Bike Renting Prediction

MD SAMEER KHALEEL M

06 February 2019

Contents

Chapter 1

1 Introduction

1.1 Problem Statement:

The aim of the Project is to develop a model by which we can successfully predict the bike rental count on a daily basis based on environmental and seasonal changes. The riders are divided into 2 groups i) registered users which are those who frequently use this service and the other group is Casual which are first time users or users which use this service less frequently and have not yet registered. The sum of these two gives the total count of bike riders for a day. There are lot of factors to be taken into account for prediction like climate, day, wind and some other uncontrollable factors like traffic, cab strikes. For any business it is a imperative to have an idea about how much resource would be used so that they can prepare for it and utilize that information to better their business like bygiving up extra bikes on some days and hiring extra bike on other days

1.2 Data:

Our task is to build a model which will predict the daily count of bike rents based on environmental and season changes, Given below is the sample of the dataset
Sample Data (Columns: 1-8)

instant	dteday	season	yr	mnth	holiday	weekday	workingday
1	01-01-2011	1	0	1	0	6	0
2	02-01-2011	1	0	1	0	0	0
3	03-01-2011	1	0	1	0	1	1
4	04-01-2011	1	0	1	0	2	1
5	05-01-2011	1	0	1	0	3	1

Sample Data (Columns: 9-16)

weathersit	temp	atemp	hum	windspeed	casual	registered	cnt
2	0.344167	0.363625	0.805833	0.160446	331	654	985
2	0.363478	0.353739	0.696087	0.248539	131	670	801
1	0.196364	0.189405	0.437273	0.248309	120	1229	1349
1	0.2	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.22927	0.436957	0.1869	82	1518	1600
1	0.204348	0.233209	0.518261	0.089565	88	1518	1606

This is the sample set we are using to predict the total count of riders the business can expect in a day. we have 12 Variables, using which we have to correctly predict the count of bike rentals per day.

Predictor Values

S.No	Predictor
1	dteday
2	season
3	yr
4	mnth
5	holiday
6	weekday
7	workingday
8	weathersit
9	temp
10	atemp
11	hum
12	windspeed

Dependent Variables

Casual: number of non-registered user
Registered: number of registered user
cnt: number of total rentals (registered + casual)

Chapter 2

2 Methodology

2.1 Preprocessing

First, we look at the data, looking at the data refers to exploring the data, cleaning the data checking for missing values and visualizing the data through graphs and plots. First we look at the size of the data and what variables contribute to our data.

```
'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 ...
```

We infer from above the shape of the data set and the variables which contribute to our data, Now, we transform the dataset in the following ways so that we can get started up with our EDA

Creating new columns from "dteday" column.

Changing the data type of season, holiday, workingday and weathersit to category for EDA

Dropping the dteday column and instant column as we already extracted useful features from it. Now, we check for missing values since there are no missing values in the dataset we can move on with EDA.

```
table(is.na(df))
```

```
FALSE
11696
```

First we will analyze all the probability distributions of the variables, For Regression ideally it is preferred to have the data normally distributed. In Fig 2.1, we can see the probability density functions for all Independent and dependent integer variables. We did not perform this on the variables yr and holiday as it is not necessary and we use histogram to infer information from them. The blue lines indicate Kernel Density Estimations (KDE) of the variable and the red lines represent the normal distribution, we can infer from the plot most variables resemble normal distribution. Also we have the histogram for all the variables which can be viewed in the Appendix A

2.2 Exploratory Data Analysis

From using histogram method we have seen that the registered users are more than casual with the max users being 6946 users with gradually increasing peak raching a max of 4000 on a single day, the minimum is 20 users every day.

Casual users range from 2-3417 with high peaks from 0-1000 and avg peaks from 1000-1500

Season has four categories of almost equal distribution

Weather has 3 categories and weather 1 has higher contribution followed by weather 2 and decreases by weather 3

Month has 12 categories and there is a gradual increase with April-November having the highest volume

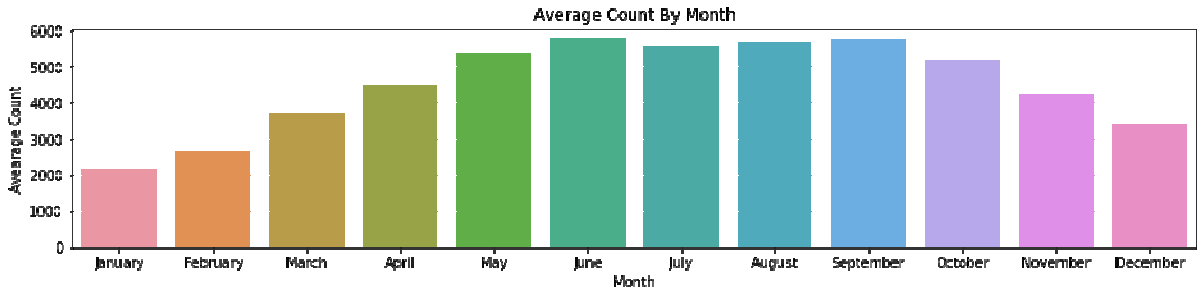

Fig 2-1 Histogram of Month

2.3 Outlier analysis:

Outlier is an observation which is inconsistent with the rest of the dataset there are many techniques to remove outliers from the dataset, graphical technique is the boxplot method, statistical technique is the Grubbs test, the outlier package in r and using experiment like replacing the outliers with NA, the disadvantage with Grubbs test is that it works only on normally distributed data and very few datasets are normally distributed, r package uses the mean concept, it will calculate the mean of whole variables and compare all the values with mean, and if the values is falling away from the mean it will count as an outlier these method are less effective than the Boxplot as boxplot gives the accurate model for regardless the data being normalized or not.From the histograms and Probability Distributions we see that a few of the variables are skewed which are most likely due to the presence of outliers, In fig 2.3, we have plotted the boxplot of all the 8 independent variables and the 3 dependent variables. The boxplot for the count and working day shows that the count on holidays are slightly higher than on working days, to make more observations on this The following inferences can be made from this

Spring has the least number of people opting to ride bikes among the four seasons (Fig:2.3)

We see that the holidays have slightly more number of people opting for biking,(Fig: 2.4)

Among holidays Sunday is more favoured among the users to ride the bike (Fig:2.4)

Casual has a few outlines which could be due to a group of people having a group exercise or an event

Similary holiday has a few outliners which could be for similar reasons. Computing the outliners with the mean would be optimal

