  < 2e-16 ***
season2      7.020e-02  2.530e-02   2.774  0.00571 **
season3      8.323e-02  2.894e-02   2.876  0.00418 **
season4      1.745e-01  2.394e-02   7.290 1.05e-12 ***
mnth2       -3.988e-03  1.962e-02  -0.203  0.83896
mnth3        1.110e-02  2.173e-02   0.511  0.60950
mnth4       -8.433e-03  3.473e-02  -0.243  0.80821
mnth5        2.066e-02  3.642e-02   0.567  0.57069
mnth6       -2.826e-02  3.758e-02  -0.752  0.45243
mnth7       -1.152e-01  4.115e-02  -2.800  0.00529 **
mnth8       -6.577e-02  3.986e-02  -1.650  0.09952 .
mnth9       -3.713e-02  3.560e-02  -1.043  0.29746
mnth10      -1.318e-01  3.241e-02  -4.066 5.47e-05 ***
mnth11      -2.108e-01  3.150e-02  -6.692 5.32e-11 ***
mnth12      -2.147e-01  2.533e-02  -8.475  < 2e-16 ***
atemp        3.631e-01  4.387e-02   8.276 9.22e-16 ***
hum         -2.326e-01  2.291e-02 -10.155  < 2e-16 ***
windspeed   -1.181e-01  2.050e-02  -5.761 1.37e-08 ***
casual       1.898e-01  2.239e-02   8.474  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.09129 on 564 degrees of freedom
Multiple R-squared:  0.8342,  Adjusted R-squared:  0.8286
F-statistic: 149.4 on 19 and 564 DF,  p-value: < 2.2e-16
```

The summary of the linear regression model has column such as Estimate, Standard Error , t value and Pr(>|t|).
Each column has its own interpretation. The estimate column tells you the intercept made by the dependent variable with our target variable . This value tells you how the unit value of target variable gets affected by the change in our intercept which is the unit value of independent variable. Before interpreting this estimate we need to check the significance of our variable .

From our summary arranging the variable in decesnding order of significance we get the following order for season variable.

```
season4     1.745e-01  2.394e-02   7.290 1.05e-12 ***
season2     7.020e-02  2.530e-02   2.774  0.00571 **
season3     8.323e-02  2.894e-02   2.876  0.00418 **
```

The fourth season carries a higher significance with our target variable and its estimate 1.745e-01 tells you that the total bike count increases in this season which is winter season. Similarly when we look at season 2 and 3 which are summer and fall the total bike users increases in this season.

For the month variable the order is

```
mnth12     -2.147e-01  2.533e-02  -8.475  < 2e-16 ***
mnth11     -2.108e-01  3.150e-02  -6.692 5.32e-11 ***
mnth10     -1.318e-01  3.241e-02  -4.066 5.47e-05 ***
mnth7      -1.152e-01  4.115e-02  -2.800  0.00529 **
```

The month 12 which is december and falls in the winter season tells us that altough the season shows significant rise in total bike users in winter season but during the month of december the total count decreases.

Similarly the months 10 and 11 which are october and november and falls in the Fall(autumn) season the total bike users decreases while in the remaining months the total bikecount increases. In summer season the total bike users increases but in the month of july which is month number 7 the total bike count decreases.

Loooking at the environmental settings the atmospheric temperature significantly affects the total bike count and it varies directly , so we can say that as atmospheric temperature increases the total bike count increses. Next variable which is our casual bike users shows that as the count of casual bike users increases ou total bike bike user count also increases.

Now the windspeed also significantly defines the total bike count but the negative sign indicates an inverse relationship which shows that if the windspeed is more our total bike users are less. Similar to that is our humidity factor which also carries inverse relation and tells that as humidity in atmosphere is high the total bike count decreases.

