**PRIDICTING BIKE RENTAL COUNT**

ASHISH KUMAR SAW

18TH AUGUST 2018

**Contents**

**Chapter 1**

**Introduction**

* 1. **Problem Statement**

The objective of this Case is to Predication of bike rental count on daily based on the

environmental and seasonal settings. Bike Renta is a program which is running around the world to get membership and renting a bike. People used to rent a bike from a place for some time and return to the same place or other place of the bike renting branch. There are two types of people rent the bike, one is the registered user who rent the bike for almost daily purpose like servicemens, students, employees on the other hand there are casuals peoples who rent the bike on need, like people in the vacation rent bikes. This is the problem in which we have to predict the count of people who rent the bike on summing registered and casual problem. It is a regression problem in which the target variable is continous one.

* 1. **Data**

Here, we have to predict the count of peoples who are renting a bike on the daily usage on the basis of temperature, humidity, weekday, holiday etc. Here, we build a regression model which will sum the registered users and the casuals users on the daily renter of the bike.

**1.3 Sample Data**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| instant | dteday | season | yr | mnth | holiday | weekday | workingday | weathersit | temp | atemp | hum | windspeed | casual | registered | cnt |
| 1 | 2011-01-01 | 1 | 0 | 1 | 0 | 6 | 0 | 2 | 0.344167 | 0.363625 | 0.805833 | 0.160446 | 331 | 654 | 985 |
| 2 | 2011-01-02 | 1 | 0 | 1 | 0 | 0 | 0 | 2 | 0.363478 | 0.353739 | 0.696087 | 0.248539 | 131 | 670 | 801 |
| 3 | 2011-01-03 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0.196364 | 0.189405 | 0.437273 | 0.248309 | 120 | 1229 | 1349 |
| 4 | 2011-01-04 | 1 | 0 | 1 | 0 | 2 | 1 | 1 | 0.2 | 0.212122 | 0.590435 | 0.160296 | 108 | 1454 | 1562 |
| 5 | 2011-01-05 | 1 | 0 | 1 | 0 | 3 | 1 | 1 | 0.226957 | 0.22927 | 0.436957 | 0.1869 | 82 | 1518 | 1600 |
| 6 | 2011-01-06 | 1 | 0 | 1 | 0 | 4 | 1 | 1 | 0.204348 | 0.233209 | 0.518261 | 0.089565 | 88 | 1518 | 1606 |
| 7 | 2011-01-07 | 1 | 0 | 1 | 0 | 5 | 1 | 2 | 0.196522 | 0.208839 | 0.498696 | 0.168726 | 148 | 1362 | 1510 |
| 8 | 2011-01-08 | 1 | 0 | 1 | 0 | 6 | 0 | 2 | 0.165 | 0.162254 | 0.535833 | 0.266804 | 68 | 891 | 959 |
| 9 | 2011-01-09 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0.138333 | 0.116175 | 0.434167 | 0.36195 | 54 | 768 | 822 |
| 10 | 2011-1-10 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0.150833 | 0.150888 | 0.482917 | 0.223267 | 41 | 1280 | 1321 |

There are 731 observations and 16 variables in which 15 are independent variables and 1 dependent variable.

* 1. **Predictor Variables**

1. Instant
2. Dateday
3. Season
4. Yr
5. Mnth
6. Holiday
7. Weekday
8. Workingday
9. Weathersit
10. Temp
11. Atemp
12. Hum
13. Windspeed
14. Casual
15. Registered
    1. **Predicted variables**
16. Cnt

> dim(data)

[1] 731 16

**1.6 Understanding the Data**

The datasets shows hourly rental data for two years(2011-2012). Here, we have to predict the total count of the bikes rented during each hour.

In the data, they have seperately given bike demand by registered, casuals users and sum of both is given as count.

The data has 16 variables in which 13 are independent variables and 3 are dependent variables.

**Chapter 2**

**Methodology**

**2.1 Pre-Processing**

Data Pre-Processing means the procesing the data before modelling the data. It in includes data exploration, data manipulation, data cleaning and visualizing the data. Pre-Processing is done to make the data in a well structured way because the data we get from the client are all the messy data. So, to convert this messy data in a well structured way data Pre-Processing is done. This phenomenon is called Exploratory Data Analysis. Is includes many techniques to convert the data in a structured format like, Missing Value Analysis, Outlier Analysis, Feature selection and Feature scaling.

Before Pre-Processing we have to convert the varaibles into other variables if required.

Since, season has only four values so better to convert it to categorical variable from numeric variable. Similarly, yr, mnth, holiday, weekday, workingday and weathersit have only some repeted values so better to convert them to the categorical variable from numerical variable.

**2.1.1 Missing Value Analysis**

Missing value is defined as the some data points are missed from the observations. Missing value can effect the observation negatively which means the target value gets biased to other observations. The target value may not be predicted the appropriate value. So, first of all we have to check for the missing value in the data, if present then we have to solve the missing value problem. It can be solved by the statistical methods such as Mean, Median and Mode or by KNN imputations. Here, in the day.csv data, it doesn’t contain any missing value as we get the result after compiling the data in R and Python code. The result we get is FALSE which means the data doesn’t conatin any missing value. We can also count the number of missing value for each variable. Since for each variable the count of missing value is 0. So, no missing value is present in the data.

> table(is.na(data))

FALSE

11696

> missing\_val = data.frame(apply(data,2,function(x){sum(is.na(x))})) # counts the missinig value for each variable

> missing\_val

apply.data..2..function.x...