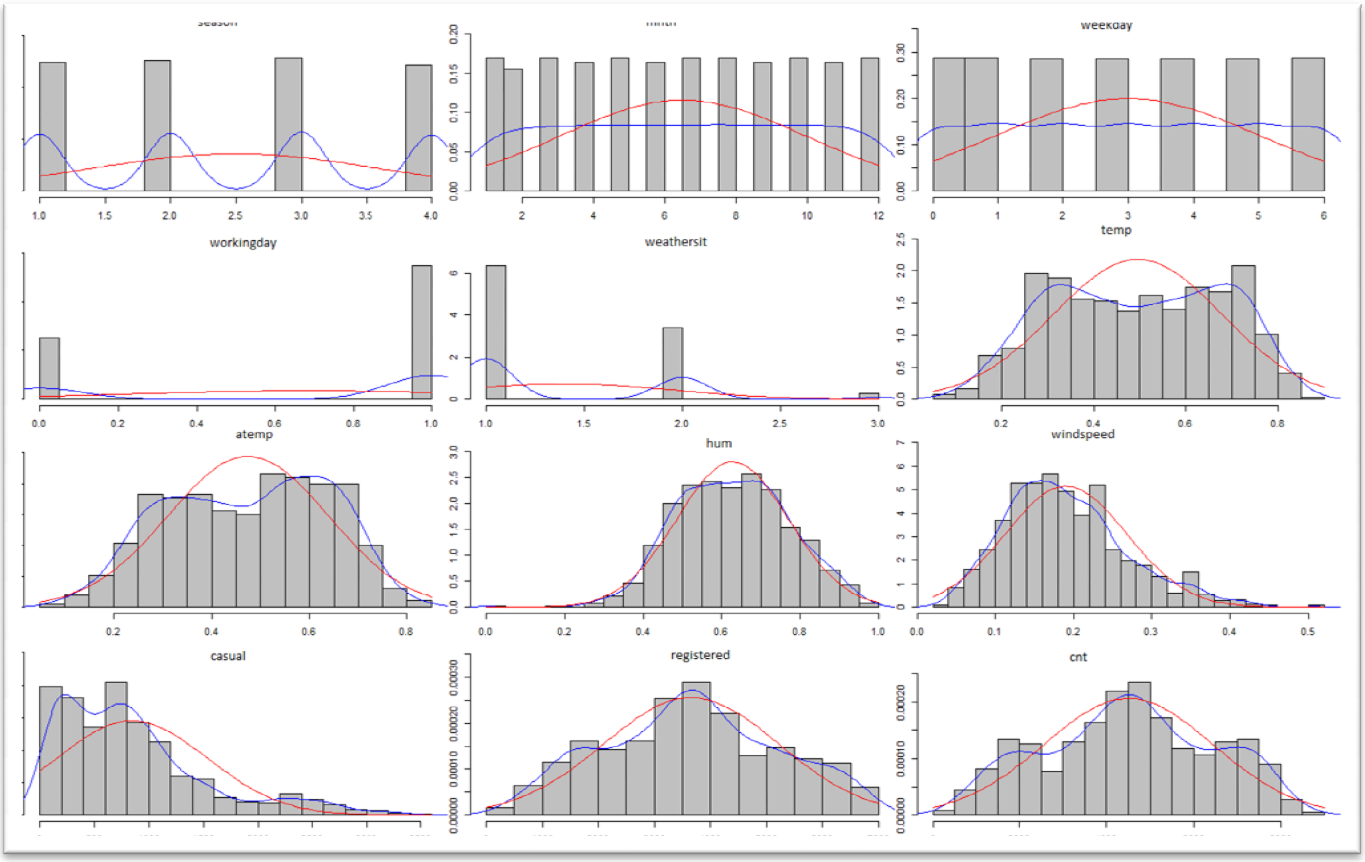

Fig 2-2: Probability Distribution plot

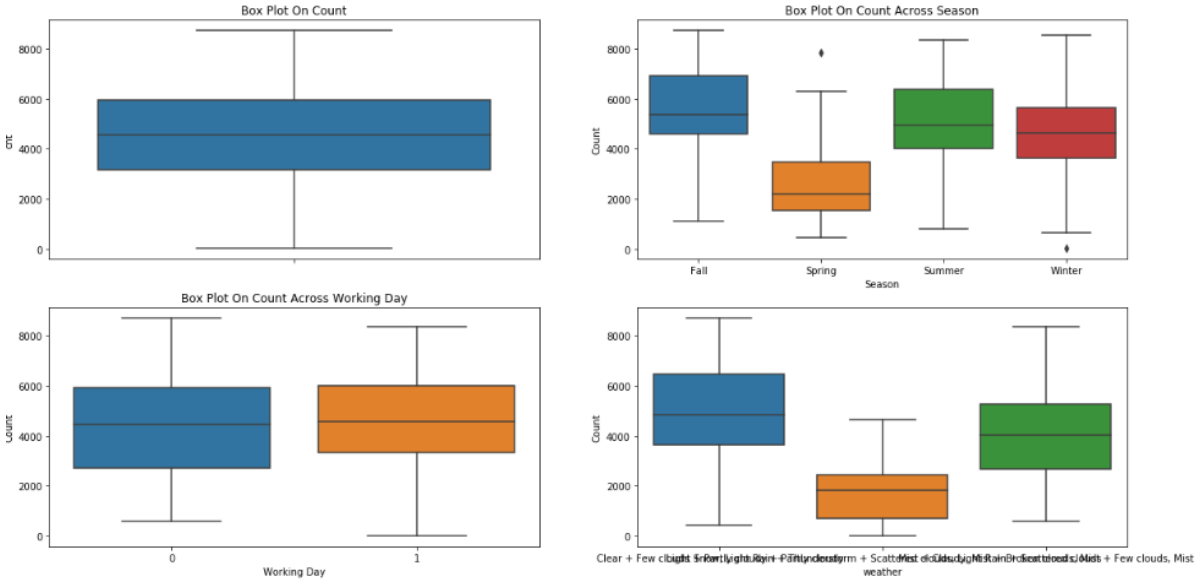

Fig 2-3 Variables Vs Count of total users Boxplot

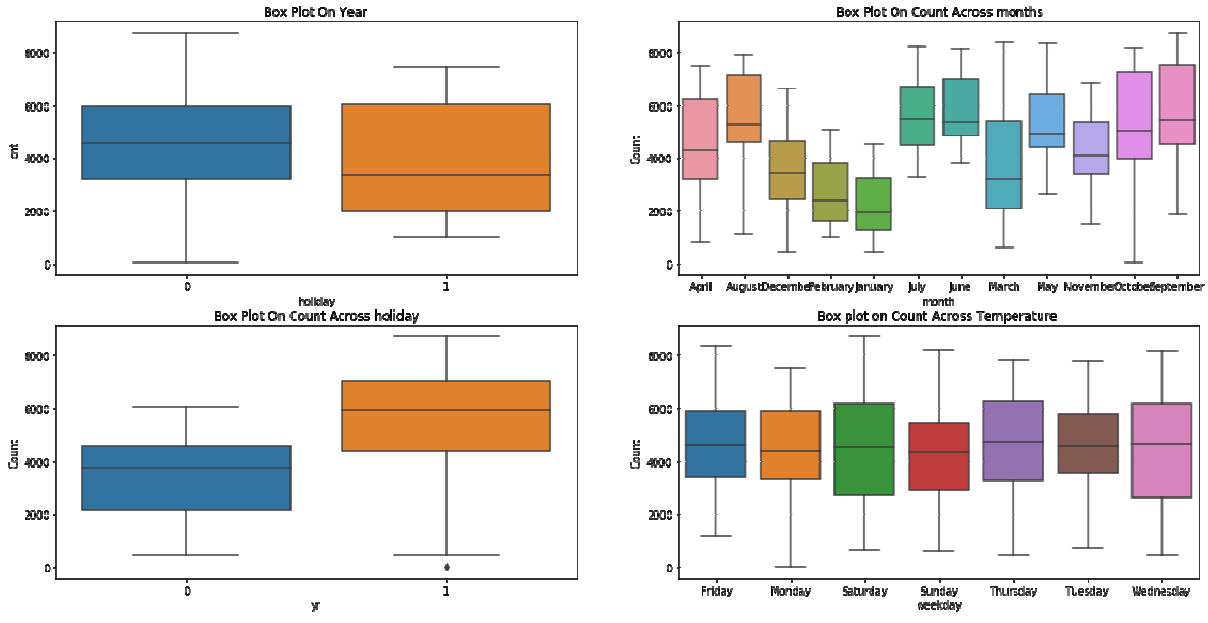

Fig 2-4 Variables Vs Count of total users Boxplot

2.3.1 Feature Selection

1. Feature Selection is an important step in data processing and modelling as it can help us avoid unnecessary variables which might bloat our model and can even have negative effects on our model one method to check which variable doesn't give much information to the model is via correlation analysis Feature selection is selecting a subset of feature from the dataset which reduces the complexity of the data, time consumption and memory consumption, correaltion applies on numeric data or the continuous variable, Correlation coefficient can be calculated by the formula (1-r2)=1/VIF, from the original formula VIF = $1/(1 - r^2)$,

Correlation analysis:

Feature Scaling: since the independent variables are already normalized we don't proceed to normalization

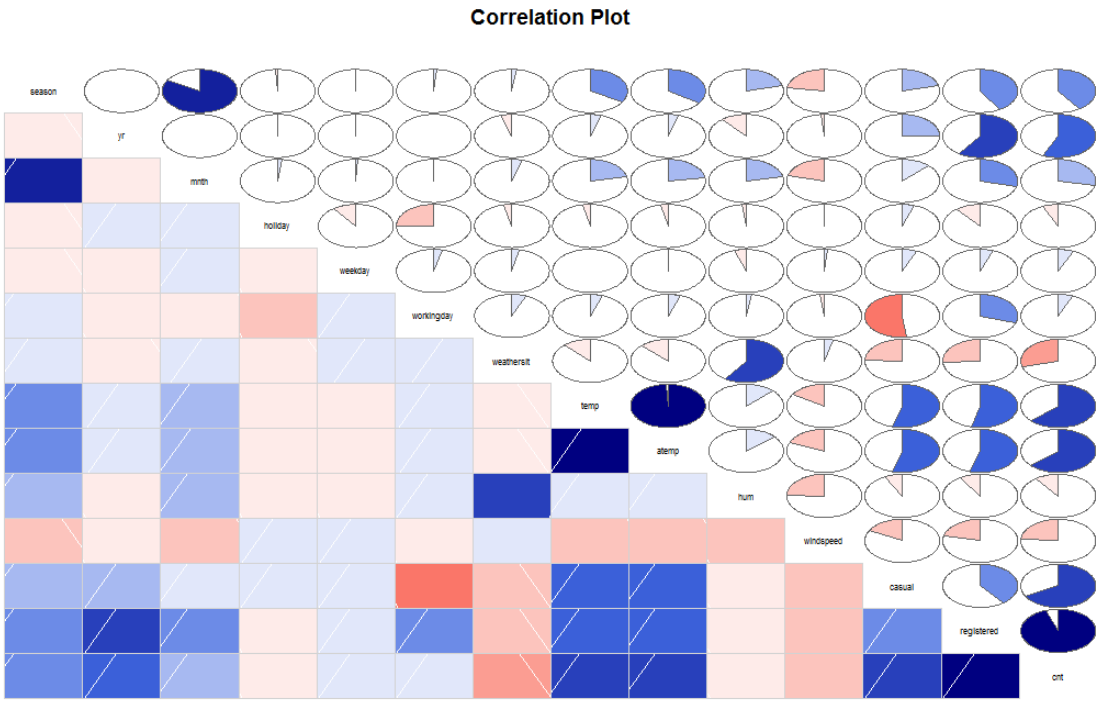

Fig 2-5 Corelation Plot

Since temp and atemp are highly correlated we can remove one of them from the anaylsis. We remove atemp from the model. Now we also need to know the importance of each variable in the data set,

we use Linear Regression method to achieve this

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

Residuals:
    Min      1Q  Median      3Q     Max
-4014.1  -422.6    17.0   511.6  2973.9

