Predication of bike rental count on daily based on the environmental and seasonal settings

ABISHEK K R

25th October 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

Across the world due to increase in road traffic, pollution, price of fuels, global warming and lack of physical activity and busy life style there is a trend shift towards use of Bi-Cycles for short commutes and travels. Thus Bi-Cycle rental industry needs to Improve along with the demand. Demand for the rental Bi-Cycles depends on numerous factors such as environment, seasonal changes, day of the week.

The users in Bi-Cycle rental industry are of two types casual and registered, industry finds it difficult to decide the number of bi-cycles that should be allotted to users everyday depending on factors such as environment, seasonal changes, day of the week. If the industry is not ready with approximate number of bi-cycles when needed they would lose their users. There is need for predicting the number of bi-cycles with the help of modern technique, so the industry could be future ready and be at a competitive edge.

1.2 Data

Our task is to build a regression(prediction) models which will predict the number of bicycles the users will use depending on various environmental and seasonal factors daily.

Given below is the sample of dataset:

instant	dteday	season	yr	mnth	holiday	weekday	workingday	weathersit
1	01-01-2011	1	0	1	0	6	0	2
2	02-01-2011	1	0	1	0	0	0	2
3	03-01-2011	1	0	1	0	1	1	1
4	04-01-2011	1	0	1	0	2	1	1
5	05-01-2011	1	0	1	0	3	1	1
6	06-01-2011	1	0	1	0	4	1	1

temp	atemp	hum	windspee	casual	registerec	cnt
0.344167	0.363625	0.805833	0.160446	331	654	985
0.363478	0.353739	0.696087	0.248539	131	670	801
0.196364	0.189405	0.437273	0.248309	120	1229	1349
0.2	0.212122	0.590435	0.160296	108	1454	1562
0.226957	0.22927	0.436957	0.1869	82	1518	1600

Target variables:

1) Casual.
2) Registered.
3) Count.

Predictor Variables:

1) instant: Record index
2) dteday – date of the observation.
3) season – seasons (1: springer, 2: summer, 3: fall, 4: winter).
4) yr - Year (0: 2011, 1:2012).
5) mnth - Month (1 to 12).
6) holiday - weather day is holiday or not (extracted from Holiday Schedule).
7) weekday - Day of the week.
8) workingday - If day is neither weekend nor holiday is 1, otherwise is 0.
9) weathersit - (extracted from Freemeteo)
   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.
10) 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).
11) atemp - Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max- t_min), t_min=-16, t_max=+50 (only in hourly scale).
12) hum - Normalized humidity. The values are divided to 100 (max).
13) windspeed - Normalized wind speed. The values are divided to 67 (max).

Chapter 2

Methodology

2.1 Pre-Processing

Before building any model, we would be exploring the data and finding if there are any missing values, outliers present in them and process them accordingly. In this day dataset there are no missing values present.

We are not performing outlier analysis as the continuous variables are environmental factors and vary from season to season, altering them would delete some important information. Continuous variables have been normalised already.

The values for the variables temp, atemp, hum, windspeed has already been normalized and we can use these values for further model building.

Chapter 3

Exploratory Data Analysis

3.1 Yearly Trend

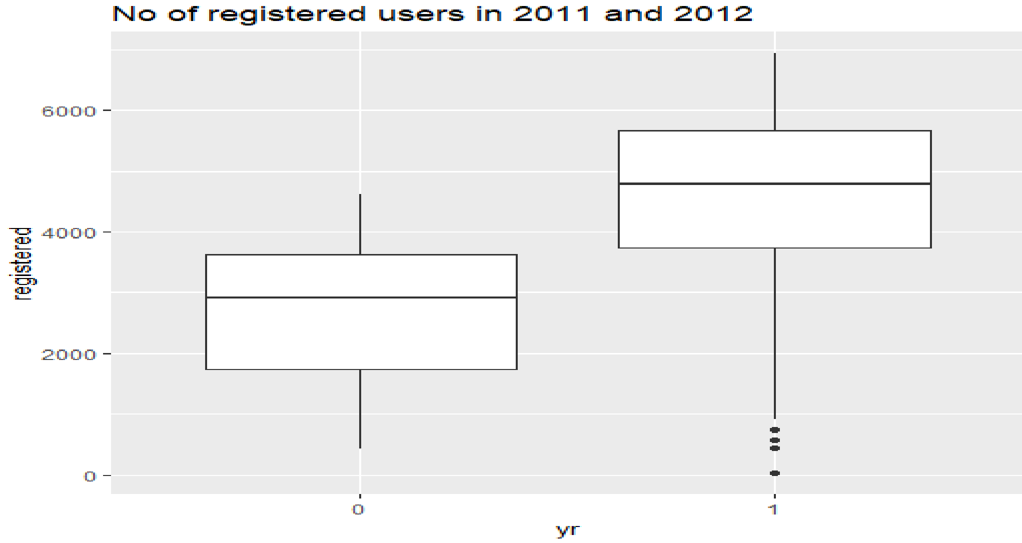
We can see that there has been a significant increase in the number of users in both registered and casual from 2011 to 2012, this is shown using boxplot and summary of the data:

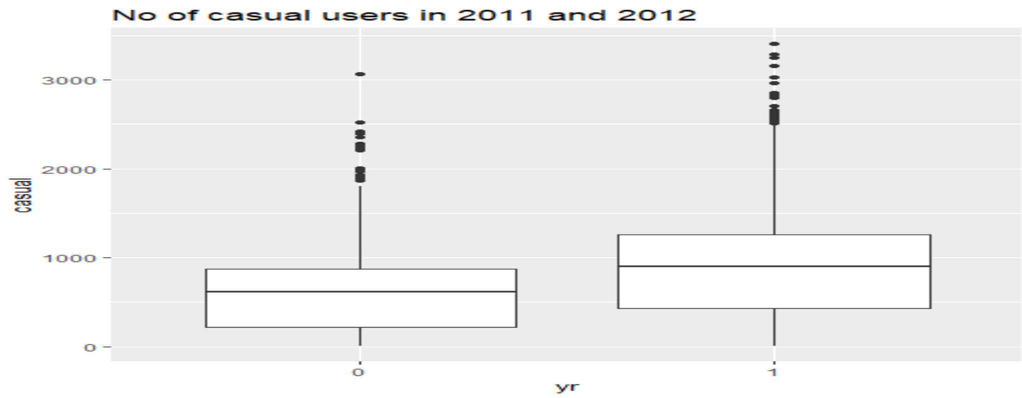
Year 2011:

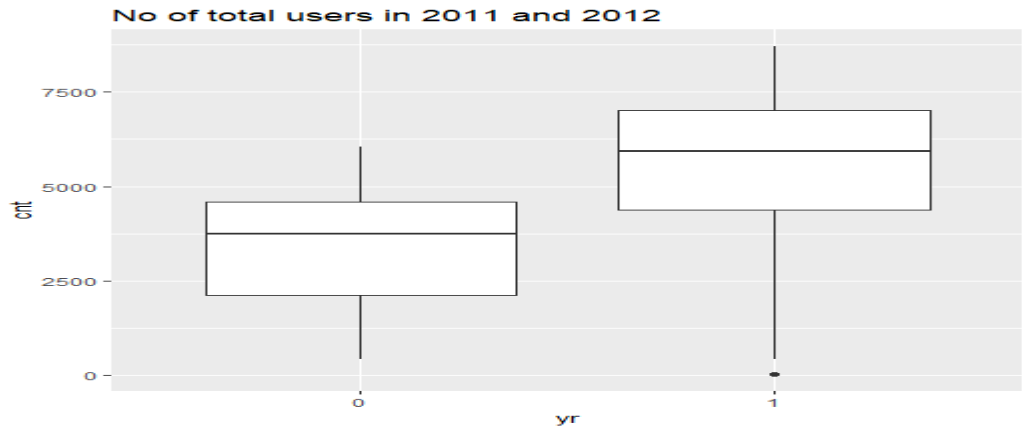
```
yr          casual          registered        cnt
 0:365   Min.   :   9.0   Min.   : 416   Min.   : 431
 1:  0   1st Qu.: 222.0   1st Qu.:1730   1st Qu.:2132
         Median : 614.0   Median :2915   Median :3740
         Mean   : 677.4   Mean   :2728   Mean   :3406
         3rd Qu.: 871.0   3rd Qu.:3632   3rd Qu.:4586
         Max.   :3065.0   Max.   :4614   Max.   :6043
```

Year 2012:

```
yr          casual          registered        cnt
 0:  0   Min.   :   2.0   Min.   :  20   Min.   :  22
 1:366   1st Qu.: 429.8   1st Qu.:3730   1st Qu.:4369
         Median : 904.5   Median :4776   Median :5927
         Mean   :1018.5   Mean   :4581   Mean   :5600
         3rd Qu.:1262.0   3rd Qu.:5663   3rd Qu.:7011
         Max.   :3410.0   Max.   :6946   Max.   :8714
```