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

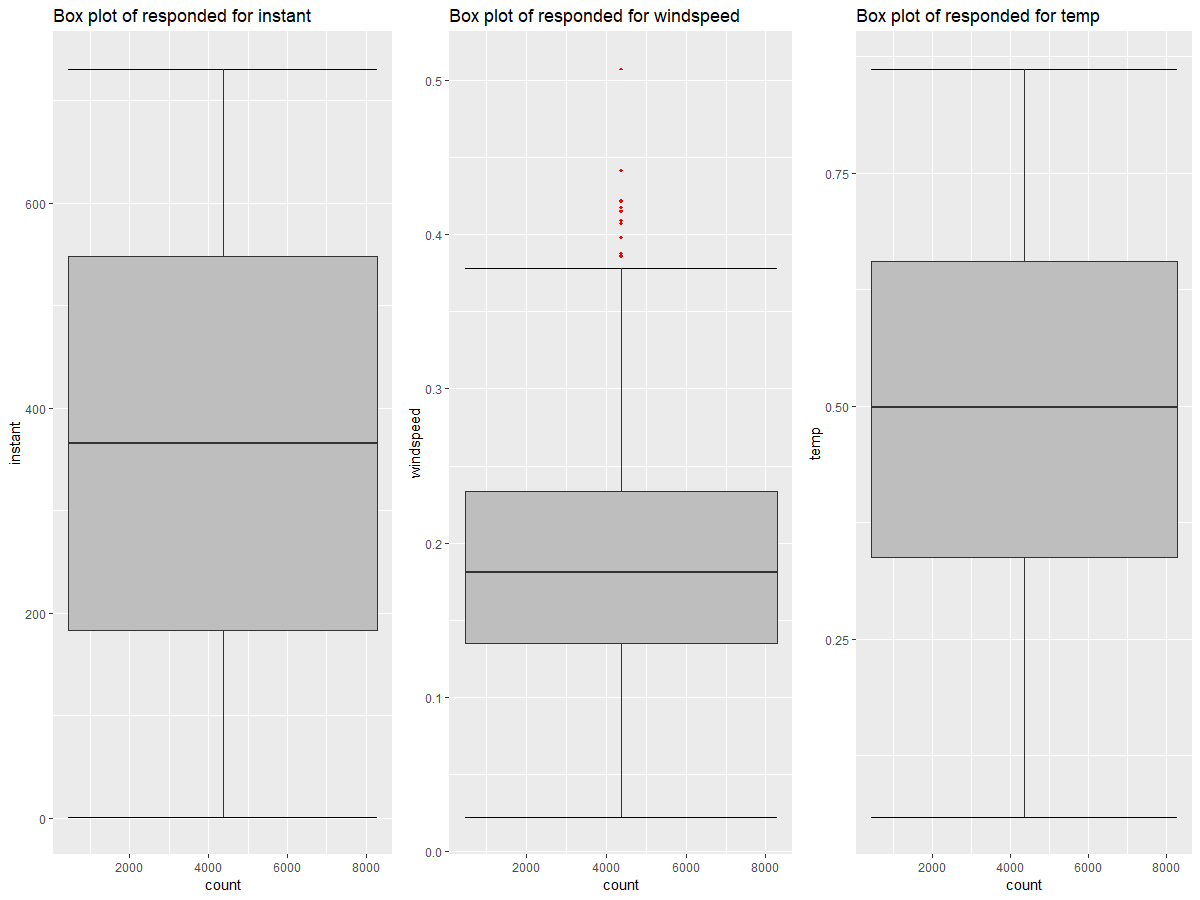
registered 0

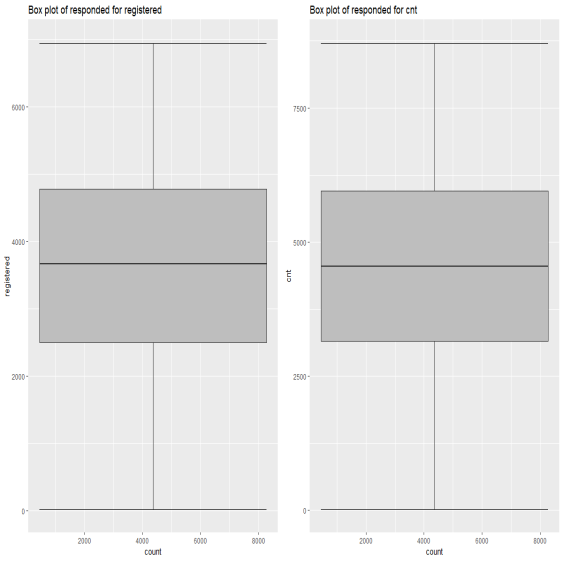
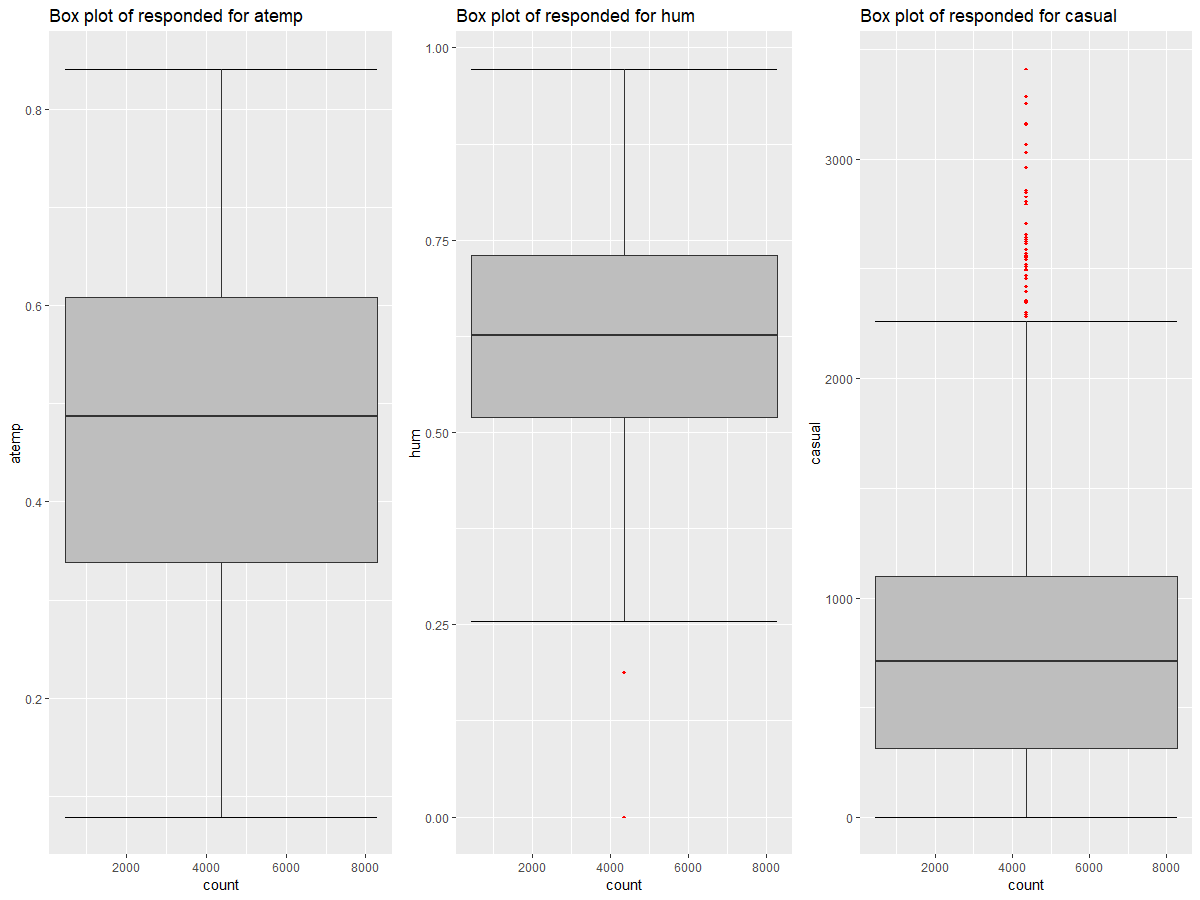
cnt 0

As we see that for all the variables the missing value is 0, which indicates that the missing value is not present in the data give.

**2.1.2 Outlier Analysis**

Outliers are the inconsistent data points within a variables. Outliers are the counts or percentage of inconsistent variables which are present in the variables. Outliers results in low quality of data and low quality of measurement. The statistical measurements for the variables may not calculate the correct result as a rsullt the predictions may get wrong or may get some error. Outliers are seen only for continous variables. It can be detected by plotting the Box plot to see the outliers visually or Grubb’s test for outliers. After detecting the outliers we can either delete the observations or replace the data points by NA.





Fig(2.1)

In the figure we see that there are several outliers present in the variable windspeed, hum and casual. This means that the observations are inconsistent with rest of the datasets. The outliers generate may be because the data might be inappropriately scaled or the errors are made during the data entry, etc(there may be also some other reasons also), so better to remove these outliers before modelling the data.

To detect the outlier we use Box-plot method and to solve this problem we replace the outliers present in the data with “NA” value and then perporm statistical technique to solve the problem.

> # #create NA on "outliers”

> for(i in cnames){

+ val = data[,i][data[,i] %in% boxplot.stats(data[,i])$out]

+ print(length(val))

+ data[,i][data[,i] %in% val] = NA

+ }

[1] 0

[1] 0

[1] 0

[1] 2

[1] 13

[1] 44

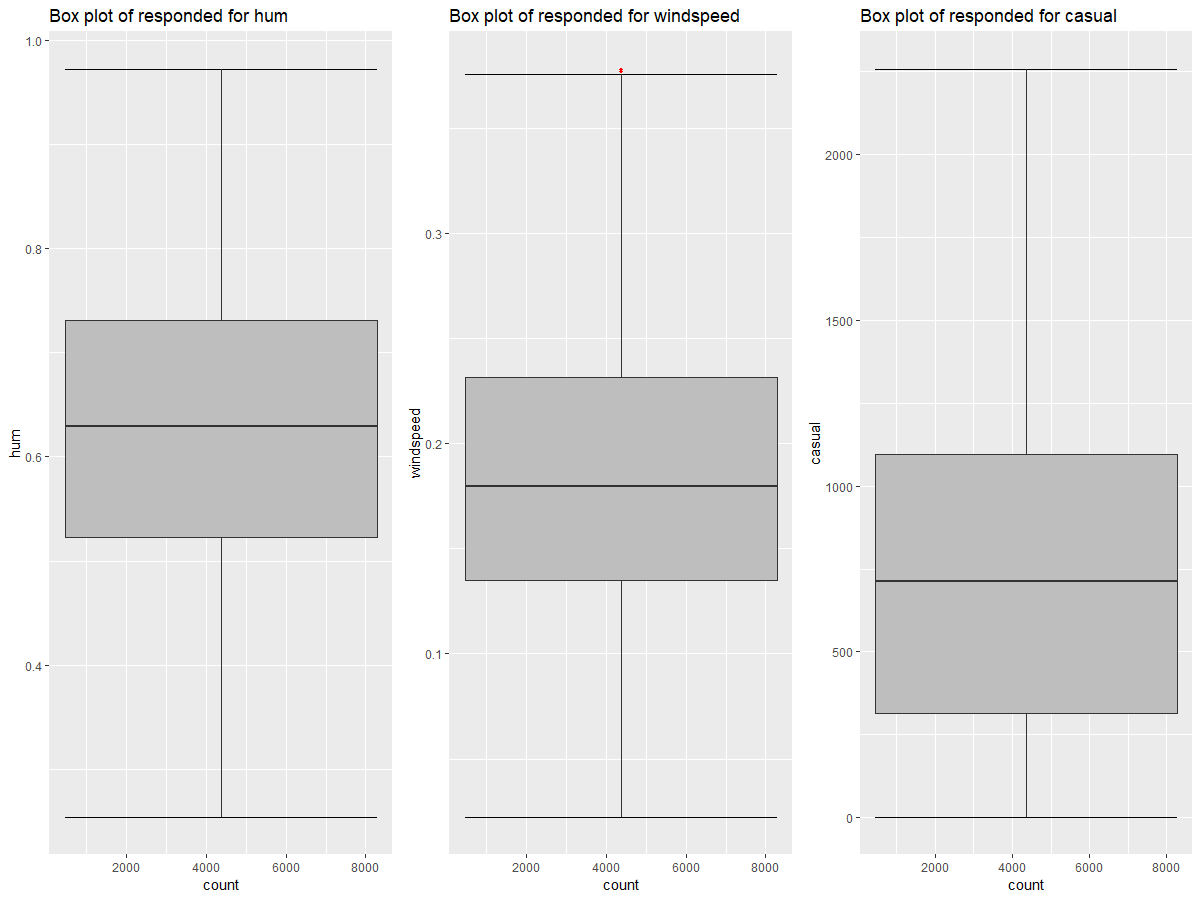
[1] 0

[1] 0

Here, 2 outliers are present in the hum, 13 in windspeed and 44 in casuals.