```
atemp       3.631e-01  4.387e-02   8.276 9.22e-16 ***
casual      1.898e-01  2.239e-02   8.474  < 2e-16 ***
windspeed  -1.181e-01  2.050e-02  -5.761 1.37e-08 ***
hum        -2.326e-01  2.291e-02 -10.155  < 2e-16 ***
```

3.2 Model Selection

We can see that linear regression model helps to answer the bike count predictions as well as the decision tree regression model. But when it comes to interpret the environmental and seasonal factor linear regression model makes it convinient to see the effect of season and environmental factor. So choosing linear regression model would be the best alternative.

Appendix A - R Code and Python Code

R code:

```
rm(list = ls(all = T))
getwd()

rm(list=ls())

#Set the directory
setwd("C:/Users/Asus/Documents/R programming")
getwd()

#Load libraries
#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)

##Read the data
data_bike = read.csv("day.csv", header = T, na.strings = c(" ", "", "NA"))

#########Converting data types ##############
str(data_bike) # Checking the required data types
class(data_bike)
data_bike$season = as.factor(data_bike$season)
data_bike$yr = as.factor(data_bike$yr)
data_bike$mnth = as.factor(data_bike$mnth)
data_bike$holiday = as.factor(data_bike$holiday)
data_bike$weekday = as.factor(data_bike$weekday)
data_bike$workingday = as.factor(data_bike$workingday)
data_bike$weathersit = as.factor(data_bike$weathersit)
data_bike$casual = as.numeric(data_bike$casual)
data_bike$registered = as.numeric(data_bike$registered)
data_bike$cnt = as.numeric(data_bike$cnt)
data_bike$windspeed = as.numeric(data_bike$windspeed)
##################Missing values analysis#############################
#Create a dataframe with missing percentage
missing_val = data.frame(apply(data_bike,2,function(x){sum(is.na(x))}))

#Convert row names into columns
missing_val$Columns = row.names(missing_val)
row.names(missing_val) = NULL

#Rename the variable name
names(missing_val)[1] =  "Missing_percentage"

#Calculate the percentage
missing_val$Missing_percentage = (missing_val$Missing_percentage/nrow(data_absent)) * 100

#Arrange in descending order
missing_val = missing_val[order(-missing_val$Missing_percentage),]

#Rearrange the column names
missing_val = missing_val[,c(2,1)]


missing_val
```

```
###################### No missing values found ###############

#######################################Outlier
Analysis#############################
# ## BoxPlots - Distribution and Outlier Check
numeric_index_bike = sapply(data_bike,is.numeric) #selecting only numeric

numeric_data_bike = data_bike[,numeric_index_bike]

cnames_bike = colnames(numeric_data_bike)

for (i in 1:length(cnames_bike))
{
  assign(paste0("gn",i), ggplot(aes_string(y = (cnames_bike[i])), data = subset(data_bike))+
           stat_boxplot(geom = "errorbar", width = 0.5) +
           geom_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,
                        outlier.size=1, notch=FALSE) +
           theme(legend.position="bottom")+
           labs(y=cnames_bike[i])+
           ggtitle(paste("Box plot of Bike rent for variable ",cnames_bike[i])))
}


## Plotting plots together
gridExtra::grid.arrange(gn1,ncol=1)
gridExtra::grid.arrange(gn2,ncol=1)
gridExtra::grid.arrange(gn3,ncol=1)
gridExtra::grid.arrange(gn4,ncol=1)
gridExtra::grid.arrange(gn5,ncol=1)
gridExtra::grid.arrange(gn6,ncol=1)
gridExtra::grid.arrange(gn7,ncol=1)
gridExtra::grid.arrange(gn8,ncol=1)


###############Creating box plot for each variable #########
boxplot(data_bike$instant)
boxplot(data_bike$temp)
boxplot(data_bike$atemp)
boxplot(data_bike$hum)
boxplot(data_bike$windspeed)
boxplot(data_bike$casual)
boxplot(data_bike$registered)
boxplot(data_bike$cnt)


############### loop to remove outlier and impute using Knn#############
for(i in cnames_bike)
{val = data_bike[,i][data_bike[,i] %in% boxplot.stats(data_bike[,i])$out]
#print(length(val))
data_bike[,i][data_bike[,i] %in% val] = NA
}

data_bike = knnImputation(data_bike, k = 5)

###################################Feature
Selection#################################################
## Correlation Plot
corrgram(data_bike[,numeric_index_bike], order = F,
         upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")