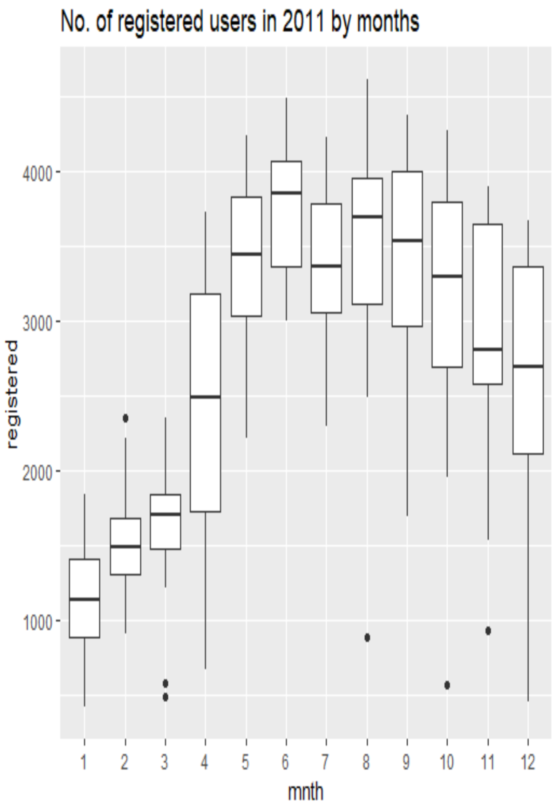

No of registered users in 2011 and 2012
6000
4000
2000
0
registered
0
1
yr

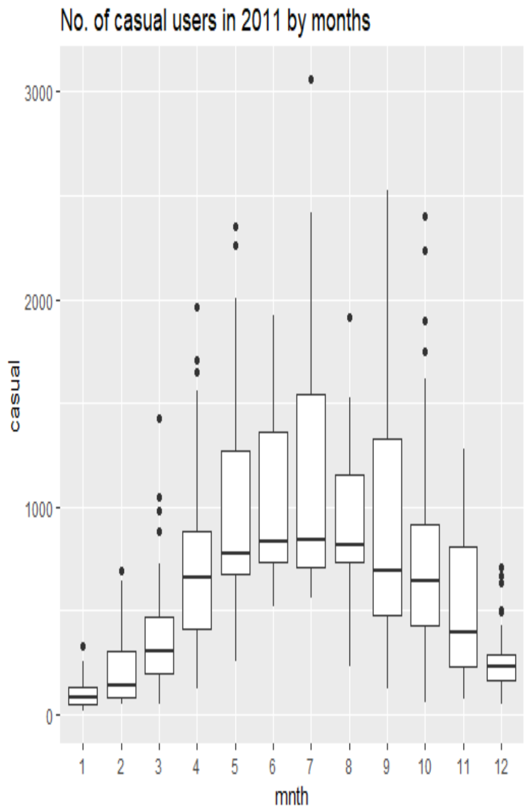

No of casual users in 2011 and 2012
3000
2000
1000
0
casual
0
1
yr


No of total users in 2011 and 2012
7500
5000
2500
0
cnt
0
1
yr

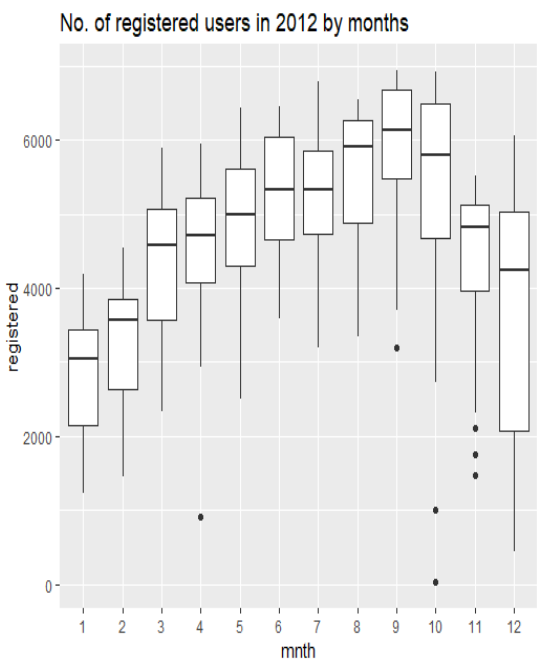
3.2 Monthly Trend

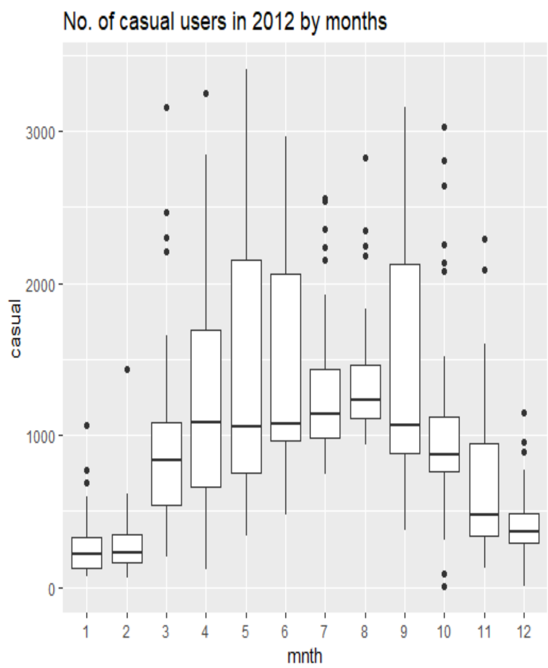
We could see that in the year 2011 number of registered users were maximum in August(8) while casual users were maximum in July(7).





Similarly, we could consider 2012 registered users were maximum in the month of September (9) while casual users were maximum in the month of May(5).

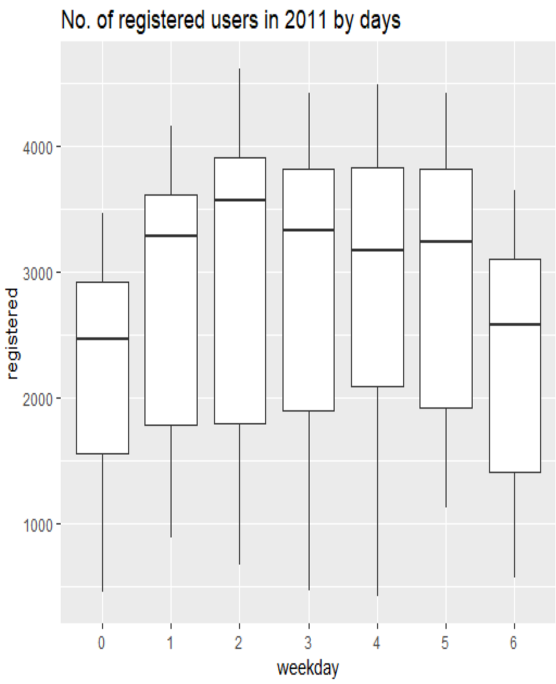


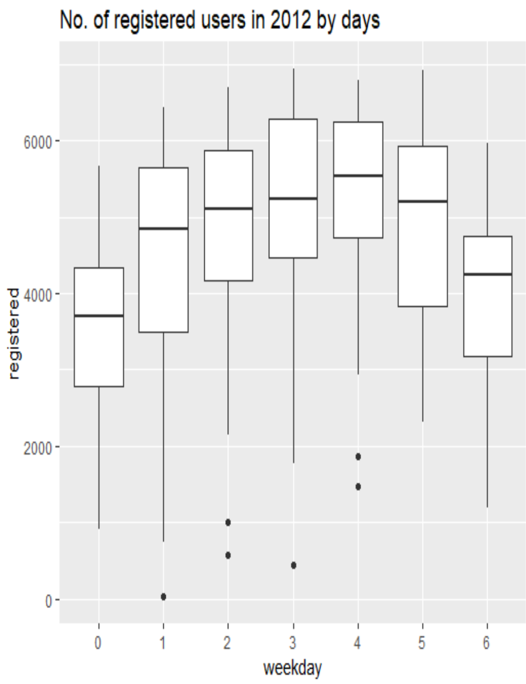


3.3 Daily Trend

We are given with data of 7 days of a week with numbers/labels from 0 to 6, we know that 0 and 6 are weekends may be Saturday and Sunday with holiday. The trend is that registered users are mostly using bikes in weekdays for their daily work commutes.