After analysing the different statistiacl techniques we see that the knnImputation is best technique to solve the outliers in the data.

The below diagram is measured after removing the outliers.

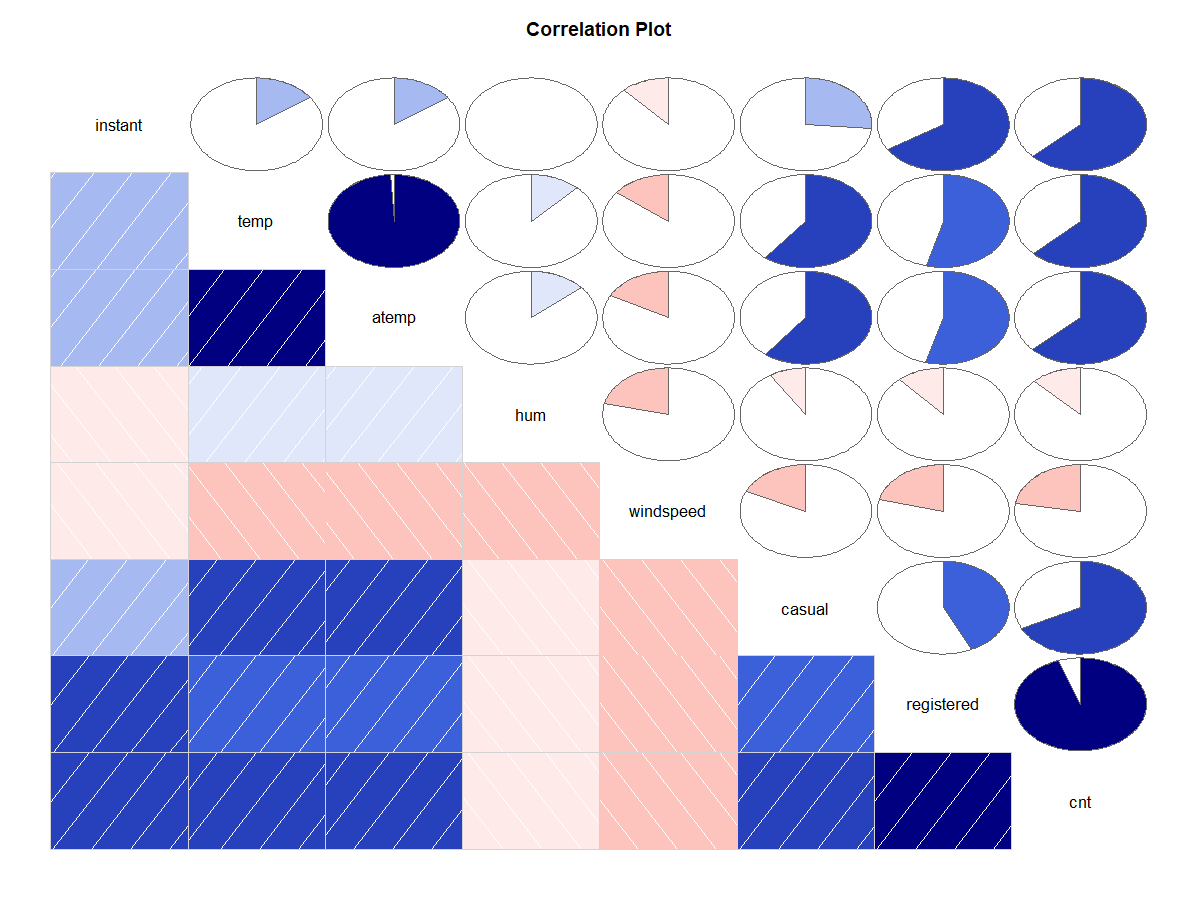


Fig(2.2)

**2.1.3 Feature Selection**

After solving the outliers problems, our data is now consistently distributed. Now we use the data to see the releationship between the viariables.Feature selection is the method of selecting a subset of relevant features(variables,predictors) for use in model construction. It reduces the dimensions of the data. There are different variables in the data which are not playing the useful role to predict the target variable on the other hand there are some variables which strongly play the role to pedict the target variable. There are also some independent variable which are highly correlated with each other. So, these all the features are taken into consideration to and the variables which are needed are only taken into consideration to predict the target variable.

For the numerical variable we go for correlation technique to see the correlation between the variables. This is done by plotting the correlation plot. The extreme blue color shows the variables are strongly positively correleated and the extreme red color shows the variables are strongly negatively correlated with each other.



In the given data, there are 8 continous variable as instant, temp, atemp, hum, windspeed, casual,registered and cnt. The above diagram shows the dependency between the different variables. Let us considered each variable one by one:-

1. Instant:- this variable is the index of the data. It shows what is the index of first and last observation present in the data. So, we can ignore this variable in the feature selection.
2. Temp:- it is normalized temperature in celsius. The max\_temp=39 and the , min\_temp=-8. The normalized value is calculated by the formula,

Value= (t-t\_min)/(t\_max-t\_min)

The temp shows a high dependency with the target variable cnt. So, temp variable is included in the feature selection to predict the target variable.

1. Atemp:- it is also the noramlized temperature in celsius. The max\_temp=50 and the min\_temp= -16. The noramlized value is calculated by the formula,

Value=(t-t\_min)/(t\_max-t\_min)

The atemp also shows a high dependency with the target variable cnt. But atemp and temp are almost strongly correlated with each other. So, while feature selection we take only one variable for prediction. So here we take only temp variable for prediction and ignore the atemp.

1. Hum:- it is also the normalized value. It is calculated by dividing the value with the max humidity which is 100. In the diagram we see that the hum variable is very less correleated with the target variable. So better to ignore this variable while feature selection.
2. Windpeed:- it is also the normalized value. It is calculated by dividing the value with the max value which is 67. The correlation between the hum and cnt is average. So this variable is included in the feature selection.
3. Casual:- it is the user which rent the bike casually. It highly correlated with the target variable, as it adds to the target variable. The casual variable is count on the daily survey of the data.
4. Registered:- it is the daily user wo rent the bikes. It is stronly correlated with each other. As the number of registered users increases by day. It is count of on the basis of daily uses. It aads to the target variable.

For the categorical variable we use the chi-square test for to see the relation between the categorical variables. The relation is based on the p-value. If the p-value is less than 0.05 the we reject the null hypothesis and say that the variables are independent else the variables are dependent.

> ## Chi-squared Test of Independence

> factor\_index = sapply(data,is.factor)

> factor\_data = data[,factor\_index]

>

> for (i in 1:8)

+ {

+ print(names(factor\_data)[i])

+ print(chisq.test(table(factor\_data[,i])))

+ }

[1] "dteday"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0, df = 730, p-value = 1

[1] "season"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0.29959, df = 3, p-value = 0.9601

[1] "yr"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0.001368, df = 1, p-value = 0.9705

[1] "mnth"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0.44186, df = 11, p-value = 1

[1] "holiday"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 649.41, df = 1, p-value < 2.2e-16

[1] "weekday"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 0.016416, df = 6, p-value = 1

[1] "workingday"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 98.989, df = 1, p-value < 2.2e-16

[1] "weathersit"

Chi-squared test for given probabilities

data: table(factor\_data[, i])

X-squared = 400.95, df = 2, p-value < 2.2e-16

From the output of the above code we see that the the variables dteday,season,yr,mnth,weekday have the p-value >0.05, which shows that these variables have no role in predicting the target variable. Whereas the variables weathersit,workingday,holiday have p-value <0.05, which shows that these variables play main role to predict the target variables. The variable holiday, workingday and the weathersit strongy correlated with the target variable. The users rent more bikes on the holidays, workingdays and weathersit.

So, after extracting the features form both the continous and categorical variable we see that the variables temp,windspeed, casual, registered, weathersit, workingday and holiday are the only variables which strongly helps in predicting the target variable.

> dim(data)

[1] 731 9

So, after the exploratory data analysis, the variables present in the data is 9. The structure of data is given as,

> str(data)

'data.frame': 731 obs. of 9 variables:

$ holiday : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ workingday: Factor w/ 2 levels "0","1": 1 1 2 2 2 2 2 1 1 2 ...