####Drop temp and registered user ######

## Chi-squared Test of Independence
factor_index = sapply(data_bike,is.factor)
factor_data = data_bike[,factor_index]

factor_data_2 = factor_data[, c(1, 8, 3, 4, 5, 6, 7, 2)]

for (i in 1:7)
```

```
{
  print(names(factor_data_2)[i])
  print(chisq.test(table(factor_data_2$season,factor_data_2[,i])))
}

############Drop variable "dteday", "weathersit","yr", "holiday", "weekday", "working day"#################

## Dimension Reduction
data_bike = subset(data_bike,
                   select = -c(dteday, weathersit, yr, holiday, weekday, workingday, temp, registered ))

str(data_bike)

#Normality check
qqnorm(data_bike$atemp)
hist(data_bike$atemp)

qqnorm(data_bike$hum)
hist(data_bike$hum)

qqnorm(data_bike$windspeed)
hist(data_bike$windspeed)

qqnorm(data_bike$casual)
hist(data_bike$casual) #### data is left skewed

qqnorm(data_bike$cnt)
hist(data_bike$cnt)


numeric_index_norm = sapply(data_bike,is.numeric) #selecting only numeric

numeric_data_norm = data_bike[,numeric_index_norm]

cnames_absent_norm = colnames(numeric_data_norm)

cnames_norm = c("atemp","hum","windspeed","casual","cnt")

for(i in cnames_norm){
  print(i)
  data_bike[,i] = (data_bike[,i] - min(data_bike[,i]))/
    (max(data_bike[,i] - min(data_bike[,i])))
}

df = data_bike

#####################################Train and test data ##################
###############Clean the environment
library(DataCombine)
rmExcept("df")

#Divide data into train and test using stratified sampling method
set.seed(1234)
train_index = sample(1:nrow(df), 0.8*nrow(df))
train = df[ train_index,]
test  = df[-train_index,]
```

```
#Load Libraries
library(rpart)
library(MASS)

############################Linear regression
#check Multicollinearity

library(usdm)
vif(df[, -10])
numeric_index = sapply(df,is.numeric) #selecting only numeric

numeric_data = df[,numeric_index]

cnames_df = colnames(numeric_data)
cnames_df = data.frame(cnames_df)

vifcor(numeric_data[, -6], th = 0.9)

#Build regression model on train data
lm_model = lm(cnt ~., data = train)

#Summary of the model
summary(lm_model)

#Predict the values of test data by applying the model on test data
predictions_LR = predict(lm_model , test[,1:8])

#MODEL EVALUATION method
regr.eval(test[,'cnt'] , predictions_LR ,stats = c('mae','rmse','mape','mse'))


################################### Decision Tree Regression ##################

#########rpart for regression ####################
fit_dt = rpart(cnt~ ., data = df, method = "anova")

#Predict for new test cases
#Predict for new test cases
predictions_DT = predict(fit_dt, test[,-8])

RMSE(predictions_DT, test$cnt)
```

Python code:

```
# In[ ]:


#Load libraries
import os
import pandas as pd
import numpy as np
from fancyimpute import KNN
import matplotlib.pyplot as plt
from scipy.stats import chi2_contingency
import seaborn as sns


# In[ ]:


#set working directory
os.chdir("F:\Others\Project Bike Rental")


# In[ ]:


#Load data
data_bike = pd.read_csv("day.csv")


# In[ ]:


data_bike.dtypes


# In[ ]:


data_bike[dteday] = int(data_bike[dteday])


# In[ ]:


#Create dataframe with missing percentage
missing_val = pd.DataFrame(data_bike.isnull().sum())


# In[ ]:
```

```
#Reset Index
missing_val = missing_val.reset_index()


# In[ ]:


#Rename variables
missing_val = missing_val.rename(columns={'index':'Variables', 0 : 'Missing_Percentage'})


# In[ ]:


#calculate percentage
missing_val["Missing_Percentage"] = (missing_val['Missing_Percentage']/len(data_bike))*100


# In[ ]:


#desecnding order
missing_val = missing_val.sort_values('Missing_Percentage', ascending = False).reset_index(drop = True)


# In[ ]:


missing_val


# In[ ]:


#plot boxplot to visualize outliers
get_ipython().run_line_magic('matplotlib', 'inline')

plt.boxplot(data_bike['casual'])