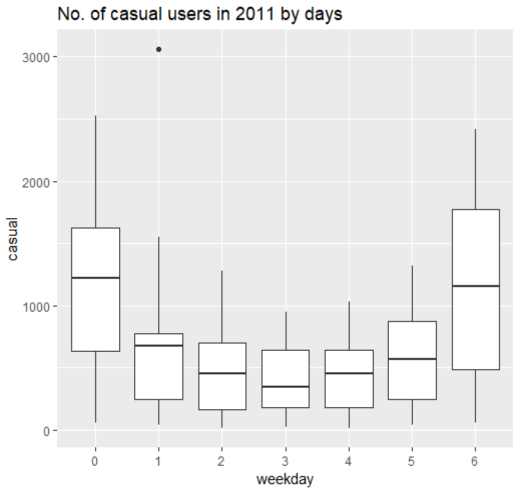
Registered Users:

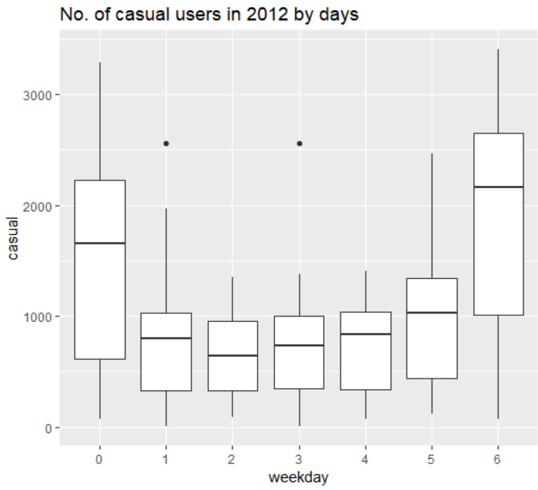




Casual Users:

Casual users seem to use rental bi-cycles mostly on their holidays or weekends:





3.4 Weathersit Trend

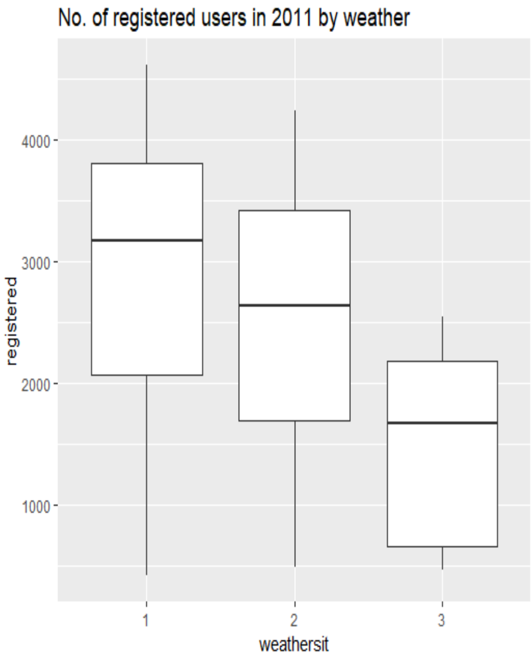
As per the data we have weathersit variable with 3 different factors:

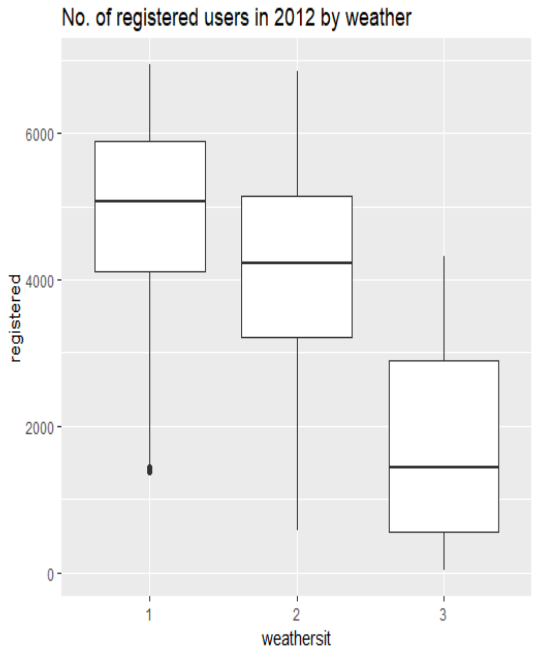
1 : Clear, Few clouds, Partly cloudy, Partly cloudy.

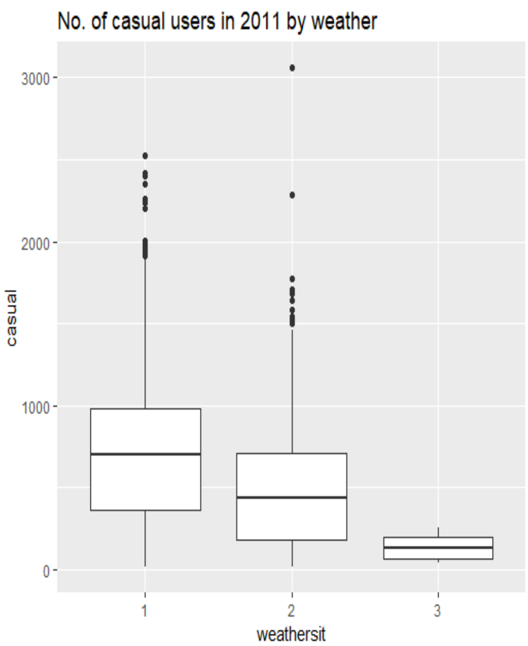
2 : Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist.

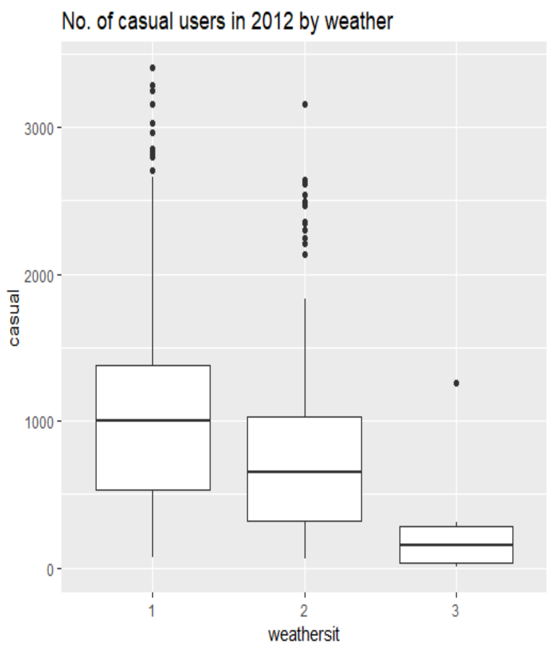
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds.

We can see that users both registered and casual prefer to rent a bicycle mostly in clear clouds and mist, i.e,1 and 2, in both the years 2011 and 2012;