$ weathersit: Factor w/ 3 levels "1","2","3": 2 2 1 1 1 1 2 2 1 1 ...

$ temp : num 0.344 0.363 0.196 0.2 0.227 ...

$ 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 : num 331 131 120 108 82 88 148 68 54 41 ...

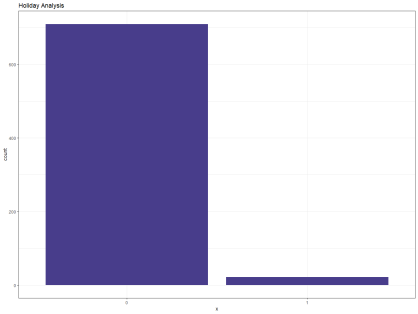
$ registered: int 654 670 1229 1454 1518 1518 1362 891 768 1280 ...

$ cnt : int 985 801 1349 1562 1600 1606 1510 959 822 1321 ...

> summary(data$holiday)

0 1

710 21 From the summary of the holiday variable and the bar

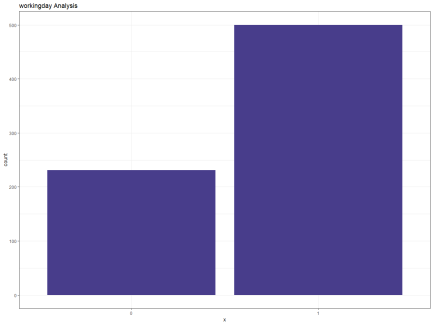
graph of the holiday, unique values of holiday are non-uniformly distributed. In this, the day without holiday has a count of 710 and the day with the holiday is 21 which means the target variable is highly dependent on the non holiday day.

> summary(data$workingday)

0 1

231 500

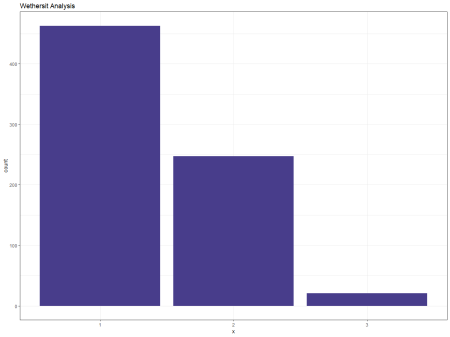
Form the summary of the workingday variable and the bar

graph of the workingday, unique values of workingday are non-uniformly distributed. In this, the working day has a count of 500 and non working day has a count of 231 which means that the the is rented more on working day than the non-working day.

> summary(data$weathersit)

1 2 3

463 247 21

From the summary of the weathersit variable and the bar graph of the weathersit, unique values of the weathersit is non-uniformly distributed. In this, there are 3 types of weathersit present.

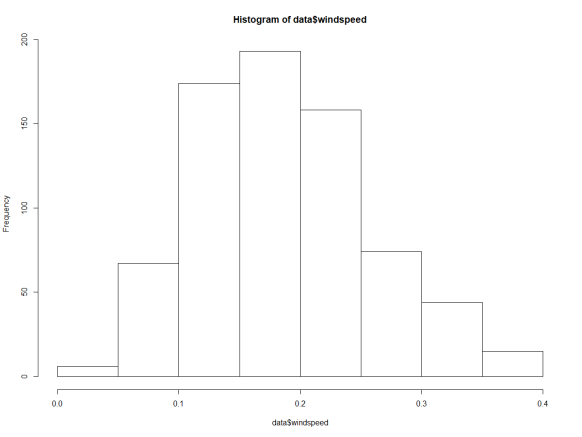
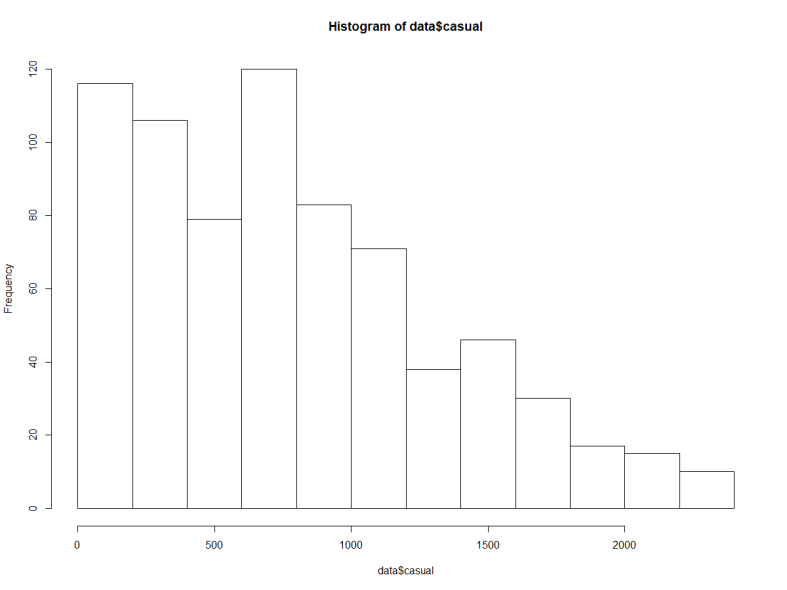
1: Clear, Few clouds, Partly cloudy, Partly cloudy

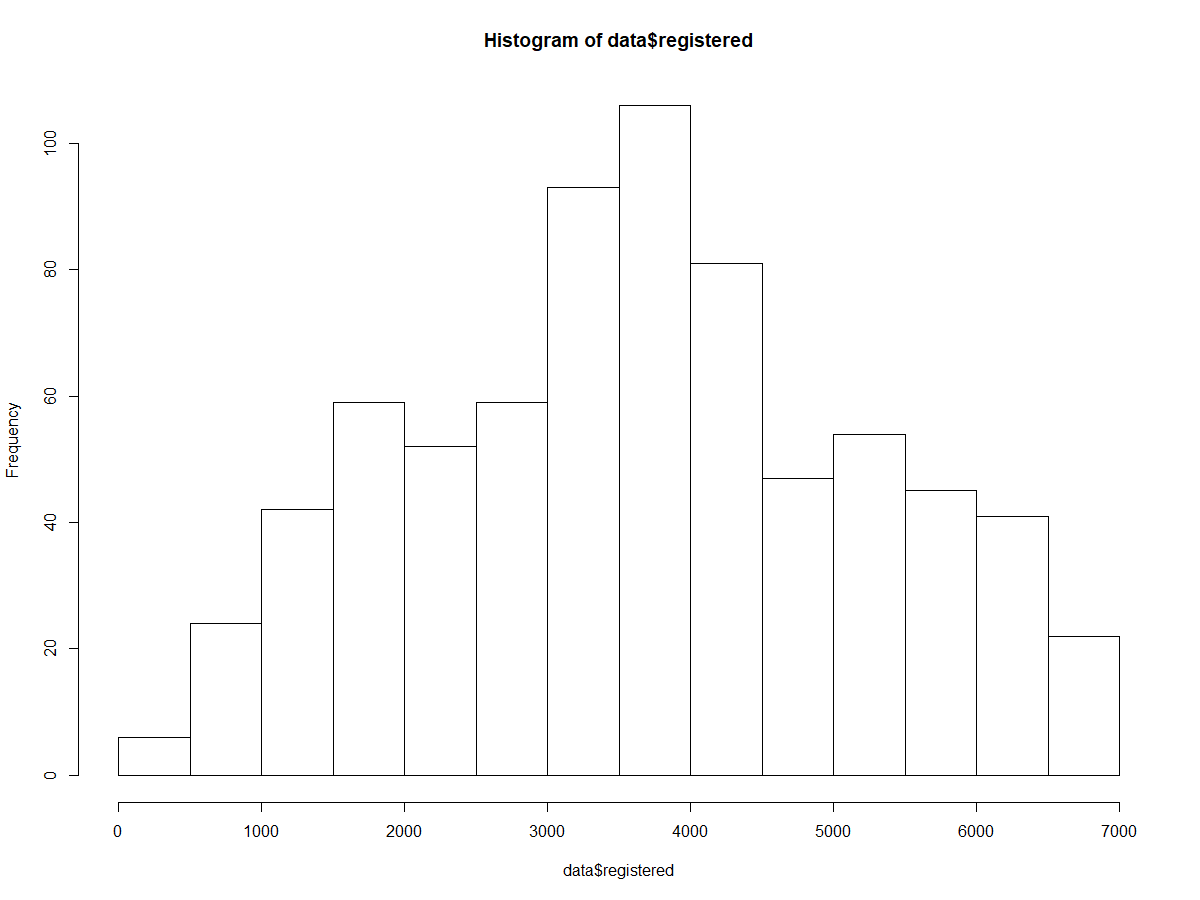
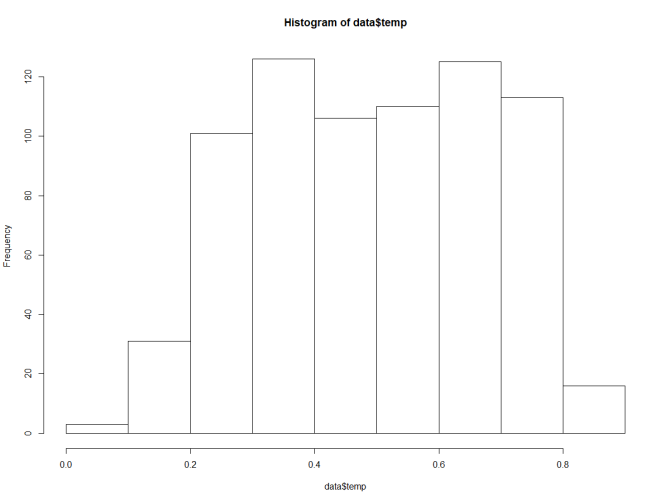
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist

3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered

Clouds.