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1634.66     261.28   6.256 7.71e-10 ***
season        462.85      63.80   7.255 1.31e-12 ***
yr           2061.45      73.29  28.129  < 2e-16 ***
mnth          -19.87      19.93  -0.997   0.3191
holiday      -315.11     220.01  -1.432   0.1526
weekday        81.24      18.58   4.372 1.46e-05 ***
workingday    182.06      82.28   2.213   0.0273 *
weathersit   -653.27      87.10  -7.500 2.45e-13 ***
temp         4985.60     217.96  22.874  < 2e-16 ***
hum          -872.53     352.53  -2.475   0.0136 *
windspeed   -2909.21     519.40  -5.601 3.31e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 877.9 on 573 degrees of freedom
Multiple R-squared:  0.794,    Adjusted R-squared:  0.7904
F-statistic: 220.8 on 10 and 573 DF,  p-value: < 2.2e-16
```

The stars indicate the significance to the target variable, we see that

season,yr,weekday,weathersit.temp.windspeed are highly significant.

holiday and mnth seems less significant to the target variable.

2.4 Modelling

We start with three cases Case 1 : Removing the outliers and replacing them with their mean and predicting Count

Case2: not removing the outliers and treating them as part of the data and predicting Count

Case 3: Computing the casual and registered users separately and adding them up.

2.5 Model selection

We know from preprocessing that the method to use would be regression method

2.5.1 Linear Regression

Linear regression helps us to predict a continuous dependent variable from a number of independent variables and help us to identify the relationship between two variables, Regression coefficients help us to determine the amount of information each independent variable is contributing, In simple or multiple linear regression, the size of the coefficient for each independent variable gives you the size of the effect that variable is having on your dependent variable, and the sign on the coefficient (positive or negative) gives you the direction of the effect. The summary of our model is below we can observe that

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

Residuals:
    Min      1Q  Median      3Q     Max
-4014.1  -422.6    17.0   511.6  2973.9

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1634.66     261.28   6.256 7.71e-10 ***
season        462.85      63.80   7.255 1.31e-12 ***
yr           2061.45      73.29  28.129  < 2e-16 ***
mnth          -19.87      19.93  -0.997   0.3191
holiday      -315.11     220.01  -1.432   0.1526
weekday        81.24      18.58   4.372 1.46e-05 ***
workingday    182.06      82.28   2.213   0.0273 *
weathersit   -653.27      87.10  -7.500 2.45e-13 ***
temp         4985.60     217.96  22.874  < 2e-16 ***
hum          -872.53     352.53  -2.475   0.0136 *
windspeed   -2909.21     519.40  -5.601 3.31e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 877.9 on 573 degrees of freedom
Multiple R-squared:  0.794,    Adjusted R-squared:  0.7904
F-statistic: 220.8 on 10 and 573 DF,  p-value: < 2.2e-16
```

That the adjusted R-Square, we can explain 79% of our model, looking at the F-satistic and combined P value, we can reject our null hypothesis which is that the target variable does not depend on the Predictor Variable.

2.5.2 Random Forest

The random forest model is very good at handling tabular data with numerical features, fewer than hundreds of categories. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target. The Gradient Boosted Regression Trees (GBRT) model (also called Gradient Boosted Machine or GBM) is one of the most effective machine learning models for predictive analytics, making it an industrial workhorse for machine learning. Unlike Random Forest which constructs all the base classifier independently, each using a subsample of data, GBRT uses a particular model ensembling technique called gradient boosting.

We see that Random Forest performs the best, so we choose Random Forest Method to predict the model. We get the accuracy MAPE error of 12%, which could be Gradient Boosted for even more accurate result

Chapter 3

3 Conclusion

3.1 Model Evaluation

After modelling a few models for predicting the count, we have to choose the right one based on

1. Prediction Performance
2. Complexity
3. Efficiency and the ability to mould according to data

   Our main aim is the Prediction Performance and hence we choose the model which gives high prediction performance. Prediction Performance can be measured by comparing Predictions of the models with real values, the avg error is used to determine the best model

3.1.1 MAPE

MAPE is Mean Absolute Percentage Error, It measures accuracy as a percentage of error. It is the mean of actual – predicted by actual values. It gives a clean description of the error in percentage. This is highly preferred in regression models

3.1.2 MAE

This is the mean absolute Error. It is the average of absolute errors. It gives the output in terms of a numerical value the closer the value is to zero the better the model.

3.1.3 RMSE

This is Root Mean Squared Error. This is the average of the square of errors taken in square roots, this method also shows the error in a clear manner and can help us to find the efficiency of the model. This is mostly used in Time based prediction

regr.eval(test[,11],RF_Predictions,stats=c('mse','rmse','mape'))

```
> regr.eval(test[,11],RF_Predictions,stats=c('mse','rmse','mape'))
          mse          rmse          mape
466784.76165    683.21648      1.27803

>
```

3.2 Model Selection

We use the MAPE method and the RMSE method to predict the model and we decide based on MAPE method for accurate result