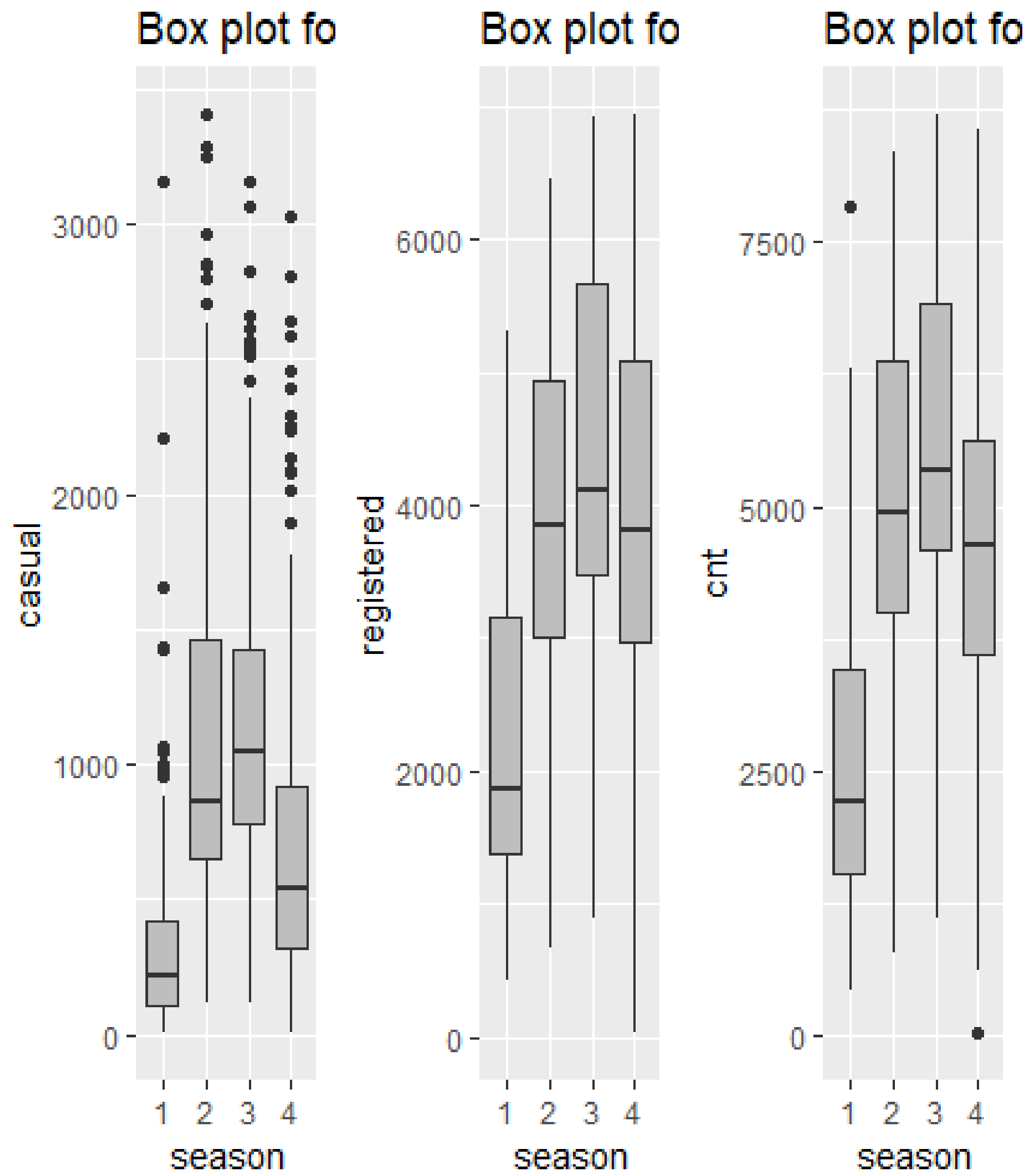






3.5 Distribution of users in different seasons:

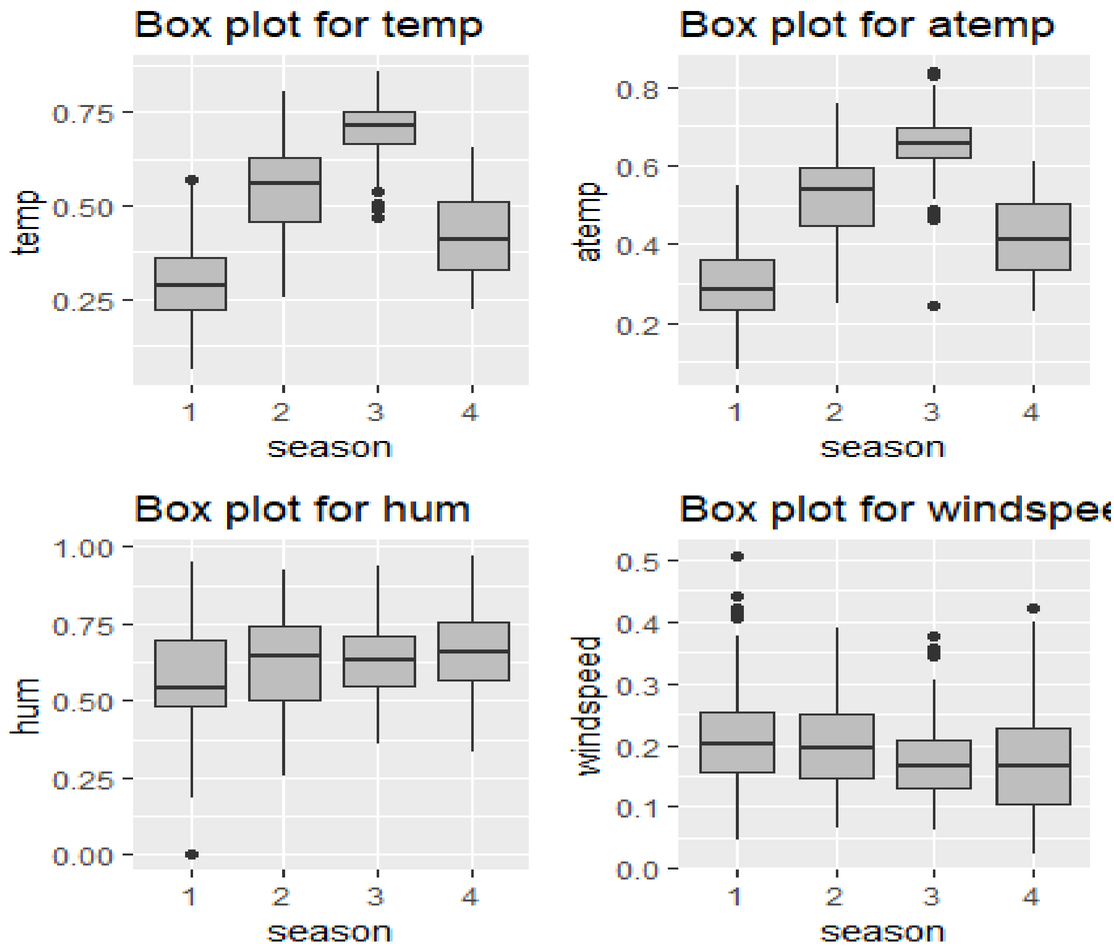
Let's see how the users vary according to seasons:



Users in all the sections prefer to mostly ride in season 2(summer), 3(fall) and 4(winter) least in 1 (spring)

3.6 Distribution of environmental factors in different seasons:

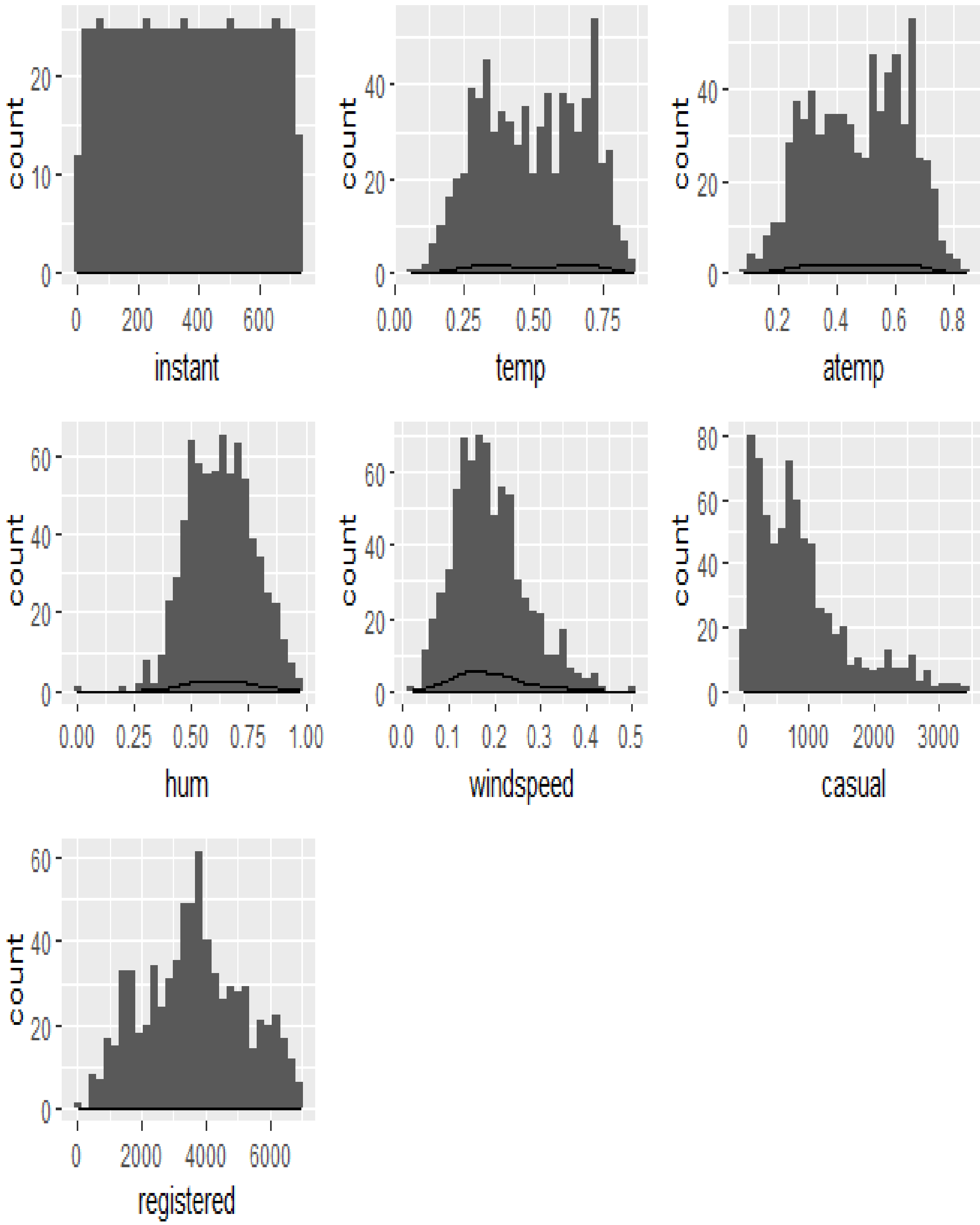
We can explore the distribution of different environmental factors as per seasons:



- From the plot we can conclude that season 2 and season 3 has highest temperature and the users are also high in season 2 and 3.
- Like temp, atemp (feeling temperature) is also high in season 2 and season 3 and users are also high in season 2 and 3.
- hum (Humidity) doesn't vary much between different seasons.
- When we look at windspeed we could conclude that seasons 1 and 4 has high windspeed while those seasons have lesser users.
- We can conclude that when temp and atemp are high users are also high, while when windspeed is high users are less.

3.7 Distribution of all numeric variables:

Let's plot and see how all the numeric variables are distributed using a histogram.