The bike rented mostly on the clear weather and rented list on the light rain, thunderstrom.

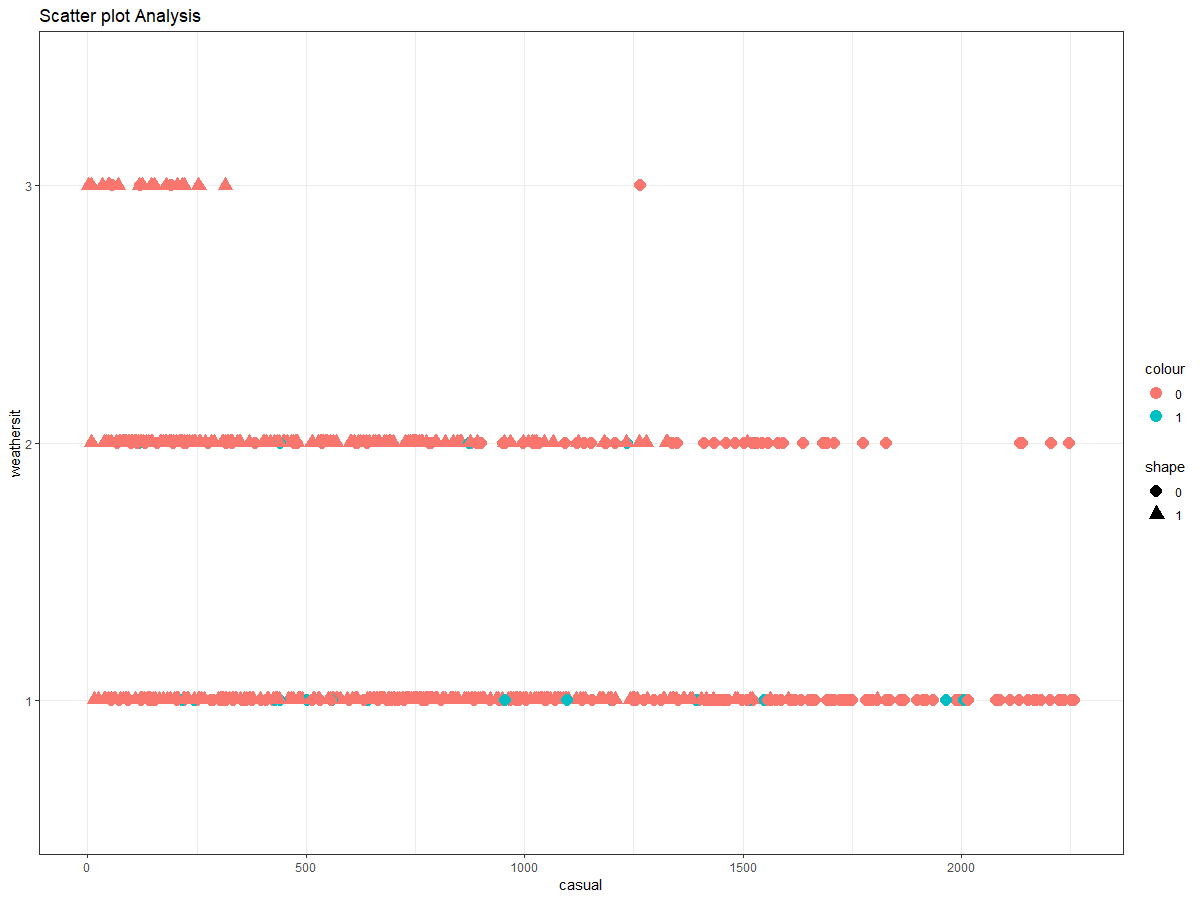
The distribution of continous variable is temp, windspeed, casual and registered is shown in the figure.

> #Multivariate #Scatter Plot

> ggplot(data, aes\_string(x = data$casual, y = data$weathersit)) +

+ geom\_point(aes\_string(colour = data$holiday, shape = data$workingday),size = 4) +

+ theme\_bw()+ ylab("weathersit") + xlab("casual") + ggtitle("Scatter plot Analysis")



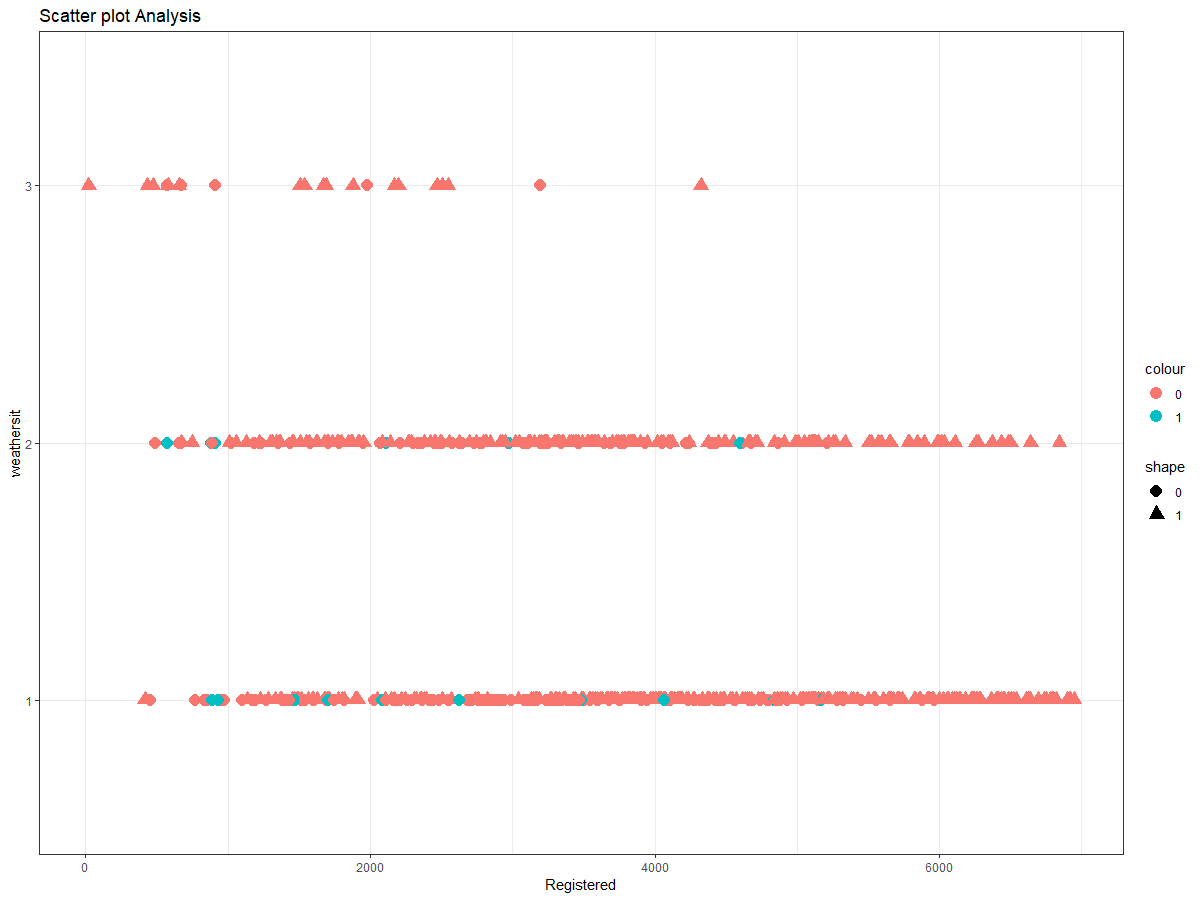
The scatter plot shows the distribution of weathersit on the casual variable. The orange color shows the points without the holiday and the green color shows the points in the holiday. The oval shape shows the points in the workingday and triangle points shows the points in the non-working day. In the scatter plot we see that the the casual variable have more points in the non-holidays and in the working days. This means that the casual users rent more bikes on the non-holidays and the workingdays. For the casual users the bike is rented more on the clean weather and less on the little rainy and thunderstroms.