# In[ ]:


data_bike.head(10)


# In[ ]:
```

```
#Save numeric variables
cnames = ["instant","temp", "atemp","hum","windspeed","casual","registered","cnt"]


# In[ ]:


#Detect and delete outliers from the data
for i in cnames:
    print (i)
    q75, q25 = np.percentile(data_bike.loc[:,i], [75 ,25])
    iqr = q75 - q25

    min = q25 - (iqr*1.5) #Innerfence
    max = q75 + (iqr*1.5) #Upperfence
    print(min)
    print(max)
    data_bike = data_bike.drop(data_bike[data_bike.loc[:,i] < min].index)
    data_bike = data_bike.drop(data_bike[data_bike.loc[:,i] > max].index)


# In[ ]:


##Correlation plot
#Corelation plot
df_corr = data_bike.loc[:,cnames]


# In[ ]:


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

#Set correlation matrix
corr = df_corr.corr()

#Plot using seaborn library
sns.heatmap(corr, mask = np.zeros_like(corr ,dtype = np.bool), cmap=sns.diverging_palette(220,
10, as_cmap=True),
            square=True, ax=ax)


# In[ ]:


##Chi square test
```

```
#Save categorical variable
cat_names = ["dteday","yr","mnyh","holiday ","weekday","workingday","weathersit"]


# In[ ]:


cat_names


# In[ ]:


#loop for chi square values
for i in cat_names:
    print(i)
    chi2, p, dof, ex = chi2_contingency(pd.crosstab(data_bike['season'],cat_names[i]))
    print(p)


# In[ ]:


## Drop the variable
#Drop the variables from data
data_bike = data_bike.drop(['dteday', 'weathersit', 'yr', 'holiday', 'weekday', 'workingday', 'temp',
'registered'], axis=1)


# In[ ]:


#Normality check
get_ipython().run_line_magic('matplotlib', 'inline')
plt.hist(data_bike['cnt'], bins = 'auto')

#Normalisation
for i in cnames:
    print(i)
    data_bike[i] = (data_bike[i] - min(data_bike[i]))/(max(data_bike[i]) -data_bike[i]))


# In[ ]:


data_bike.head(10)


# In[ ]:
```

```
#Import libraries for decision tree
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.cross_validation import train_test_split


# In[ ]:


##############decision tree for regression############

#Divide data into train and test
train , test = train_test_split(data_bike , test_size = 0.2)


# In[ ]:


#Load libraries
import os
import pandas as pd
import numpy as np
from sklearn.cross_validation import train_test_split
from sklearn.tree import DecisionTreeRegressor


# In[ ]:


#decision tree for regresion
fit_DT = DecisionTreeRegressor(max_depth = 2).fit(train.iloc[:,0:7], train.iloc[:,7])


# In[ ]:


fit_DT


# In[ ]:


#Apply model on test data
predictions_DT = fit_DT.predict(test.iloc[:,0:7])


# In[ ]:
```

```
#Calculate MAPE
def MAPE(y_true, y_pred):
    mape = np.mean(np.abs((y_true - y_pred)/y_true))*100
    return mape


# In[ ]:


MAPE(test.iloc[:,7], predictions_DT)


###################### Linear Regression ##################

# In[ ]:


#Import libraries for LR
import statsmodels.api as sm

#Train the model using the training sets
model = sm.OLS(train.iloc[:,7],
           train.iloc[:,0:7]).fit()


# In[ ]:


#Print out the summary
model.summary()


# In[ ]:


#make the predictions by the model
predictions_LR = model.predict(test.iloc[:,0:7])


# In[ ]:


#Calculate MAPE
MAPE(test.iloc[:,7], predictions_LR)
```

References

1. https://stackoverflow.com/users/10254950/sarang-kapse

2. https://towardsdatascience.com/the-random-forest-algorithm-d457d499ffcd

3. https://www.rdocumentation.org/packages/Metrics/versions/0.1.4/topics/rmse

4. https://www.youtube.com/watch?v=tSPg-JDAF4M

5. http://www.sthda.com/english/articles/40-regression-analysis/167-simple-linear-regression-in-r/