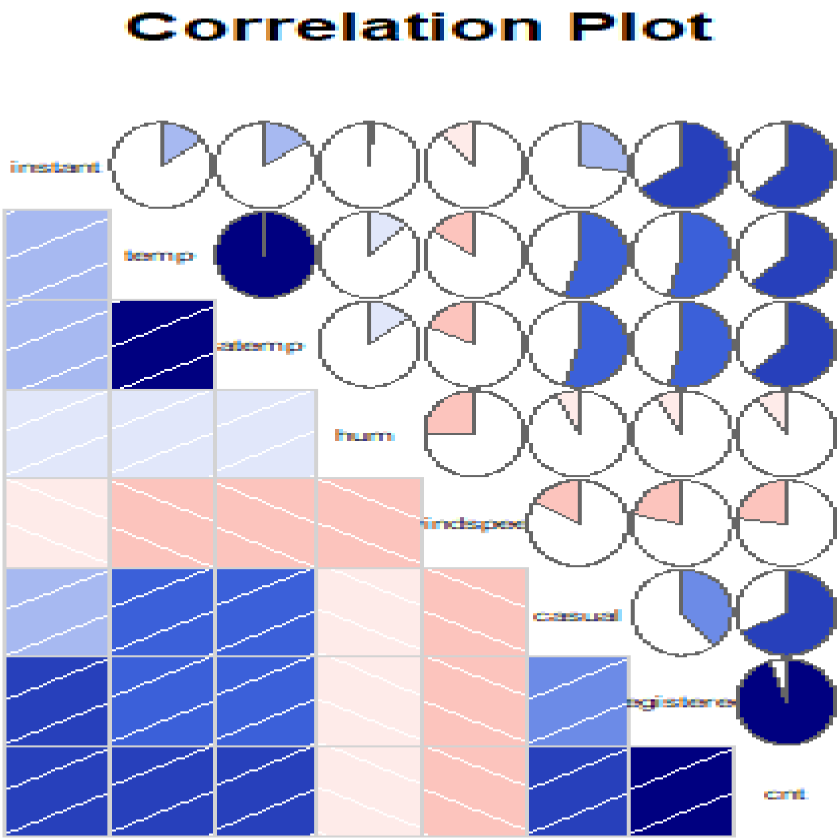
Chapter 4

Feature Selection

While creating a model we can't include all the variables present in the data, that will also contain some variables that carries duplicate information or similar information to another variable. Thus, feature selection is most important step in the modelling.

4.1 Correlation Analysis

While creating a machine learning model we assume that all the independent(predictor) variables are not dependent or doesn't have or can have little correlation between them, while correlation between independent and dependent variables can be present. Thus, it is mandatory to check for correlation before creating a model. We can plot correlation plot and check for correlation:



4.2 Dummy Variables

We have categorical variables with more than 2 categories in our data such as season, mnth, weathersit, weekday, workingday. All these categorical variables are Nominal (it does not have any natural ordering within them) when we create machine learning models as it is, model would assume that categorical variables contain natural ordering. Thus, to avoid this we will create dummy variables (binary variables) and delete the original categorical variables and (n-1) variable to avoid duplication. We will also remove instant and dteday variable as instant is an index variable and dteday information is already present in weekday, holiday, month, year variables.

After creating dummy variables and deleting original categorical variables in our data, we will get a total of 31 variables:

```
> names(dummy_var)
 [1] "yr"            "holiday"       "workingday"   "temp"          "hum"
"windspeed"     "casual"
 [8] "registered"    "cnt"           "season_1"     "season_2"      "season_3
"      "mnth_1"        "mnth_2"
[15] "mnth_3"        "mnth_4"        "mnth_5"       "mnth_6"        "mnth_7"
"mnth_8"        "mnth_9"
[22] "mnth_10"       "mnth_11"       "weathersit_2" "weathersit_1" "weekday_
0"     "weekday_1"     "weekday_2"
[29] "weekday_3"     "weekday_4"     "weekday_5"
```

Chapter 5

Modelling

There are 3 target variables registered, casual and cnt (count), where count is the sum of both registered and casual users. This is a regression problem and we should use regression models in predicting. We can fit and predict for both registered and casual users separately and add them both to arrive at cnt (total count):

5.1 Multiple Linear Regression

Let’s apply multiple linear regression on data with registered as target variable once and casual as target variable once and add both to arrive at cnt.

5.1.1 Registered as Target variable

```
> summary(regress)

Call:
lm(formula = registered ~ yr + holiday + workingday + temp +
    hum + windspeed + season_1 + season_2 + season_3 + mnth_1 +
    mnth_2 + mnth_3 + mnth_4 + mnth_5 + mnth_6 + mnth_7 + mnth_8 +
    mnth_9 + mnth_10 + mnth_11 + weathersit_1 + weathersit_2 +
    weekday_0 + weekday_1 + weekday_2 + weekday_3 + weekday_4 +
    weekday_5, data = dummy_var)

Residuals:
    Min      1Q  Median      3Q     Max
-3671.2  -274.2    69.8   366.4  1408.0

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    894.01     283.29   3.156 0.001669 **
yr1           1732.88      46.08  37.608  < 2e-16 ***
holiday1      -356.24     163.56  -2.178 0.029737 *
workingday1    757.67      84.64   8.951  < 2e-16 ***
temp          2811.84     325.92   8.627  < 2e-16 ***
hum           -983.80     231.25  -4.254 2.38e-05 ***
windspeed    -1816.95     321.44  -5.653 2.30e-08 ***
season_1     -1532.58     143.27 -10.697  < 2e-16 ***
season_2      -859.59     168.06  -5.115 4.05e-07 ***
season_3      -764.39     151.48  -5.046 5.75e-07 ***
mnth_1          92.46     144.21   0.641 0.521643
mnth_2         246.03     145.25   1.694 0.090746 .
mnth_3         375.92     146.53   2.565 0.010509 *
mnth_4         294.39     191.66   1.536 0.124996
mnth_5         549.61     203.96   2.695 0.007215 **
mnth_6         470.52     207.78   2.265 0.023843 *
mnth_7          23.61     221.11   0.107 0.914989
mnth_8         343.85     211.71   1.624 0.104792
mnth_9         748.50     172.81   4.331 1.70e-05 ***
mnth_10        239.98     129.42   1.854 0.064126 .
mnth_11       -180.28     122.54  -1.471 0.141689
weathersit_1  1662.51     155.64  10.682  < 2e-16 ***
weathersit_2  1301.69     145.80   8.928  < 2e-16 ***
weekday_0     -289.20      84.35  -3.428 0.000642 ***
weekday_1      -92.95      86.40  -1.076 0.282403
weekday_2       58.41      84.88   0.688 0.491579
weekday_3      131.49      85.25   1.542 0.123443
weekday_4      129.42      84.72   1.528 0.127078
weekday_5          NA         NA      NA       NA
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 609 on 703 degrees of freedom
Multiple R-squared:  0.8533,	Adjusted R-squared:  0.8477
F-statistic: 151.4 on 27 and 703 DF,  p-value: < 2.2e-16
```

We have arrived at R-squared : 0.8533 and Adjusted R-squared : 0.8477 which is quite near to 1, this model explains 84% of the target variable. Thus, let’s see how this model will predict the test data.

5.1.2 Casual as Target variable:

```
> summary(casu)

Call:
lm(formula = casual ~ yr + holiday + workingday + temp + hum + 
    windspeed + season_1 + season_2 + season_3 + mnth_1 + mnth_2 + 
    mnth_3 + mnth_4 + mnth_5 + mnth_6 + mnth_7 + mnth_8 + mnth_9 + 
    mnth_10 + mnth_11 + weathersit_1 + weathersit_2 + weekday_0 + 
    weekday_1 + weekday_2 + weekday_3 + weekday_4 + weekday_5, 
    data = dummy_var)

Residuals:
     Min       1Q   Median       3Q      Max 
-1030.25  -204.77   -14.15   162.96  1754.17 

Coefficients: (1 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)    601.44     164.37   3.659 0.000272 ***
yr1            285.18      26.74  10.667  < 2e-16 ***
holiday1      -257.46      94.91  -2.713 0.006835 ** 
workingday1   -767.77      49.11 -15.633  < 2e-16 ***
temp          1675.47     189.11   8.860  < 2e-16 ***
hum           -534.38     134.18  -3.983 7.52e-05 ***
windspeed    -1108.49     186.51  -5.943 4.40e-09 ***
season_1       -46.37      83.13  -0.558 0.577160    
season_2       169.94      97.51   1.743 0.081806 .  
season_3        17.68      87.89   0.201 0.840683    
mnth_1          -8.07      83.68  -0.096 0.923193    
mnth_2         -24.78      84.28  -0.294 0.768796    
mnth_3         253.60      85.02   2.983 0.002955 ** 
mnth_4         246.50     111.21   2.217 0.026976 *  
mnth_5         258.30     118.35   2.183 0.029397 *  
mnth_6         104.42     120.56   0.866 0.386702    
mnth_7          69.18     128.29   0.539 0.589882    
mnth_8         145.45     122.84   1.184 0.236814    
mnth_9         319.84     100.27   3.190 0.001487 ** 
mnth_10        365.35      75.10   4.865 1.41e-06 ***
mnth_11        153.31      71.10   2.156 0.031409 *  
weathersit_1   318.85      90.31   3.531 0.000442 ***
weathersit_2   214.47      84.60   2.535 0.011455 *  
weekday_0     -149.50      48.94  -3.054 0.002340 ** 
weekday_1     -120.78      50.13  -2.409 0.016251 *  
weekday_2     -177.88      49.25  -3.612 0.000326 ***
weekday_3     -182.69      49.47  -3.693 0.000239 ***
weekday_4     -172.82      49.16  -3.515 0.000467 ***
weekday_5          NA         NA      NA       NA    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 353.3 on 703 degrees of freedom
Multiple R-squared:  0.745,	Adjusted R-squared:  0.7352 
F-statistic: 76.05 on 27 and 703 DF,  p-value: < 2.2e-16
```

We have arrived at a R-squared value of 0.745 and Adjusted R-square of 0.7352, which indicated the model could explain 73% of the target variable. p-value is also acceptable and thus rejects null hypothesis. Let’s see how the model would predict when applied with test data.

5.2 Random Forest

Random Forest is the combination of number of decision trees, like linear regression we can apply random forest on registered as target variable once and casual as target variable once and add them both to arrive at total count.