Appendix A- Extra Figures

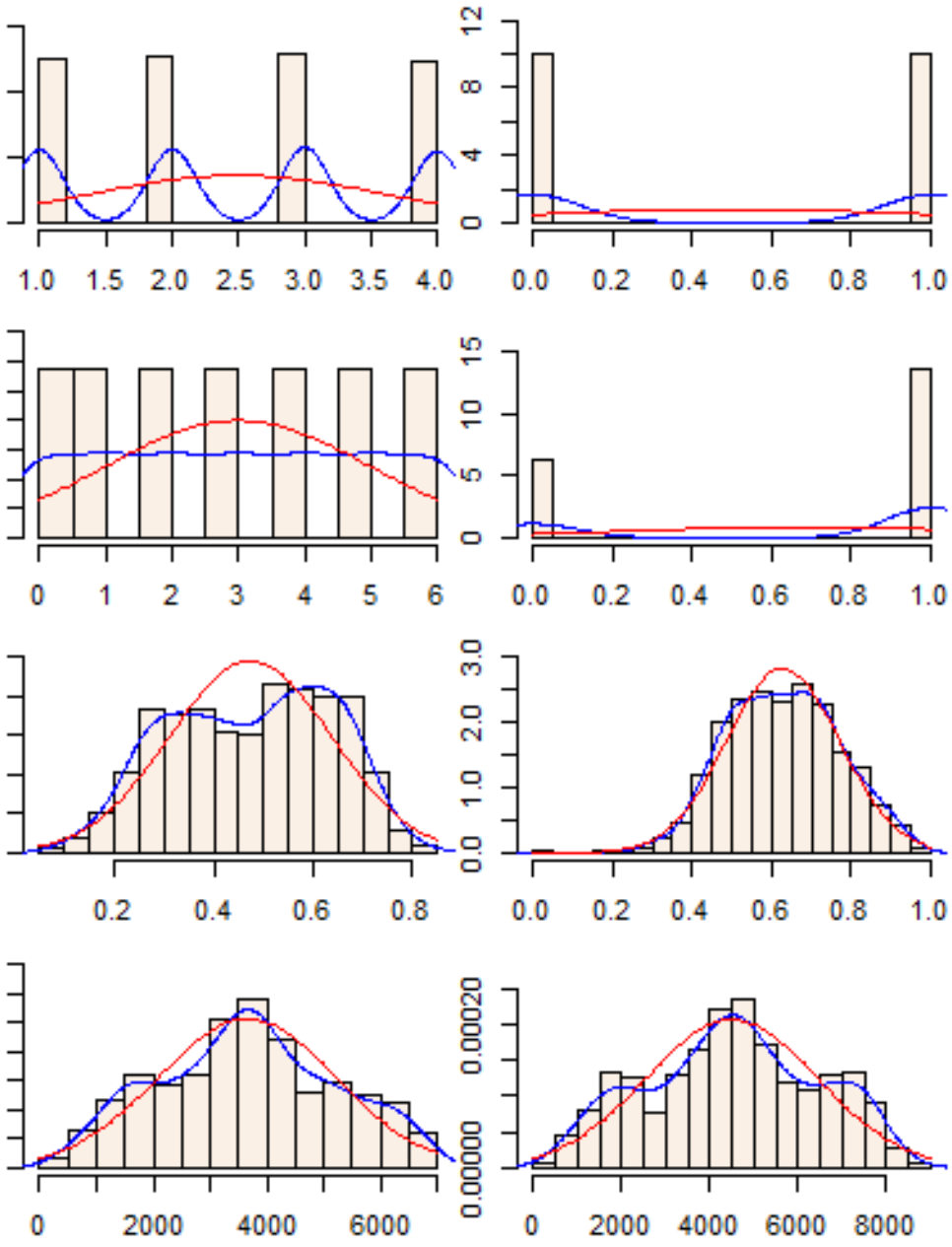

Fig 3-1 Probability Density Function

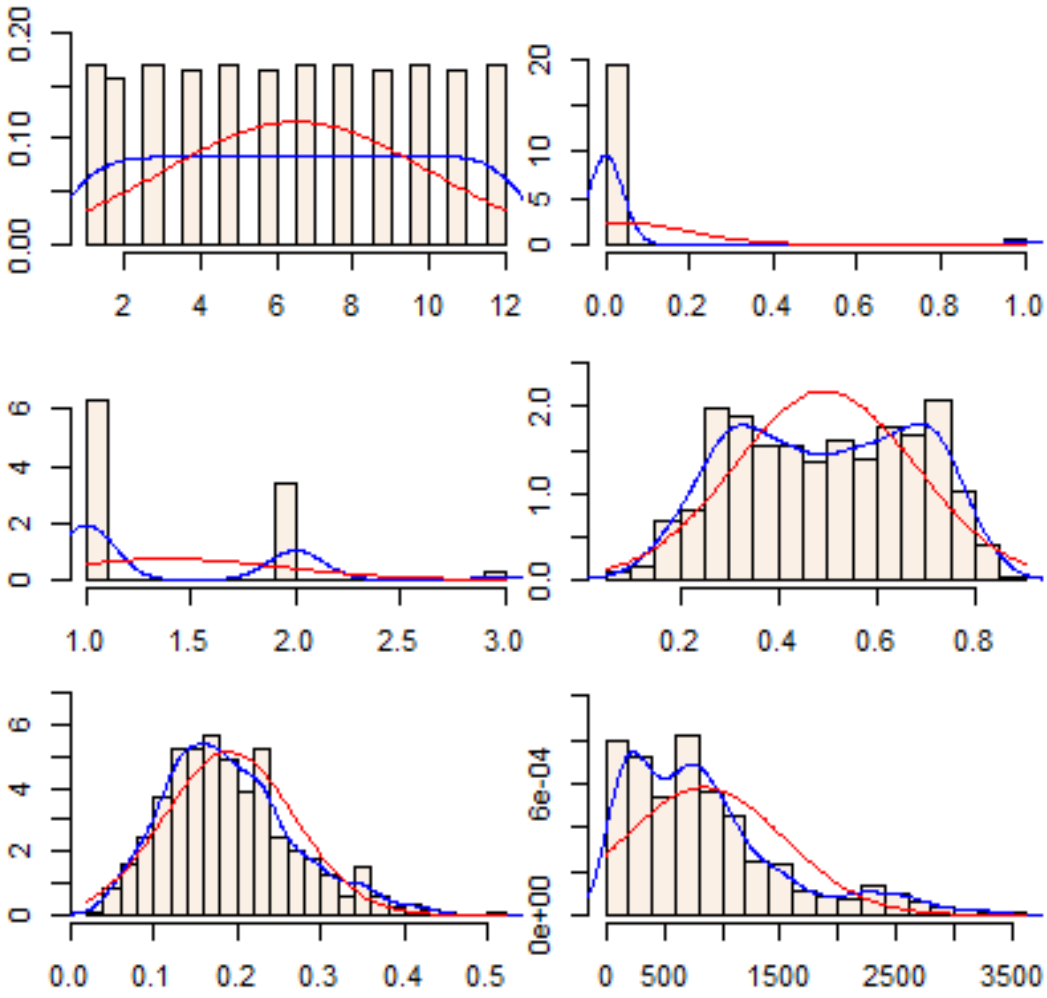

Fig 3-2 Probability Density Function

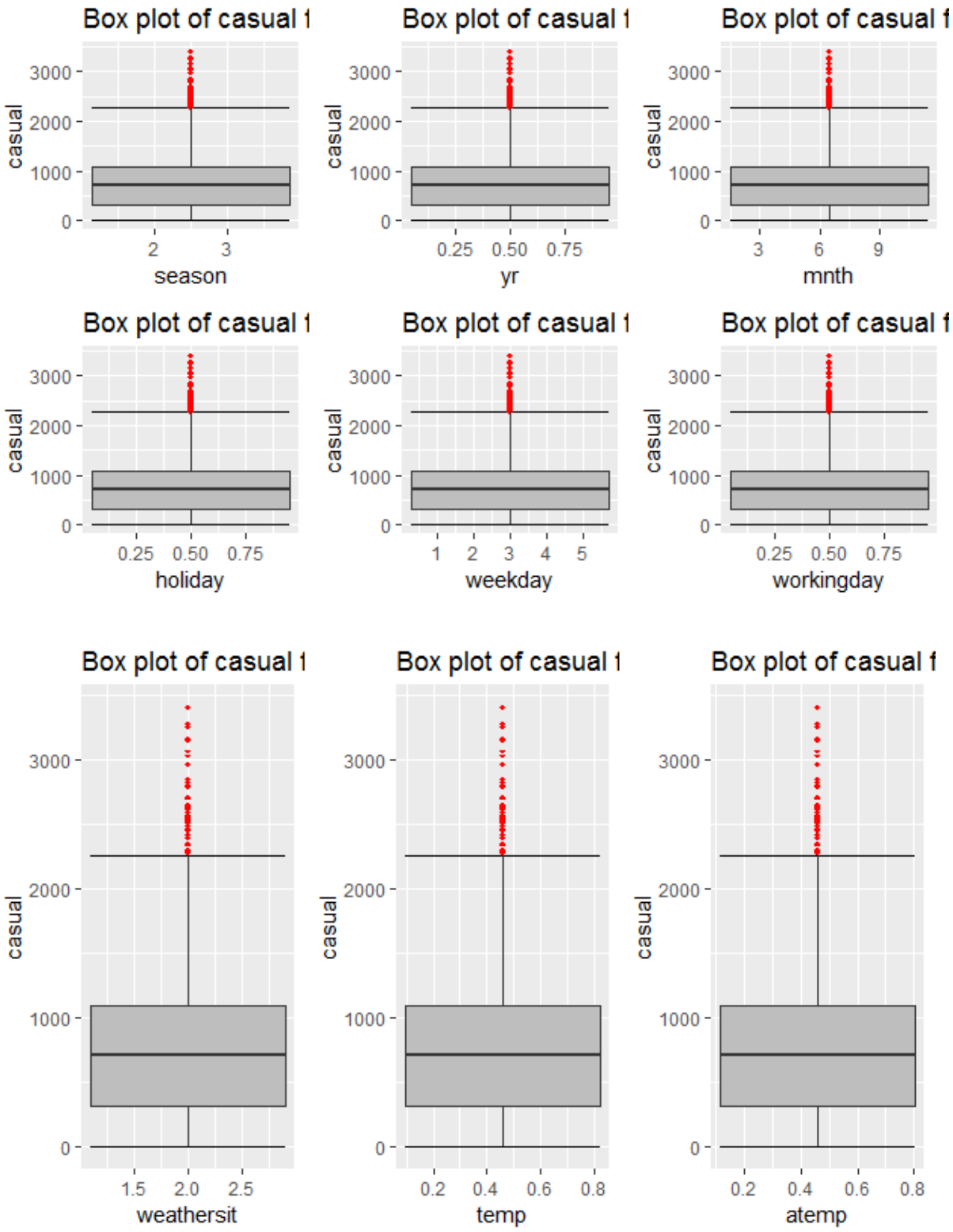

Fig 3-3 Variables Vs Casual users Boxplot

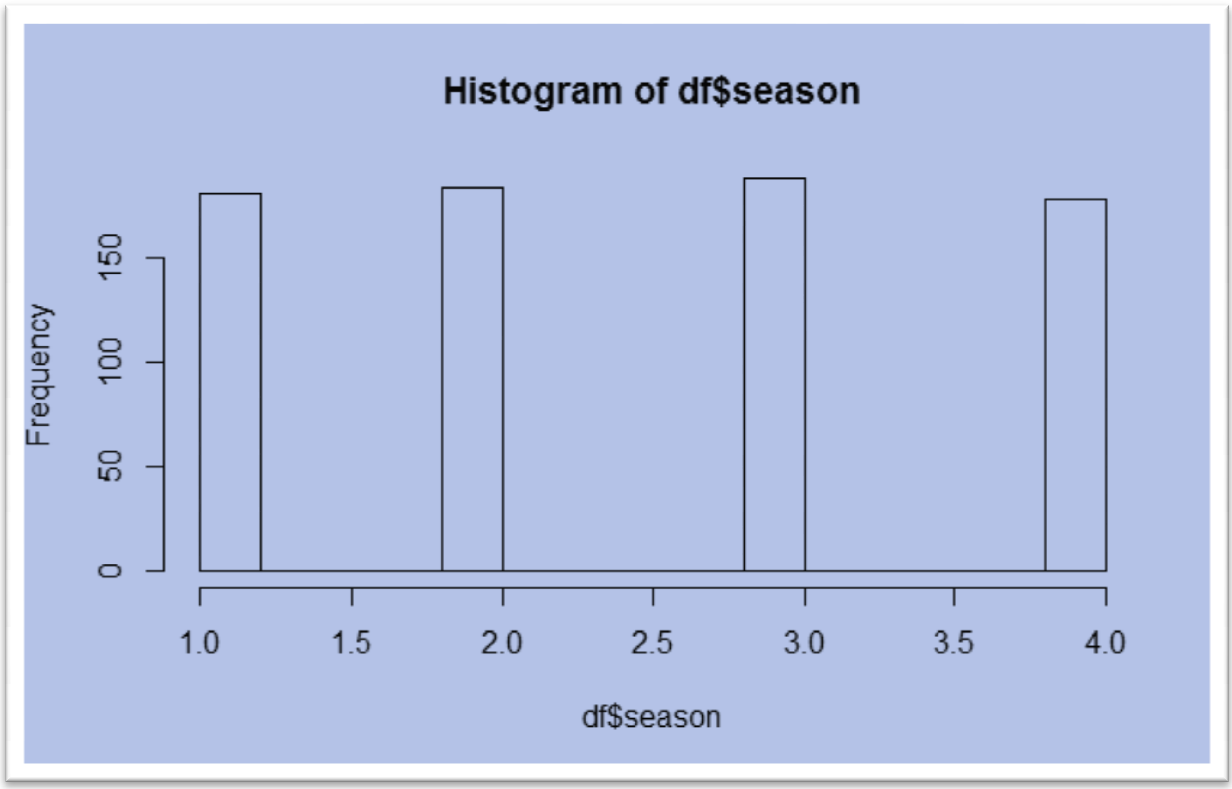

Fig 3-4 Histogram of Season

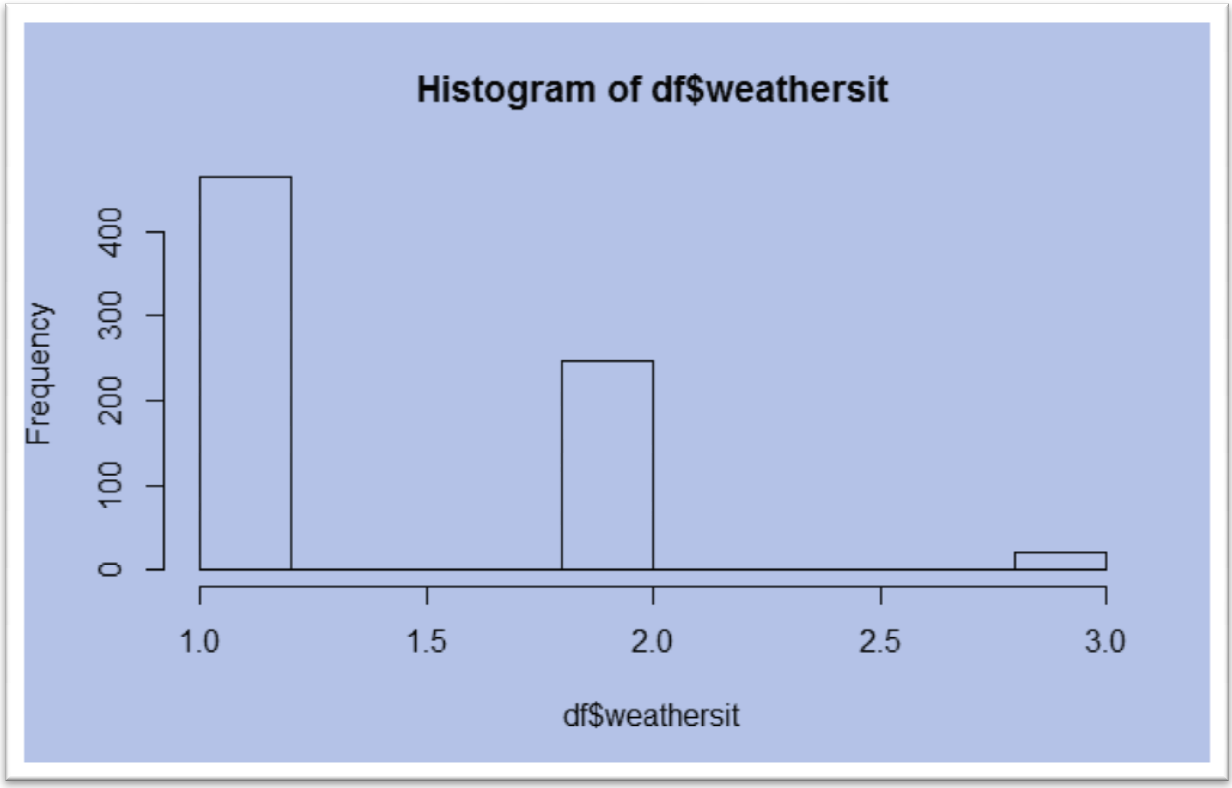

Fig 3-5 Histogram of Weather

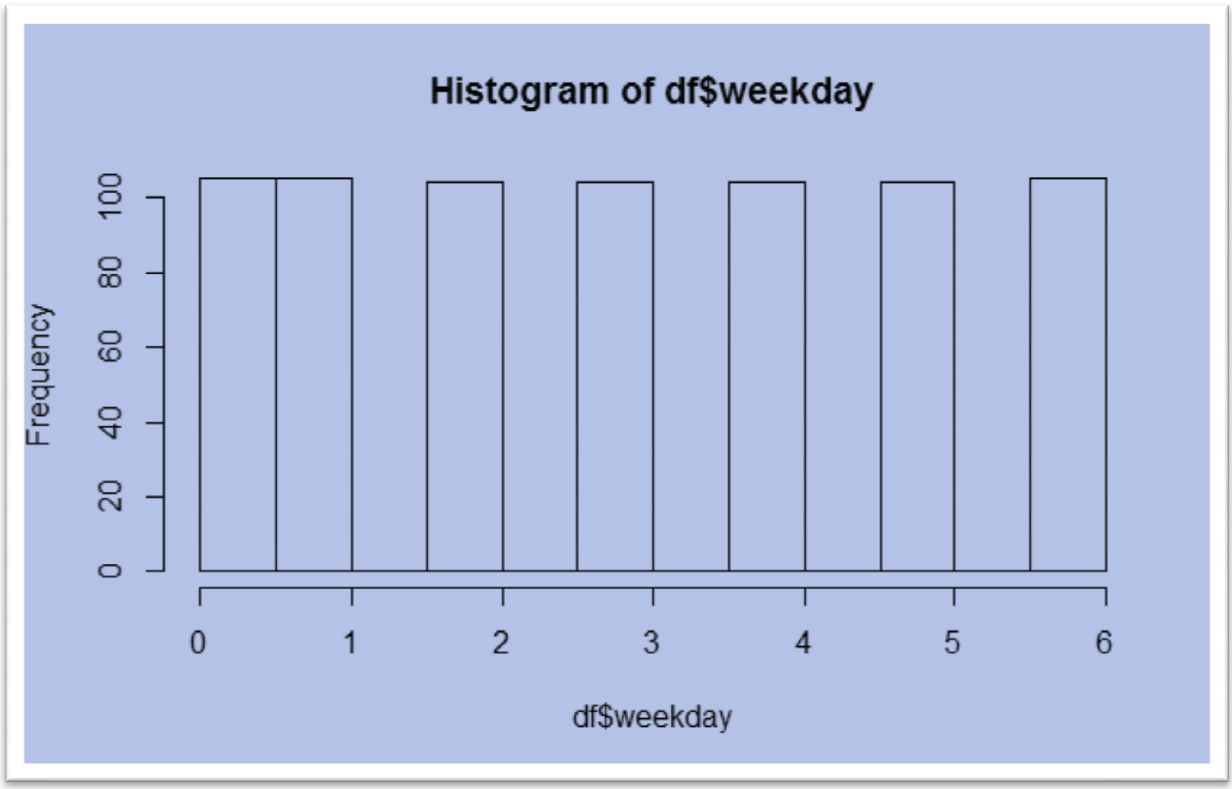

Fig 3-6 Histogram of Weekday

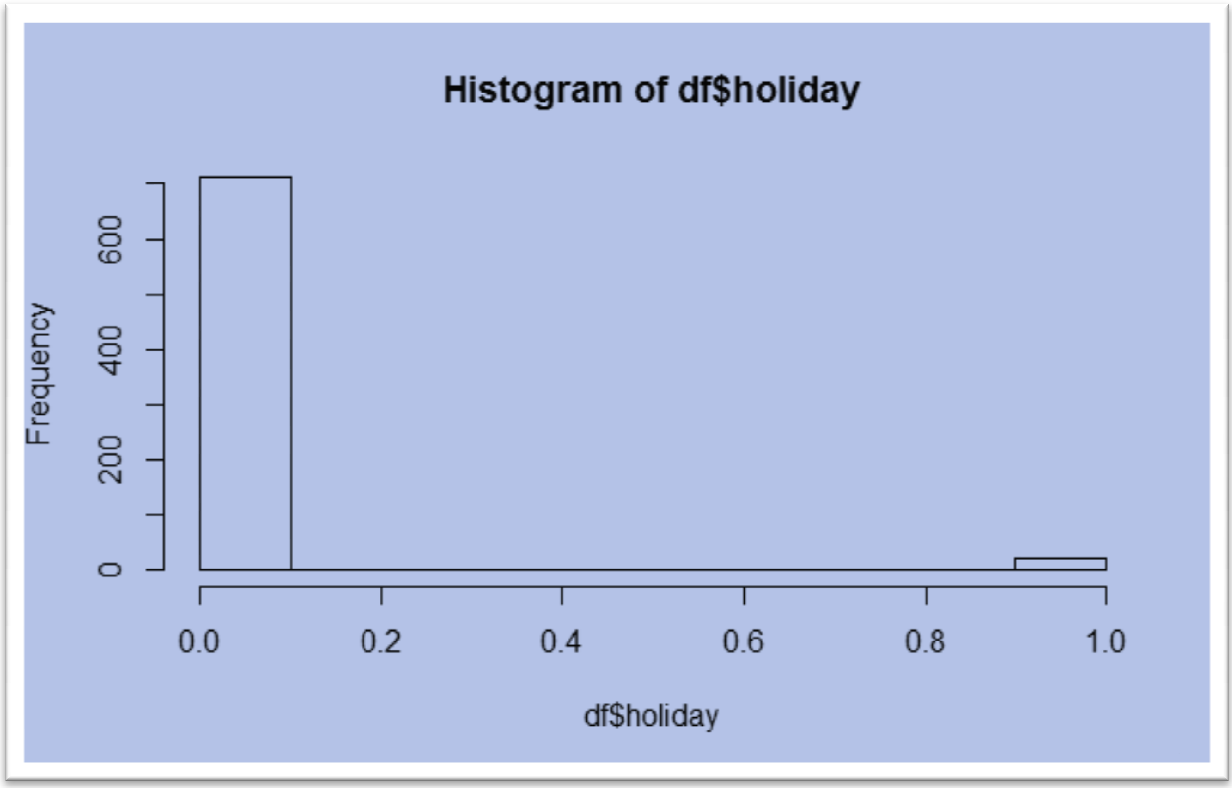

Fig 3-7 Histogram of Holiday

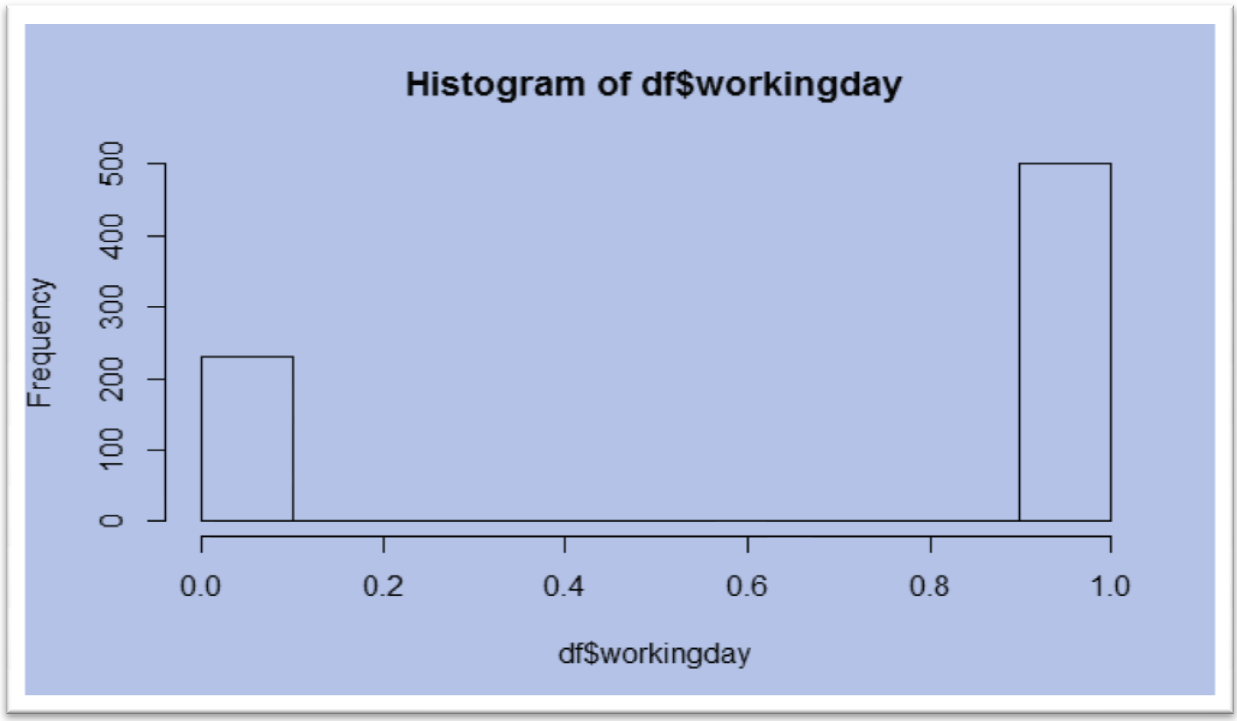

Fig 3-8 Histogram of Working day

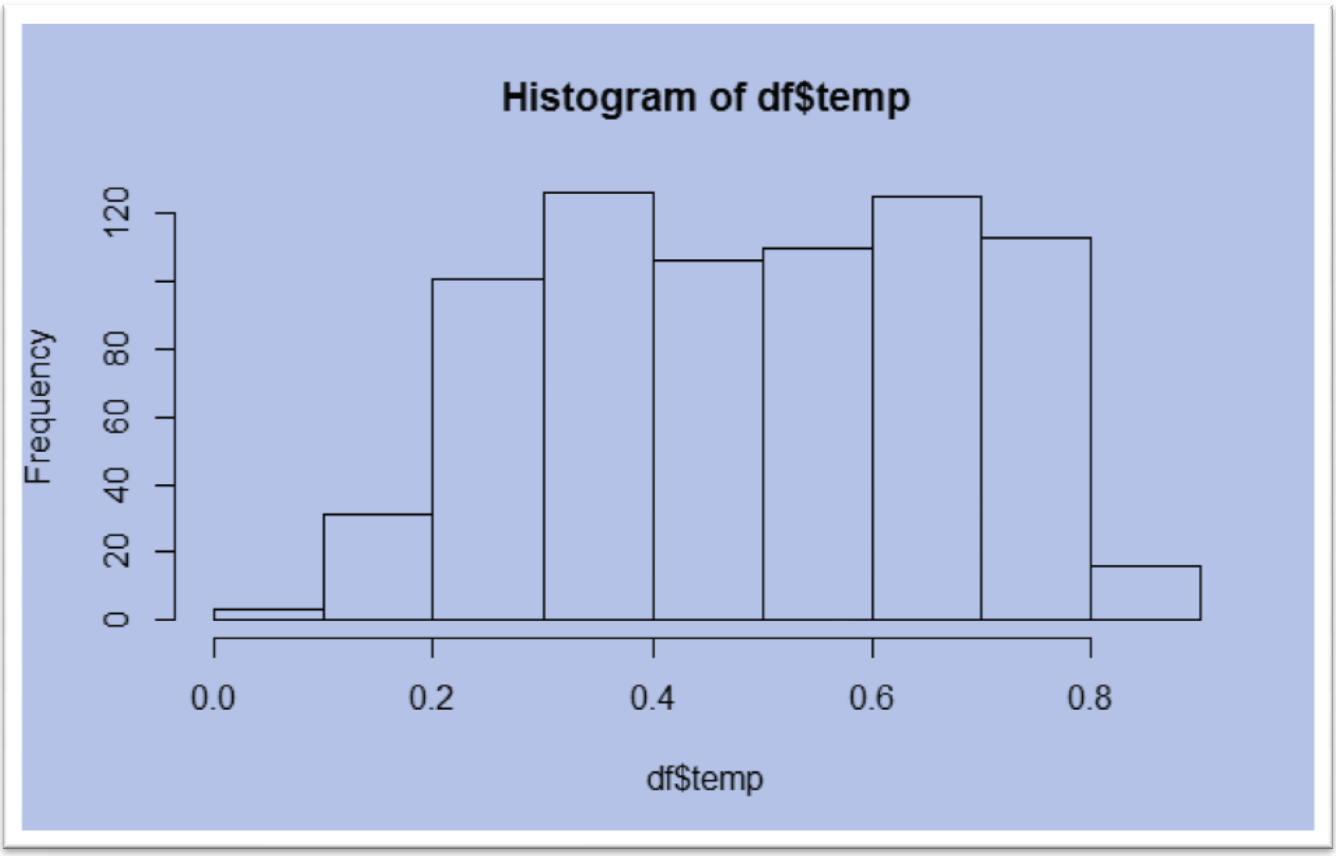

Fig 3-9 Histogram of Temperature

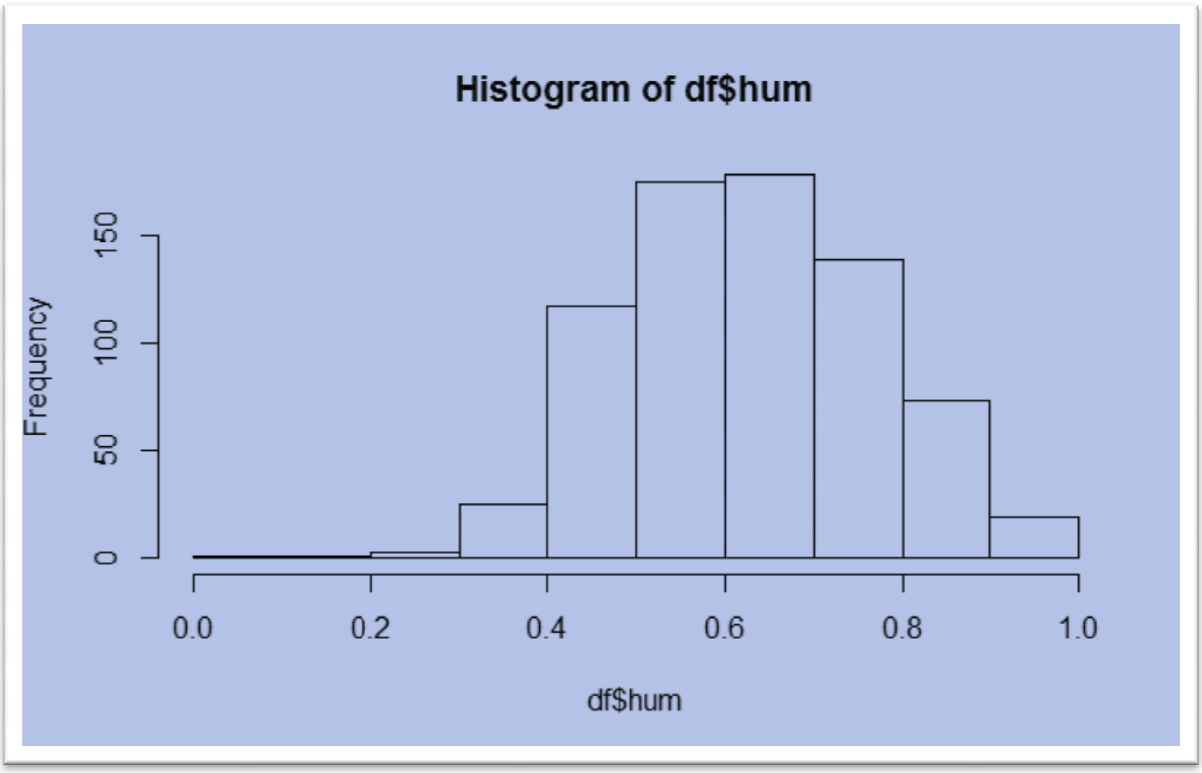

Fig 3-10 Histogram of Humidity

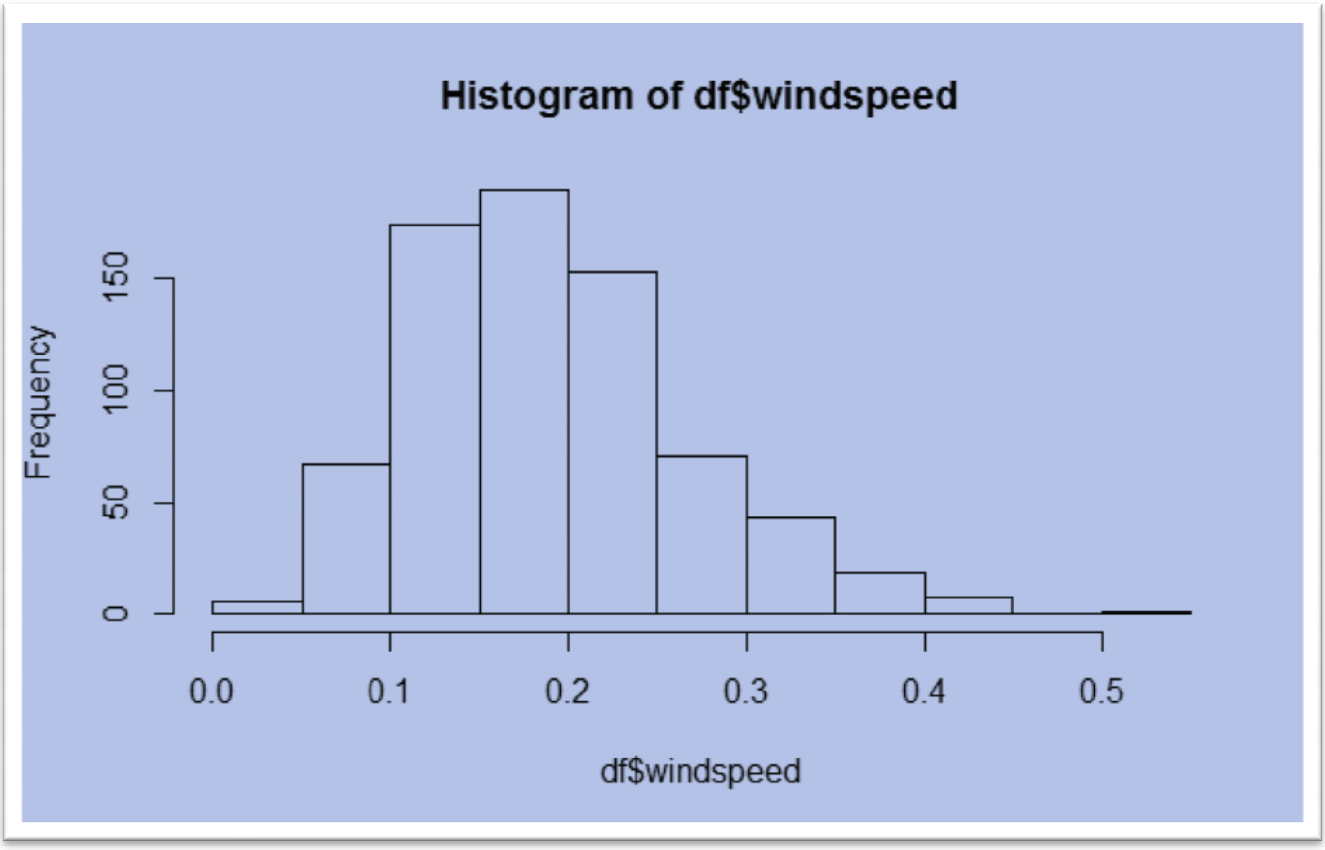

Fig 3-11 Histogram of windspeed

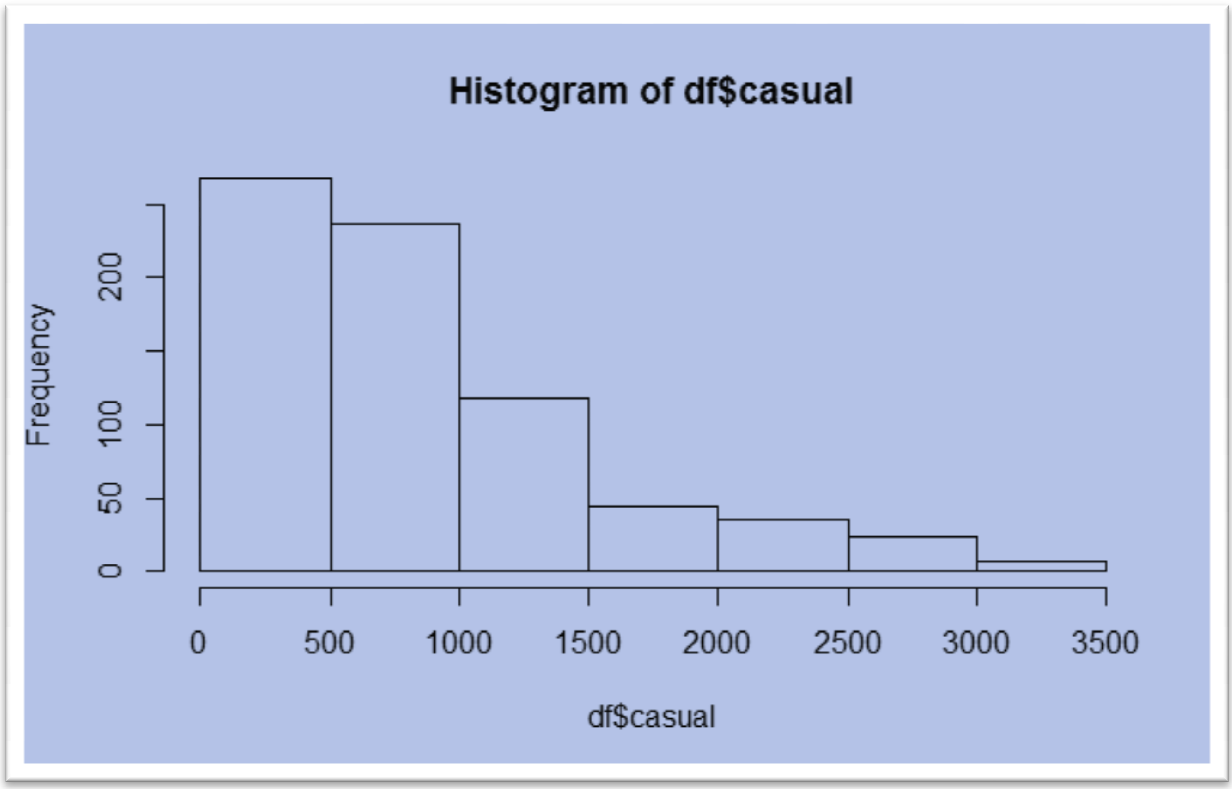

Fig 3-12 Histogram of Casual riders

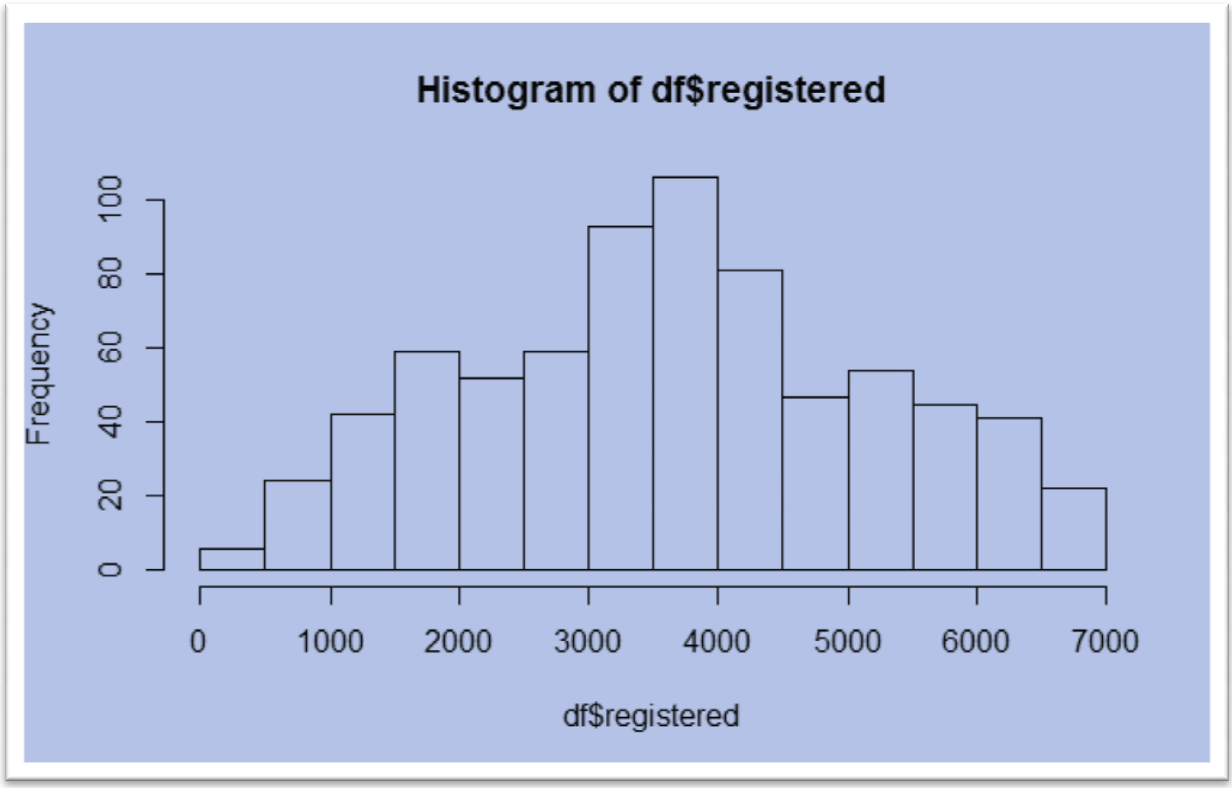

Fig 3-13 Histogram of registered riders

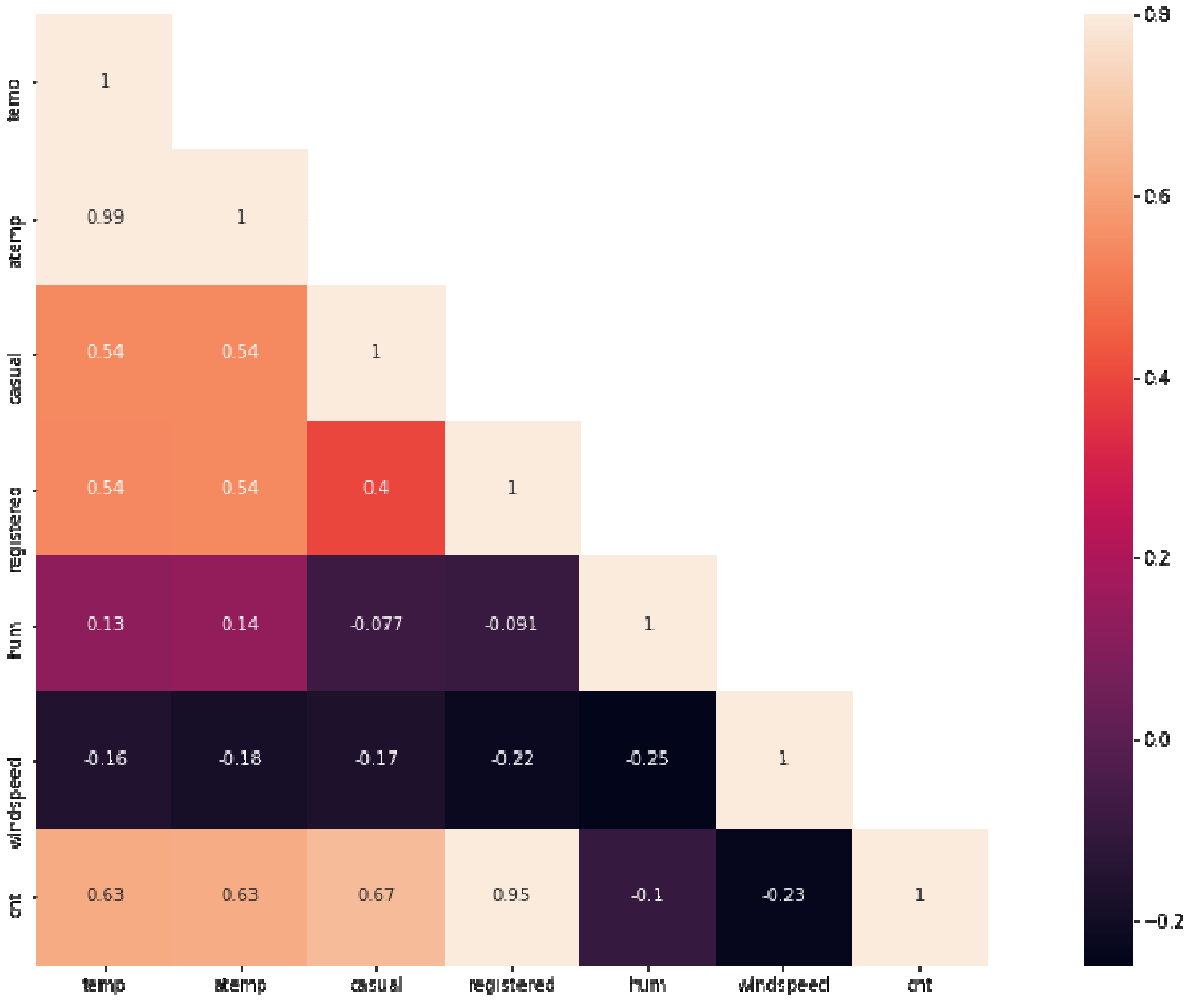

Fig 3-14 Co-relation plot

Appendix B – R Code

Histogram:

```
multi.hist(df,main=NA,dcol=c("blue","red"),dlty=c("solid","solid"),bcol="grey")
```

Boxplot analysis:

```
For count
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "count"), data =subset (df))+
          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(x=cnames[i],y="count")+
          ggtitle(paste("Box plot of total count for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
```

For Casual

```
numeric_index = sapply(df,is.numeric)
numeric_data = df[,numeric_index]
cnames = colnames(numeric_data)
#box plot for casual riders
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "casual"), data =subset (df))+
          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(x=cnames[i],y="casual")+