> #Multivariate #Scatter Plot

> ggplot(data, aes\_string(x = data$registered, y = data$weathersit)) +

+ geom\_point(aes\_string(colour = data$holiday, shape = data$workingday),size = 4) +

+ theme\_bw()+ ylab("weathersit") + xlab("Registered") + ggtitle("Scatter plot Analysis")



The scatter plot shows the distribution of weathersit on the registered variable. The orange color shows the points without the holiday and the green color shows the points in the holiday. The oval shape shows the points in the workingday and triangle points shows the points in the non-working day. In the scatter plot we see that the the registered variable have more points in the non-holidays and in the working days. This means that the casual users rent more bikes on the non-holidays and the workingdays. For the registered users the bike is rented more on the clean weather and less on the little rainy and thunderstroms.

**3.2 Model development**

After exploratory data analysis, our main aim is to design a model which can predict the target variable.

**3.2.1 Model Development**

The target variable given in data is the continous one. So, there are different regression model present which helps to desing a model for the data.

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

Before, making the model we first divide the model in train and test case. The sampling is done by the stratified sampling method.

> set.seed(123)

> train.index = createDataPartition(data$cnt, p = .80, list = FALSE)

> train = data[ train.index,]

> test = data[-train.index,]

The train data contains 80% of data and the test case contains 20% of data.

> dim(train)

[1] 587 8

> dim(test)

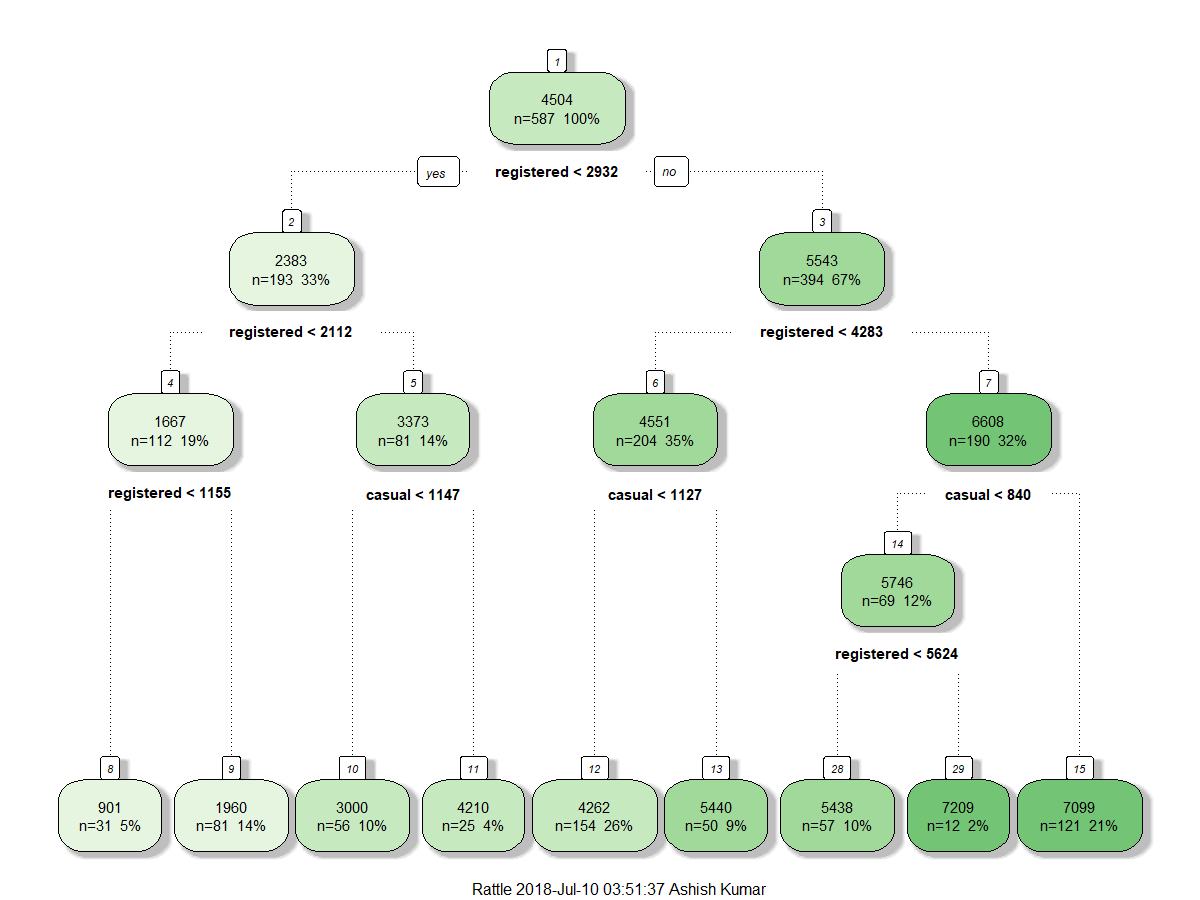
[1] 144 8

Let us start with the Decision tree model.

1. Decision tree model:- A Decision tree is a Predictive model based on a branching series of boolean test. In this the data are diveded into differnet branches based on the categories present in the data.

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

> fancyRpartPlot(fit)



The diagram shows the decisions tree model on the data.

1. Random Forest:- Random forest is an ensemble that consists of many decision tree. It is calculated on the basis on GINI Index.

> RF\_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 1000)

> RF\_model

Call:

randomForest(formula = cnt ~ ., data = train, importance = TRUE, ntree = 1000)

Type of random forest: regression

Number of trees: 1000

No. of variables tried at each split: 2

Mean of squared residuals: 107933

% Var explained: 97.12

1. Multiple Regression:- Multiple Regression describes the relation between different variable and helps in making a model for continous variable.

> vifcor(data[,4:7], th = 0.9)

No variable from the 4 input variables has collinearity problem.

The linear correlation coefficients ranges between:

min correlation ( windspeed ~ temp ): -0.1470409

max correlation ( casual ~ temp ): 0.6043266

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

Variables VIF

1. temp 1.857541

2 windspeed 1.056378

3 casual 1.627423

4 registered 1.475092

Here, temp is strongly correlated value with the target value

> #run regression model

> lm\_model = lm(cnt ~., data = train)

> #Summary of the model

> summary(lm\_model)

Call:

lm(formula = cnt ~ ., data = train)

Residuals:

Min 1Q Median 3Q Max

-287.37 -106.14 -23.14 32.98 1859.56

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 4.460e+01 4.931e+01 0.904 0.366

holiday1 -4.624e+01 6.217e+01 -0.744 0.457

workingday1 -2.296e+02 3.559e+01 -6.451 2.35e-10 \*\*\*

weathersit2 2.368e+01 2.196e+01 1.078 0.281

weathersit3 5.413e+01 6.722e+01 0.805 0.421

temp 1.199e+01 8.080e+01 0.148 0.882

windspeed 3.064e+01 1.439e+02 0.213 0.831

casual 1.015e+00 3.579e-02 28.356 < 2e-16 \*\*\*

registered 1.039e+00 9.428e-03 110.199 < 2e-16 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 241.5 on 578 degrees of freedom

Multiple R-squared: 0.9847, Adjusted R-squared: 0.9845

F-statistic: 4644 on 8 and 578 DF, p-value: < 2.2e-16

As we see that the Adjusted R-squared value is 0.9845 which means that the 98.45% of data is explained by the Multiple Regression, which is best one.

**Chapter 3**

**Conclusion**

**3.1 Model Evaluation**

Model evaluation is defined as how the predicted value are correct with the actual value.

For this we have defferent techniques like:-

1. MAE or MAD(Mean Absolute Error/Deviation)

It Averages the Absolute Error



1. MAPE(Mean Absolute Percentage Error)

It measures accuracy as a percentage of error.



1. RMSE/RMSD(Root Mean Square Error/Deviation)

It squares the error, find their average and take the square root.



Here, for the evaluation purpose we take the MAPE techinque to check the accuracy of the model.

#calculate MAPE

MAPE = function(y, yhat){

mean(abs((y - yhat)/y))\*100

}

1. Decission Tree:-

> MAPE(test[,8], predictions\_DT)

[1] 12.61215

Here, we get a error of 12.61% which indicates that our model is giving the accuracy of a 87.69%, which is a good result.

1. Random Forest:-

> #Random Forest

> MAPE(test[,8], RF\_Predictions)

[1] 5.995044

Here, we get a error of aproximate 6% which indicates that our model is 94% accurate which shows that the result is better than the Decission Tree.

1. Multiple Regression:-

> MAPE(test[,8], predictions\_LR)

[1] 3.216991

Here, the error we get a error of 3.2% which indicates that our result is 96.8% accurate which is the best one.