5.2.1 Registered as Target Variable:

```
> regress <- randomForest(registered ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
+                         mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
+                         weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, importance = TRUE, ntrees = 500)
```

We can look at the top rules using the following procedure in R;

```
> treelist_reg <- RF2List(regress)
> exec_reg <- extractRules(treelist_reg, y)
4761 rules (length<=6) were extracted from the first 100 trees.
```

There are about 4761 rules being created from first 100 trees;

Let's have a look at the top 5 rules obtained:

```
> readablerules_reg[1:5]
[1] "yr %in% c('0') & temp<=0.455 & hum<=0.89875 & season_1<=0.5 & season_2<=0.5 & mnth_11<=0.5"
[2] "yr %in% c('0') & temp<=0.455 & hum>0.89875 & season_1<=0.5 & season_2<=0.5 & mnth_11<=0.5"
[3] "yr %in% c('0') & workingday %in% c('0') & temp<=0.455 & season_1<=0.5 & season_2>0.5 & mnth_11<=0.5"
[4] "yr %in% c('0') & workingday %in% c('1') & temp<=0.455 & season_1<=0.5 & season_2>0.5 & mnth_11<=0.5"
[5] "yr %in% c('0') & holiday %in% c('1') & temp<=0.455 & season_1>0.5 & mnth_11<=0.5"
```

5.2.2 Casual as Target variable

```
> casu <- randomForest(casual ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
+                 mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
+                 weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, importance = TRUE, ntrees = 500)
```

We can look at the top rules using the following procedure in R;

```
> treelist_cas <- RF2List(casu)
> exec_casu <- extractRules(treelist_cas, y)
4817 rules (length<=6) were extracted from the first 100 trees.
```

There are about 4817 rules being created from first 100 trees;

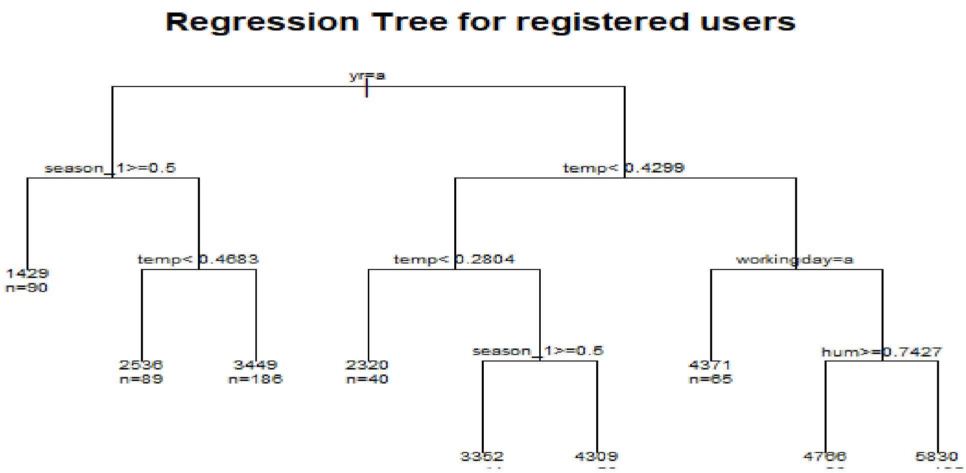
```
> readablerules_cas[1:5]
[1] "yr %in% c('1') & hum<=0.4883335 & season_1<=0.5 & season_2<=0.5 & sea
son_3<=0.5 & mnth_11<=0.5"
[2] "yr %in% c('0') & hum<=0.4883335 & season_1<=0.5 & season_2<=0.5 & sea
son_3<=0.5 & mnth_11<=0.5"
[3] "hum<=0.4883335 & windspeed<=0.298198 & season_1<=0.5 & season_2<=0.5
& season_3<=0.5 & mnth_11>0.5"
[4] "hum<=0.4883335 & windspeed>0.298198 & season_1<=0.5 & season_2<=0.5 &
season_3<=0.5 & mnth_11>0.5"
[5] "workingday %in% c('1') & temp<=0.415833 & hum>0.4883335 & season_1<=0
.5 & season_2<=0.5 & season_3<=0.5"
```

5.3 Decision Tree

Let’s use Decision tree with argument method = ‘anova’ for regression.

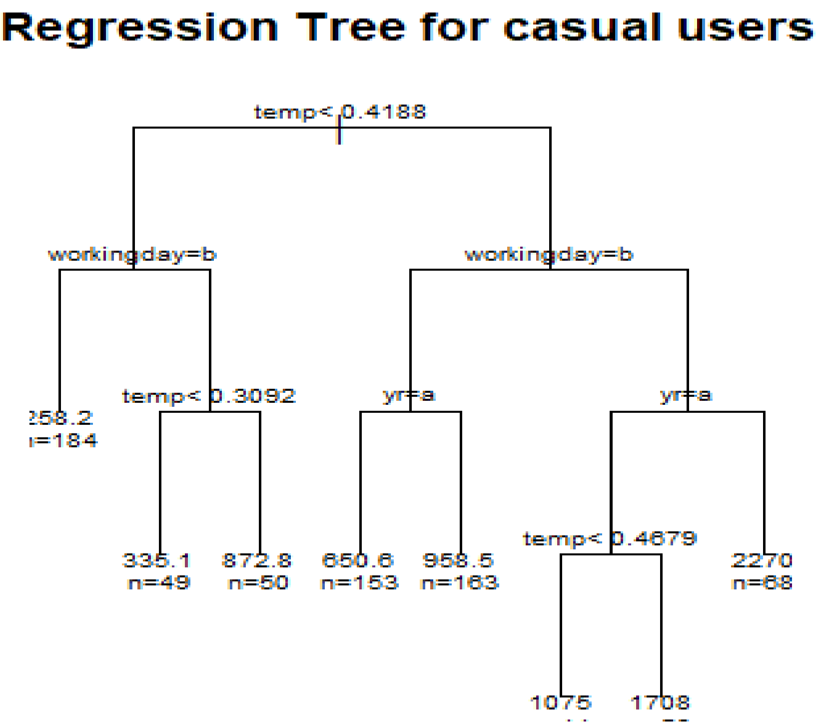
5.3.1 Registered as Target variable

```
> regress <- rpart(registered ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
+                 mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
+                 weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, method = 'anova')
```



5.3.2 Casual as Target variable

```
> casu <- rpart(casual ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
+               mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
+               weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, method = 'anova')
```



Chapter 6

Model Evaluation

We are dealing with a regression problem and we can use MAPE (Mean Absolute Percentage Error) as error metric to select the best model:

MAPE = mean (abs ((Y - Yhat)/Y))*100

Where Y is actual value and Yhat is the predicted value.

Lower the value of MAPE higher the accuracy percentage of the model.

As the problem statement is to predict the total count we will be adding both the prediction of registered with the prediction of casual users and arrive at predictions of total count.

6.1 Multiple Linear Regression

```
> MAPE = function(Y, Yhat){
+   mean(abs((Y - Yhat)/Y))*100
+ }
> MAPE(y_test$cnt, y$count)
[1] 17.5405
```

We have a MAPE value of 17.5405 which indicates that our model can predict 82.3% nearer to actual value, let’s explore what other models can predict. (PYTHON MAPE = 14.2173)

6.2 Random Forest

```
> MAPE = function(Y, Yhat){
+   mean(abs((Y - Yhat)/Y))*100
+ }
> MAPE(y_test$cnt, y$count)
[1] 8.247438
```

We have a MAPE value of 8.2474 which indicates that our model can predict 91% nearer to actual value, let’s explore what other models can predict. (PYTHON MAPE = 11.1503)

6.3 Decision Tree

```
> MAPE = function(Y, Yhat){
+   mean(abs((Y - Yhat)/Y))*100
+ }
> MAPE(y_test$cnt, y$count)
[1] 20.06097
```

We have a MAPE value of 20.0609 which indicates that our model can predict 79% nearer to actual value. (PYTHON MAPE = 14.4723)

Chapter 7

Conclusion

7.1 Model Selection

We can conclude that RandomForest has performed well than the other two regression models with around Mean Absolute Percentage Error of 8.24% in R and 11.15% in Python.