          ggtitle(paste("Box plot of casual for",cnames[i])))
}


For Registered

for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "registered"), data =subset
(df))+
```

```
        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(x=cnames[i],y="registered")+
        ggtitle(paste("Box plot of registered for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
```

Complete R file

```
#clean the enviroment
rm(list=ls())
#set working directory
setwd("E:/study/data")
getwd()
#read the input dataset
df=read.csv("day2.csv")
DB=df
# importing library
x=c("ggplot2", "corrgram", "DMwR", "caret", "randomForest", "unbalanced", "C50",
"dummies", "e1071", "Information",
    "MASS", "rpart", "gbm", "ROSE", 'sampling', 'DataCombine',
'inTrees','knitr','tidyr','xtable','MASS','psych')
lapply(x, require, character.only = TRUE)
rm(x)
library(DMWR)
#Expoloring the data
str(df)
multi.hist(df,main=NA,dcol=c("blue","red"),dlty=c("solid","solid"),bcol="grey")
#Checking for any missing values
table(is.na(df))

#boxplot analysis
#removing instant from dataset
df$instant=NULL
#cnames
numeric_index = sapply(df,is.numeric)
numeric_data = df[,numeric_index]
cnames = colnames(numeric_data)
#box plot for casual riders
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "casual"), data =subset (df))+
        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(x=cnames[i],y="casual")+
        ggtitle(paste("Box plot of casual for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
# box plot for registered
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "registered"), data =subset
(df))+
        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(x=cnames[i],y="registered")+
        ggtitle(paste("Box plot of registered for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)
# box plot for total count
for (i in 1:length(cnames))