**3.2 Model selection**

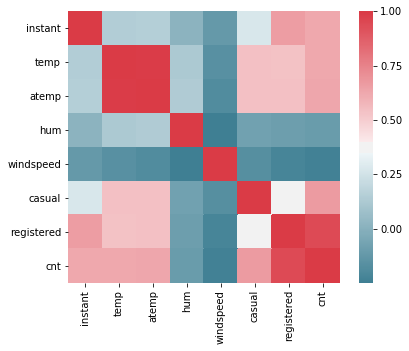
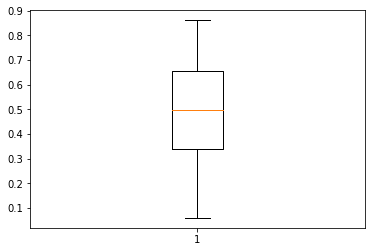
After analyzing all the models and checking the evaluation result the best model is the Multiple Regression which gives the accuracy of 96.8%. so, for the given data we choose the multiple Regression Model for predictions.

The casual and registered users cand be calculated independently, but the data is very less and it does not contains the hour variable, which shows the bike rented in certain interval of time. So calculating the data independtly causes more error due to less data.

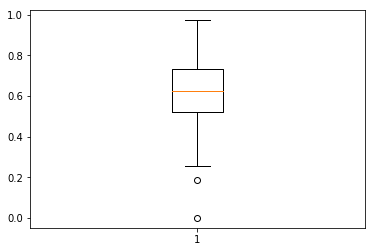
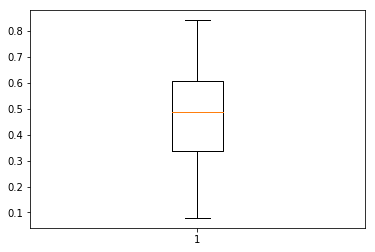
And the casual and registered user are counted on the daily basis not the hourley basis, so to get the better result the model is developed on combining the casual and registered data. As both the variable combine to give the total count users who rent the bike. So the cnt is highly dependent on the casual and registered users.

**Appendix A - Extra Figures**

Correlation diagram in Python Box Plot for Temp in Python

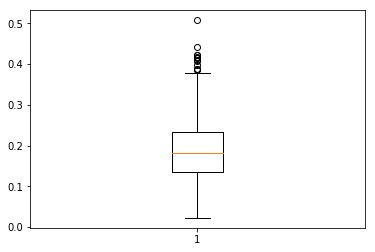
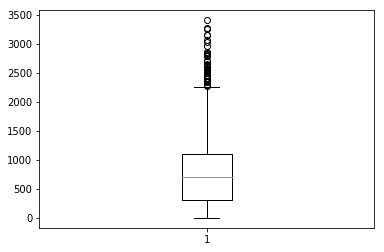
**** ****