APPENDIX A

R code:

```
rm(list = ls())

library(ggplot2)

library(dplyr)

library(corrgram)

library(randomForest)

library(fastDummies)

#install.packages('inTrees')

library(inTrees)

library(rpart)

library(DMwR)

getwd()

setwd('C:\\Users\\abish\\Desktop\\edwisor\\projects')

df<- read.csv('day.csv')

#View(df)

names(df)

str(df)

##############################################################################
```

#we are converting this int type to factors so they can be suitable for EDA

#we know that variables are already categorical and they are labelled accordingly as 0,1,2,...convert them from int to factors:

```
categorical_variables <- c('season','yr','mnth','holiday','weekday','workingday','weathersit')

df[categorical_variables] <- lapply(df[categorical_variables],factor)

str(df)

##########################################################################

#lets looking fot missing values

#we will look for outliers in the dataset:

missing <- data.frame(apply(df,2,function(df){sum(is.na(df))}))

missing

##########################################################################

#lets see the distribution of all the numerical variables according to the seasons:

numerical_variables <- sapply(df, is.numeric)

num_data <- df[numerical_variables]

num_cols <- colnames(num_data)

#looking for outliers in boxplots:#

for (i in 1:length(num_cols))

{

 assign(paste0("gn",i), ggplot(aes_string(y = (num_cols[i]), x = 'season'), data = subset(df))+

      stat_boxplot() +

      geom_boxplot(fill = "grey") +

      theme(legend.position="season")+

      labs(y=num_cols[i],x="season")+

      ggtitle(paste("Box plot for",num_cols[i])))

}



# ## Plotting plots together

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

gridExtra::grid.arrange(gn2,gn3,gn4,gn5,ncol=2, nrow = 2)

gridExtra::grid.arrange(gn6,gn7,gn8, ncol=3)

####################################################################
```

```
################exploratory Data Analysis#########################

#checking/comparing the number of bikes rented in years 2011 and 2012:

years_registered <- df %>% group_by(yr) %>%
  ggplot(aes(x = yr, y = registered)) +
  geom_boxplot() +
  ggtitle("No of registered users in 2011 and 2012")
years_registered

years_casual <-   df %>% group_by(yr) %>%
  ggplot(aes(x = yr, y = casual)) +
  geom_boxplot() +
  ggtitle("No of casual users in 2011 and 2012")
years_casual

years_cnt <- df %>% group_by(yr) %>%
  ggplot(aes(x = yr, y = cnt)) +
  geom_boxplot() +
  ggtitle("No of total users in 2011 and 2012")
years_cnt
####################################################################
year_2011 <- df %>% filter(yr == 0) %>% select(c("yr","casual","registered","cnt"))
year_2012 <- df %>% filter(yr == 1) %>% select(c("yr","casual","registered","cnt"))
yr_2011 <- summary(year_2011)
yr_2012 <- summary(year_2012)
yr_2011
yr_2012
####################################################################
#we could check for the monthly trend of users
months_2011 <- df %>% filter(yr == 0) %>%
  ggplot(aes(mnth,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2011 by months")
months_2011
months_2012 <- df %>% filter(yr == 1) %>%
  ggplot(aes(mnth,casual)) +
```

```
  geom_boxplot() +
  ggtitle("No. of registered users in 2012 by months")
months_2012
months_2011_casual <- df %>% filter(yr == 0) %>%
  ggplot(aes(mnth,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2011 by months")
months_2011_casual
months_2012_casual <- df %>% filter(yr == 1) %>%
  ggplot(aes(mnth,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2012 by months")
months_2012_casual
################################################################################
#checking for daily trends:
days_2011_registered <- df %>% filter(yr == 0) %>%
  ggplot(aes(weekday,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2011 by days")
days_2011_registered
days_2012_registered <- df %>% filter(yr == 1) %>%
  ggplot(aes(weekday,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2012 by days")
days_2012_registered
days_2011_casual <- df %>% filter(yr == 0) %>% 9
  ggplot(aes(weekday,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2011 by days")
days_2011_casual
days_2012_casual <- df %>% filter(yr == 1) %>%
  ggplot(aes(weekday,casual)) +
  geom_boxplot() +
```

```
  ggtitle("No. of casual users in 2012 by days")
days_2012_casual/
############################################################################
#Holiday_trend
holiday_2011_registered <- df %>% filter(yr == 0) %>%
  ggplot(aes(holiday,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2011 by holiday")
holiday_2011_registered
holiday_2012_registered <- df %>% filter(yr == 1) %>%
  ggplot(aes(holiday,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2012 by holiday")
holiday_2012_registered
holiday_2011_casual <- df %>% filter(yr == 0) %>%
  ggplot(aes(holiday,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2011 by holiday")
holiday_2011_casual
holiday_2012_casual <- df %>% filter(yr == 1) %>%
  ggplot(aes(holiday,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2012 by holiday")
holiday_2012_casual
############################################################################
# weathersit:
weathersit_2011_registered <- df %>% filter(yr == 0) %>%
  ggplot(aes(weathersit,registered)) +
  geom_boxplot() +
  ggtitle("No. of registered users in 2011 by weather")
weathersit_2011_registered
weathersit_2012_registered <- df %>% filter(yr == 1) %>%
  ggplot(aes(weathersit,registered)) +
```

```
  geom_boxplot() +
  ggtitle("No. of registered users in 2012 by weather")
weathersit_2012_registered
weathersit_2011_casual <- df %>% filter(yr == 0) %>%
  ggplot(aes(weathersit,casual)) +
  geom_boxplot() +
  ggtitle("No. of casual users in 2011 by weather")
weathersit_2011_casual
##############################################################################
#Distribution between temperature and season:
temp_2011_season <- df %>% filter(yr == 0) %>%
  ggplot(aes(season,temp)) +
  geom_boxplot() +
  ggtitle("Temperature distribution in different seasons")
temp_2011_season
temp_2012_season <- df %>% filter(yr == 0) %>%
  ggplot(aes(season,temp)) +
  geom_boxplot() +
  ggtitle("Temperature distribution in different seasons")
temp_2012_season
##############################################################################
#looking for the distribution of atemp in different seasons
atemp_2011_season <- df %>% filter(yr == 0) %>%
  ggplot(aes(season,atemp)) +
  geom_boxplot() +
  ggtitle("Feeling_Temperature distribution in different seasons(2011)")
atemp_2011_season
atemp_2012_season <- df %>% filter(yr == 1) %>%
  ggplot(aes(season,atemp)) +
  geom_boxplot() +
  ggtitle("Feeling_Temperature distribution in different seasons(2012)")
atemp_2012_season
##############################################################################
```

```
#looking for the distribution of windspeed according to the seasons:
windspeed_2011_season <- df %>% filter(yr == 0) %>%
  ggplot(aes(season,windspeed)) +
  geom_boxplot() +
  ggtitle("windspeed distribution in different seasons(2011)")
windspeed_2011_season
windspeed_2012_season <- df %>% filter(yr == 1) %>%
  ggplot(aes(season,windspeed)) +
  geom_boxplot() +
  ggtitle("windspeed distribution in different seasons(2012)")
windspeed_2012_season
###########################################################################
#Distribution of humidity according to seasons:
humidity_2011_season <- df %>% filter(yr == 0) %>%
  ggplot(aes(season,hum)) +
  geom_boxplot() +
  ggtitle("humidity distribution in different seasons(2011)")
humidity_2011_season
humidity_2012_season <- df %>% filter(yr == 1) %>%
  ggplot(aes(season,hum)) +
  geom_boxplot() +
  ggtitle("humidity distribution in different seasons(2012)")
humidity_2012_season
#########################################################################
#########looking for the distribution of all the continious variables##################
for (i in 1:ncol(df)){
  print(i)
  if (class(df[,i]) == 'integer' | class(df[,i]) == 'numeric'){
    print(i)
    assign(paste0("g",i), ggplot(aes_string(x = names(df)[i]), data = subset(df))+
             geom_histogram()+
             geom_density())
  }
```

```
}
gridExtra::grid.arrange(g1,g10,g11,g12,g13,g14,g15, nrow = 3, ncol = 3)
###############################################################################
#correlTION ANALYSIS:
numeric_index <- sapply(df, is.numeric)
## Correlation Plot
corrgram(df[,numeric_index], order = F,
        upper.panel=panel.pie, text.panel=panel.txt, main = "Correlation Plot")
###############################################################################
#as per correlation analysis there is a high correlation between temp and atemp
#we will be removing atemp from our further modelling
###############################################################################
df <- df %>% select(-c('atemp'))
names(df)
###############################################################################
#lets try to convert the categorical variables with more than two factors to dummies and try applying machine learning model
#so lets improve the accuracy of model:
dummy_df <- dummy_columns(df, select_columns = c('season','mnth','weathersit','weekday'))
dummy_var <- dummy_df %>% select(-c("season","mnth","weathersit","weekday"))
names(dummy_var)
###############################################################################
#we can remove any one of the dummy variable of every category to avoid multicollinearity and high correlation
dummy_var <- dummy_var %>% select(-c('season_4','mnth_12','weathersit_3','weekday_6'))
names(dummy_var)
###############################################################################
#we can remove instant and dteday variables as they are not useful for the model we havwe enough information from them;
dummy_var <- dummy_var %>% select(-c('instant','dteday'))
###############################################################################
#Train Test Split (sampling - simple random sampling)
set.seed(121)
```

```
index = sample(nrow(dummy_var), 0.80 * nrow(dummy_var), replace = F)

x = dummy_var[index,]

y_test = dummy_var[-(index),]

y <- y_test %>% select(-c( "casual","registered","cnt"))

###############################################################################

#regress <- lm(registered ~
yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+m
nth_4+

#
mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+we
ekday_1+

#                    weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var)

#casu <- lm(casual ~
yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+m
nth_4+

#
mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+we
ekday_1+

#                  weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var)

###############################################################################

#pred_regress <- predict(regress, y)

#pred_regress <- as.data.frame(pred_regress)

#pred_casu <- predict(casu, y)

#pred_casu <- as.data.frame(pred_casu)

#y['count'] <- pred_regress + pred_casu

#y$count <- as.integer(y$count)

###############################################################################

#regr.eval(y_test$cnt, y$count, stats =  c('mae','rmse','mape','mse'))

#       mae        rmse        mape        mse

#5.651497e+02 7.885029e+02 1.754050e-01 6.217369e+05

###############################################################################

#MAPE = function(Y, Yhat){

#  mean(abs((Y - Yhat)/Y))*100

#}

#MAPE(y_test$cnt, y$count)

#17.5405
```

```
###############################################################################

#RandomForest:

regress <- randomForest(registered ~
yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+

mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+

            weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, importance = TRUE, ntrees = 250)

casu <- randomForest(casual ~
yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+

mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+

            weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, importance = TRUE, ntrees = 250)

############################################################################

treelist_reg <- RF2List(regress)

treelist_cas <- RF2List(casu)

exec_reg <- extractRules(treelist_reg, y)

exec_casu <- extractRules(treelist_cas, y)

readablerules_reg <- presentRules(exec_reg, colnames(y))

readablerules_cas <- presentRules(exec_casu, colnames(y))

###############################################################################

pred_regress <- predict(regress, y)

pred_regress <- as.data.frame(pred_regress)

pred_casu <- predict(casu, y)

pred_casu <- as.data.frame(pred_casu)

y['count'] <- pred_regress + pred_casu

y$count <- as.integer(y$count)

###############################################################################

regr.eval(y_test$cnt, y$count, stats =  c('mae','rmse','mape','mse'))

#mae        rmse        mape        mse

#2.524830e+02 3.423996e+02 8.247438e-02 1.172375e+05

###############################################################################
```

```
MAPE = function(Y, Yhat){
  mean(abs((Y - Yhat)/Y))*100
}
MAPE(y_test$cnt, y$count)
#[1] 8.247438
#this is a way perfect model as the mean absolute percentage error is 8.24 % which indicates that
#our predictions are 91% nearer to the actual values present thus we could consider that Random
#forest with dummy vaiables are perfect....
##########################################################################
#Decision trees:
#regress <- rpart(registered ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
#  mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
#          weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, method = 'anova')
#plot(regress, uniform=TRUE,
#    main="Regression Tree for registered users")
#text(regress, use.n=TRUE, cex = .6)
#casu <- rpart(casual ~ yr+holiday+workingday+temp+hum+windspeed+season_1+season_2+season_3+mnth_1+mnth_2+mnth_3+mnth_4+
#  mnth_5+mnth_6+mnth_7+mnth_8+mnth_9+mnth_10+mnth_11+weathersit_1+weathersit_2+weekday_0+weekday_1+
#         weekday_2+weekday_3+weekday_4+weekday_5, data = dummy_var, method = 'anova')
#plot(casu, uniform=TRUE,
#    main="Regression Tree for casual users")
#text(casu, use.n=TRUE, cex = .6)
##########################################################################
#pred_regress <- predict(regress, y)
#pred_regress <- as.data.frame(pred_regress)
#pred_casu <- predict(casu, y)
#pred_casu <- as.data.frame(pred_casu)
#y['count'] <- pred_regress + pred_casu
```

```
#y$count <- as.integer(y$count)
################################################################################
#regr.eval(y_test$cnt, y$count, stats =  c('mae','rmse','mape','mse'))
#mae        rmse        mape         mse
#6.560748e+02 8.385436e+02 2.006097e-01 7.031553e+05
################################################################################
#MAPE = function(Y, Yhat){
#  mean(abs((Y - Yhat)/Y))*100
#}
#MAPE(y_test$cnt, y$count)
#20.06097
```