{
  assign(paste0("gn",i), ggplot(aes_string(x = (cnames[i]), y = "count"), data =subset (df))+
        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(x=cnames[i],y="count")+
        ggtitle(paste("Box plot of total count for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,gn4,gn5,gn6,ncol=3)
gridExtra::grid.arrange(gn7,gn8,gn9,ncol=3)


#histogram - preprocessing
h1=hist(df$season)
hist(df$weathersit)
hist(df$weekday)
hist(df$holiday)
hist(df$workingday)
hist(df$temp)
hist(df$atemp)
hist(df$hum)
hist(df$windspeed)
hist(df$casual)
hist(df$registered)
#correlation analysis
df$dteday=NULL
numeric_index = sapply(df,is.numeric)
```

```
corrgram(df[,numeric_index], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
sub=data.frame(train$registered,train$casual,train$cnt,train$temp,train$hum,train$atemp,trai
n$windspeed)
cor(sub)
#since atemp is highly corelated we remove atemp from the dataset
df$atemp=NULL
# model 1 - removing outliers from variables
cnames = colnames(df)
#

# #Replace all outliers with NA and impute
# #create NA on "custAge

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

 df$holiday[is.na(df$holiday)] = mean(df$holiday, na.rm = T)
 df$hum[is.na(df$hum)] = mean(df$hum, na.rm = T)
 df$windspeed[is.na(df$windspeed)] = mean(df$windspeed, na.rm = T)
 df$casual[is.na(df$casual)] = mean(df$casual, na.rm = T)
 table(is.na(df))

 #Model Development
 #remove dependent variables except the one which is to be targeted
 #df2 =df
 df=df2
 df$casual=NULL
 df$registered=NULL
 train.index = createDataPartition(df$cnt, p = .80, list = FALSE)
 train = df[ train.index,]
 test  = df[-train.index,]

  #Decision Tree method


 fit = rpart(cnt ~ ., data = train, method = "anova")
 predictions_DT = predict(fit, test[,-11])
 predictions_DT
 MAPE = function(y, yhat){
   mean(abs((y - yhat)/y)*100)
 }