Box Plot for atemp in Python Box Plot for hum in Python

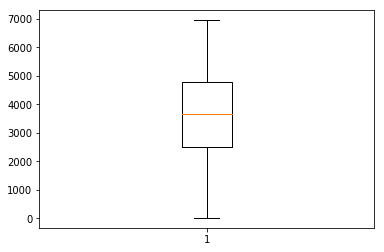
****

Box Plot for casual in Python

Box Plot for windspeed in Python

**** ****

Box Plot for registered in Python

****

**Appendix B - Code**

**Complete R Code**

## #Remove all objects stored

## rm(list=ls())

## #set current working directory

## setwd("E:/")

## #Current working directory

## getwd()

## #Load Libraries

## x = c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50", "dummies", "e1071", "Information",

## "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine', 'inTrees','gplots','scales','psych')

## lapply(x, require, character.only = TRUE)

## rm(x)

## ## Read the data

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

## ###########################################Explore the data##########################################

## str(data)

## #count of unique values in a columns

## length(unique(data$season))

## #convert into factor as only four repeated values present in the seasons

## data$season=as.factor(data$season)

## #similarly calculating count of unique values in a colums and convert it into the appropriate data types

## length(unique(data$yr))

## data$yr=as.factor(data$yr)

## length(unique(data$mnth))

## data$mnth=as.factor(data$mnth)

## length(unique(data$holiday))

## data$holiday=as.factor(data$holiday)

## length(unique(data$weekday))

## data$weekday=as.factor(data$weekday)

## length(unique(data$workingday))

## data$workingday=as.factor(data$workingday)

## length(unique(data$weathersit))

## data$weathersit=as.factor(data$weathersit)

## ##################################Missing Values Analysis###############################################

## table(is.na(data))

## missing\_val = data.frame(apply(data,2,function(x){sum(is.na(x))})) # counts the missinig value for each variable

## #There are no missing value present in the data

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

## # ## BoxPlots - Distribution and Outlier Check

## numeric\_index = sapply(data,is.numeric) #selecting only numeric

## numeric\_data = data[,numeric\_index]

## cnames = colnames(numeric\_data)

## for(i in 1:length(cnames))

## {

## assign(paste0("gn",i), ggplot(aes\_string(y = (cnames[i]), x = "cnt"), data = subset(data))+

## stat\_boxplot(geom = "errorbar", width = 0.5) +

## geom\_boxplot(outlier.colour="red", fill = "grey" ,outlier.shape=18,

## outlier.size=1, notch=FALSE) +

## theme(legend.position="bottom")+

## labs(y=cnames[i],x="count")+

## ggtitle(paste("Box plot of responded for",cnames[i])))

## }

## # ## Plotting plots together

## gridExtra::grid.arrange(gn2,gn3,gn4,ncol=3)

## gridExtra::grid.arrange(gn1,gn5,gn6,ncol=3)

## gridExtra::grid.arrange(gn7,gn8,ncol=2)

## #make copy of data

## df=data

## data=df

## # #Replace all outliers with NA and impute

## # #create NA on "windspeed"

## valw = data$windspeed[data$windspeed %in% boxplot.stats(data$windspeed)$out]

## print(length(valw))

## data$windspeed[data$windspeed %in% valw] = NA

## #For windspeed

## #Actual value[13]=0.30100

## #mean=0.19038

## #median=0.18

## #KNN = 0.25

## #Mean Method

## #data$windspeed[is.na(data$windspeed)] = mean(data$windspeed, na.rm = T)

## #Median Method

## #data$windspeed[is.na(data$windspeed)] = median(data$windspeed, na.rm = T)

## # kNN Imputation

## #data = knnImputation(data, k = 3)

## data = knnImputation(data, k = 3)

## # #create NA on "hum"

## valh = data$hum[data$hum %in% boxplot.stats(data$hum)$out]

## print(length(valh))

## data$hum[data$hum %in% valh] = NA

## #For hum

## #Actual value[16]=0.483750

## #mean=0.6280915

## #median=6270835

## #KNN = 0.5253335

## #Mean Method

## #data$hum[is.na(data$hum)] = mean(data$hum, na.rm = T)

## #Median Method

## #data$hum[is.na(data$hum)] = median(data$hum, na.rm = T)

## # kNN Imputation

## #data = knnImputation(data, k = 3)

## data = knnImputation(data, k = 3)

## # #create NA on "casual"

## valc = data$casual[data$casual %in% boxplot.stats(data$casual)$out]

## print(length(valc))

## data$casual[data$casual %in% valc] = NA

## #For casual

## #Actual value[3]=120

## #mean=849.174

## #median=717

## #KNN = 93.59666

## #Mean Method

## #data$hum[is.na(data$hum)] = mean(data$hum, na.rm = T)

## #Median Method

## #data$hum[is.na(data$hum)] = median(data$hum, na.rm = T)

## # kNN Imputation

## #data = knnImputation(data, k = 3)

## data = knnImputation(data, k = 3)

## ##################################Feature Selection################################################

## ## Correlation Plot

## corrgram(data[,numeric\_index], order = F,

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

## ## Chi-squared Test of Independence

## factor\_index = sapply(data,is.factor)

## factor\_data = data[,factor\_index]

## for (i in 1:8)

## {

## print(names(factor\_data)[i])

## print(chisq.test(table(factor\_data[,i])))

## }

## ## Dimension Reduction

## data = subset(data,

## select = -c(hum,atemp,season,yr,mnth,weekday))

## #Bar plot(categorical data)

## #Holiday

## ggplot(data, aes\_string(x = data$holiday)) +

## geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +ggtitle("Holiday Analysis")

## #workingday

## ggplot(data, aes\_string(x = data$workingday)) +

## geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +ggtitle("workingday Analysis")

## #weathersit

## ggplot(data, aes\_string(x = data$weathersit)) +

## geom\_bar(stat="count",fill = "DarkSlateBlue") + theme\_bw() +ggtitle("Wethersit Analysis")

## #histograms for the continous variable

## hist(data$temp)

## hist(data$windspeed)

## hist(data$casual)

## hist(data$registered)

## #casual #Scatter Plot

## ggplot(data, aes\_string(x = data$casual, y = data$weathersit)) +

## geom\_point(aes\_string(colour = data$holiday, shape = data$workingday),size = 4) +

## theme\_bw()+ ylab("weathersit") + xlab("casual") + ggtitle("Scatter plot Analysis")

## #registered #Scatter Plot

## ggplot(data, aes\_string(x = data$registered, y = data$weathersit)) +

## geom\_point(aes\_string(colour = data$holiday, shape = data$workingday),size = 4) +

## theme\_bw()+ ylab("weathersit") + xlab("Registered") + ggtitle("Scatter plot Analysis")

## ###################################Model Development#######################################

## #Divide data into train and test using stratified sampling method

## #Divide the data into train and test

## set.seed(123)

## train.index = createDataPartition(data$cnt, p = .80, list = FALSE)

## train = data[ train.index,]

## test = data[-train.index,]

## train = subset(train,

## select = -c(instant, dteday))

## test1 = test

## test = subset(test,

## select = -c(instant, dteday))

## dim(train)

## dim(test)

## ##Decision tree for Regression

## #Develop Model on training data

## # ##rpart for regression

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

## library("rattle")

## library("rpart.plot")

## library("RColorBrewer")

## fancyRpartPlot(fit)

## #Predict for new test cases

## predictions\_DT = predict(fit, test[,-8])

## ###Random Forest

## RF\_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 1000)

## #Presdict test data using random forest model

## RF\_Predictions = predict(RF\_model, test[,-8])

## #Linear Regression

## #check multicollearity

## library(usdm)

## vifcor(data[,6:9], th = 0.9)

## #run regression model

## lm\_model = lm(cnt ~., data = train)

## #Summary of the model

## summary(lm\_model)

## #Predict

## predictions\_LR = predict(lm\_model, test[,1:7])

## ###########################################MODEL EVALUATION############################################

## #calculate MAPE

## MAPE = function(y, yhat){

## mean(abs((y - yhat)/y))\*100

## }

## #Decision Tree

## # Accuracy=87.69%

## # Error=12.61%

## MAPE(test[,8], predictions\_DT)

## #Random Forest

## # Accuracy=94%

## # Error=6%

## MAPE(test[,8], RF\_Predictions)

## #Multiple Regression

## # Accuracy=96.8%

## # Error=3.2%

## MAPE(test[,8], predictions\_LR)

## sample=data.frame(instant=test1$instant, dteday=test1$dteday,cnt = predictions\_LR)

## write.csv(sample,file="Sample\_Data.csv",row.names=F)

**Python Code**

## import os

## os.chdir("E:")

## os.getcwd()

## #Load libraries

## import numpy as np

## from fancyimpute import KNN

## import matplotlib.pyplot as plt

## from scipy.stats import chi2\_contingency

## import seaborn as sns

## from random import randrange, uniform

## import pandas as pd

## day=pd.read\_csv("day.csv", sep=',')

## day.head(5)

## day.shape

## day.columns

## type(day)

## day['season'].unique()

## day['season'].nunique()

## day['season']=day['season'].astype(object)

## day['weathersit'].unique()

## day['weathersit'].nunique()

## day['weathersit']=day['weathersit'].astype(object)

## day['yr'].unique()

## day['yr']=day['yr'].astype(object)

## day['mnth'].unique()

## day['mnth']=day['mnth'].astype(object)

## day['holiday'].unique()

## day['holiday']=day['holiday'].astype(object)

## day['weekday']=day['weekday'].astype(object)

## day['workingday'].unique()

## day['workingday']=day['workingday'].astype(object)

## day['weathersit'].unique()

## #Explarotory Data Analysis

## #Missing Value Analysis

## #Create dataframe with missing percentage

## missing\_val = pd.DataFrame(day.isnull().sum())

## missing\_val

## # #Plot boxplot to visualize Outliers

## %matplotlib inline

## plt.boxplot(day['temp'])

## plt.boxplot(day['atemp'])

## plt.boxplot(day['hum'])

## plt.boxplot(day['windspeed'])

## plt.boxplot(day['casual'])

## plt.boxplot(day['registered'])

## #There are only hum,windspeed and casual variable which are having outliers

## #For hum

## #Actual value = 0.3010

## #Mean = 0.19038

## #Median = 0.18

## #KNN = 0.25

## #For windspeed

## #Actual value[13]=0.30100

## #mean=0.19038

## #median=0.18

## #KNN = 0.25

## #For casual

## #Actual value[3]=120

## #mean=849.174

## #median=717

## #KNN = 93.59666

## #Detect and replace with NA

## # #Extract quartiles

## iqr = q75 - q25

## # #Calculate inner and outer fence

## minimum = q25 - (iqr\*1.5)

## maximum = q75 + (iqr\*1.5)

## # #Replace with NA

## for i in range(len(day)):

## if((day['hum'].iloc[i]>maximum) | (day['hum'].iloc[i]<minimum)):

## day['hum'].iloc[i] = np.nan

## # #Calculate missing value

## missing\_val = pd.DataFrame(day.isnull().sum())

## # #Impute with KNN

## marketing\_train = pd.DataFrame(KNN(k = 3).complete(marketing\_train), columns = marketing\_train.columns)

## #Detect and replace with NA

## # #Extract quartiles

## q75, q25 = np.percentile(day['windspeed'], [75 ,25])

## # #Calculate IQR

## iqr = q75 - q25

## # #Calculate inner and outer fence

## minimum = q25 - (iqr\*1.5)

## maximum = q75 + (iqr\*1.5)

## # #Replace with NA

## for i in range(len(day)):

## if((day['windspeed'].iloc[i]>maximum) | (day['windspeed'].iloc[i]<minimum)):

## day['windspeed'].iloc[i] = np.nan

## #Detect and replace with NA

## # #Extract quartiles

## q75, q25 = np.percentile(day['casual'], [75 ,25])

## # #Calculate IQR

## iqr = q75 - q25

## # #Calculate inner and outer fence

## minimum = q25 - (iqr\*1.5)

## maximum = q75 + (iqr\*1.5)

## # #Replace with NA

## for i in range(len(day)):

## if((day['casual'].iloc[i]>maximum) | (day['casual'].iloc[i]<minimum)):

## day['casual'].iloc[i] = np.nan

## # #Calculate missing value

## missing\_val = pd.DataFrame(day.isnull().sum())

## missing\_val

## # #Impute with KNN

## day = pd.DataFrame(KNeighborsClassifier(n\_neighbors = 3).complete(day), columns = day.columns)

## #Feature Selection

## cnames = ["instant","temp","atemp","hum","windspeed","casual","registered","cnt"]

## ##Correlation analysis

## #Correlation plot

## df\_corr = day.loc[:,cnames]

## #Set the width and hieght of the plot

## f, ax = plt.subplots(figsize=(7, 5))

## #Generate 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)

## #Chisquare test of independence

## #Save categorical variables

## cat\_names = ["dteday","yr","mnth","weathersit","season","workingday","holiday","weekday"]

## #loop for chi square values

## for i in cat\_names:

## print(i)

## chi2, p, dof, ex = chi2\_contingency(pd.crosstab(day['cnt'], day[i]))

## print(p)

## day = day.drop(['yr','mnth','season','weekday','atemp','hum'], axis=1)

## from sklearn.cross\_validation import train\_test\_split

## from sklearn.tree import DecisionTreeRegressor

## #Divide data into train and test

## train, test = train\_test\_split(day, test\_size=0.2)

## train = train.drop(['instant','dteday'])

## test1 = test

## test = test.drop(['instant','dteday'])

## train.head(5)

## #Decision Tree

## # ## Decision Tree

## # In[ ]:

## #Decision tree for regression

## fit\_DT = DecisionTreeRegressor(max\_depth=2).fit(train.iloc[:,0:7], train.iloc[:,7])

## #Apply model on test data

## predictions\_DT = fit\_DT.predict(test.iloc[:,0:7])

## #Error Metric

## #Calculate MAPE

## def MAPE(y\_true, y\_pred):

## mape = np.mean(np.abs((y\_true - y\_pred) / y\_true))\*100

## return mape

## MAPE(test.iloc[:,7], predictions\_DT)

## #Multiple Regression

## #Import libraries ofr LR

## import statsmodels.api as sm

## #Train the model using the train sets

## model=sm.OLS(train.iloc[:,7],train.iloc[:,0:7]).fit()

## #Print Summary

## model.summary()

## #Make the Predictions by the model

## prediction\_LR = model.predict(test.iloc[:,0:7])

## #Calculate MAPE

## MAPE(test.iloc[:,7],prediction\_LR)

## #Writing to the directory

## sample = pd.DataFrame({"instant" = test1['instant'], "dteday" = test1['instant'], "cnt" = prediction\_LR})

## # Writing a csv (output)

## df\_csv.to\_csv("df\_csv\_practice.csv", index = False)