 regr.eval(test[,11],predictions_DT,stats=c('mse','rmse','mape'))
 MAPE(test[,11],predictions_DT)

 #error rate =22.69%
```

```
#accuracy = 77.30%
#linear regression
library(usdm)

vif(df[,-13])

vifcor(df[,-13], 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[,-11])

#Calculate MAPE
MAPE(test[,11], predictions_LR)
regr.eval(test[,11],predictions_DT,stats=c('mse','rmse','mape'))
#error= 0.18%

#MAE = 635
###Random Forest
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)
#
# #Extract rules
exec = extractRules(treeList, train[,-11])  # R-executable conditions
#
# #Visualize some rules
exec[1:2,]
#
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
# readableRules[1:2,]
#
# #Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-11], train$cnt)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-11],)
```

```
#error metric
regr.eval(test[,11],RF_Predictions,stats=c('mse','rmse','mape'))
 #mape error= 12.7%
MAPE(test[,11],RF_Predictions)

#Model 3 with outliners
##
df=DB
df$casual=NULL
df$registered=NULL
df$instant=NULL
df$dteday=NULL
train.index = createDataPartition(df$cnt, p = .80, list = FALSE)
train = df[ train.index,]
test  = df[-train.index,]
###Random Forest
RF_model = randomForest(cnt ~ ., train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)
#
# #Extract rules
exec = extractRules(treeList, train[,-11])  # R-executable conditions
#
# #Visualize some rules
exec[1:2,]
#
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
# readableRules[1:2,]
#
# #Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-11], train$cnt)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions = predict(RF_model, test[,-11],)
#error metric
regr.eval(test[,11],RF_Predictions,stats=c('mse','rmse','mape'))

MAPE(test[,11],RF_Predictions)
#Mape error =17.76

#case 3 without removing casual and registered but removing cnt
df=df2

df=df[-12]
```

```
df=df[-11]
train.index = createDataPartition(df$cnt, p = .80, list = FALSE)
train = df[ train.index,]
test  = df[-train.index,]
#random forest
RF_model = randomForest(casual ~ ., train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)
#
# #Extract rules
exec = extractRules(treeList, train[,-11])  # R-executable conditions
#
# #Visualize some rules
exec[1:2,]
#
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
# readableRules[1:2,]
#
# #Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-11], train$casual)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions_cas = predict(RF_model, test[,-11],)
#error metric
regr.eval(test[,11],RF_Predictions_cas,stats=c('mse','rmse','mape'))
#mape error= 12.7%
MAPE(test[,11],RF_Predictions_cas)
##registered
df=df2
#df2=df
df$cnt=NULL
df$casual=NULL
train.index = createDataPartition(df$registered, p = .80, list = FALSE)
train = df[ train.index,]
test  = df[-train.index,]
#random forest
RF_model = randomForest(registered ~ ., train, importance = TRUE, ntree = 500)

#Extract rules fromn random forest
#transform rf object to an inTrees' format
treeList = RF2List(RF_model)
#
# #Extract rules
exec = extractRules(treeList, train[,-11])  # R-executable conditions
```

```
#
# #Visualize some rules
exec[1:2,]
#
# #Make rules more readable:
readableRules = presentRules(exec, colnames(train))
# readableRules[1:2,]
#
# #Get rule metrics
ruleMetric = getRuleMetric(exec, train[,-11], train$registered)  # get rule metrics
#
# #evaulate few rules
# ruleMetric[1:2,]

#Presdict test data using random forest model
RF_Predictions_reg = predict(RF_model, test[,-11],)
#error metric
regr.eval(test[,11],RF_Predictions_reg,stats=c('mse','rmse','mape'))
#mape error= 12.7%
MAPE(test[,11],RF_Predictions_cas)
##dropping this case as error rate is of previous case is better##
```