PYTHON code:

```
import os
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import pandas as pd
from fancyimpute import KNN
os.chdir("C:\\Users\\abish\\Desktop\\edwisor\\projects")
df = pd.read_csv("day.csv")
df.head()
df.describe()
#converting numerical variables with factors to categorical:
categorical_var = ['season','yr','mnth','holiday','weekday','workingday','weathersit']
for i in categorical_var:
   print(i)
```

```
    df[i] = pd.Categorical(df[i])
df.head()
#check for missing values:
missing = pd.DataFrame(df.isnull().sum())
missing
#there are no missing values
a4_dims = (11.7, 8.27)
f,(ax1,ax2,ax3) = plt.subplots(3,1, figsize = a4_dims)
s = sns.boxplot(x = 'yr', y = 'registered', data = df, ax = ax1)
t = sns.boxplot(x = 'yr', y = 'casual', data = df, ax = ax2)
u = sns.boxplot(x = 'yr', y = 'cnt', data = df, ax = ax3)
s.set(title = 'No of users by Year 2011 and 2012')
#registered users have also increased from 2011 to 2012
#casual users have also increased over the year 2012
#obviously the count has increased in the year 2012
#we could now chwck the distribution of users as per months of both 2011 and 2012:
df_2011 = df[df.yr == 0]
df_2012 = df[df.yr == 1]
a4_dims = (11.7, 8.27)
f, (ax1,ax2,ax3) = plt.subplots(3, figsize = a4_dims)
a = sns.boxplot(x = 'mnth', y = 'registered', data = df_2011, ax = ax1)
b = sns.boxplot(x = 'mnth', y = 'casual', data = df_2011, ax = ax2)
c = sns.boxplot(x = 'mnth', y = 'cnt', data = df_2011, ax = ax3)
a.set(xlabel = 'months in 2011', ylabel = 'registered users in 2011', title = 'Users_2011')
b.set(xlabel = 'months in 2011', ylabel = 'casual users in 2011')
c.set(xlabel = 'months in 2011', ylabel = 'total users in 2011')
#we can conclude that in the year 2011 registered users were high in september and lowest in January.
#for the casual users in the year 2011 were high in may and lowest in december and jan
a4_dims = (11.7, 8.27)
f, (ax1,ax2,ax3) = plt.subplots(3, figsize = a4_dims)
a = sns.boxplot(x = 'mnth', y = 'registered', data = df_2012, ax = ax1)
b = sns.boxplot(x = 'mnth', y = 'casual', data = df_2012, ax = ax2)
c = sns.boxplot(x = 'mnth', y = 'cnt', data = df_2012, ax = ax3)
```

```
a.set(xlabel = 'months in 2012', ylabel = 'registered users in 2012', title = 'Users_2012')

b.set(xlabel = 'months in 2012', ylabel = 'casual users in 2012')

c.set(xlabel = 'months in 2012', ylabel = 'total users in 2012')

#in the year 2012 registered users are high in september, and lowest in jan

#in the year 2012 casual users were high in may, and lowest in jan and dec

#totally the number of users were high in september and lowest in January

#checking for the daily trend weather the no. of users vary on day by day:

f, (ax1,ax2,ax3,ax4) = plt.subplots(4, figsize = a4_dims )

reg_2011_days = sns.boxplot(x = 'weekday', y = 'registered', data = df_2011, ax = ax1)

cas_2011_days = sns.boxplot(x = 'weekday', y = 'casual', data = df_2011, ax = ax2)

reg_2012_days = sns.boxplot(x = 'weekday', y = 'registered', data = df_2012, ax = ax3)

cas_2012_days = sns.boxplot(x = 'weekday', y = 'casual', data = df_2012, ax = ax4)

reg_2011_days.set(ylabel = 'registered_2011', title = 'Distribution of users in days' )

cas_2011_days.set(ylabel = 'casual_2011')

reg_2012_days.set(ylabel = 'registered_2012')

cas_2012_days.set(ylabel = 'casual_2012')

#it is clear that in the years 2011 and 2012 registered users use bi-cycles in weekdays where as casual users
use mostly on their weekends.

#check for the holiday trend wethere users prefer using bicycle in holidays :

f,(ax1,ax2,ax3,ax4) = plt.subplots(4, figsize = (14,14))

registered_holiday_2011 = sns.boxplot(x = 'holiday', y = 'registered', data = df_2011, ax = ax1).set(ylabel =
'registered_2011', title = 'Distribution of users by Holiday')

casual_holiday_2011 = sns.boxplot(x = 'holiday', y = 'casual', data = df_2011, ax = ax2).set(ylabel =
'casual_2011')

registered_holiday_2012 = sns.boxplot(x = 'holiday', y = 'registered', data = df_2012, ax = ax3).set(ylabel =
'registered_2012')

casual_holiday_2011 = sns.boxplot(x = 'holiday', y = 'casual', data = df_2012, ax = ax4).set(ylabel =
'casual_2012')

##########################################################

#In the year 2011 registered users were high at weekdays than holidays, casual users were also significantly
high on non holidays.

#lets check the distribution of the users as per weather:

f,(ax1,ax2,ax3,ax4) = plt.subplots(4, figsize = (14,14))

registered_weather_2011 = sns.boxplot(x = 'weathersit', y = 'registered', data = df_2011, ax = ax1).set(ylabel =
'registered_2011', title = 'Distribution of users by weather')
```

casual_weather_2011 = sns.boxplot(x = 'weathersit', y = 'casual', data = df_2011, ax = ax2).set(ylabel = 'casual_2011')

registered_weather_2012 = sns.boxplot(x = 'weathersit', y = 'registered', data = df_2012, ax = ax3).set(ylabel = 'registered_2012')

casual_weather_2012 = sns.boxplot(x = 'weathersit', y = 'casual', data = df_2012, ax = ax4).set(ylabel = 'casual_2012')

##

#It is clear from the below plot that users prefer to rent a bicycle only during clear and little cloudy weather,

#both registered and casua users both in 2011 and 2012 has rented bicycles mostly only in 1 and weathers.

#lets check the distribution of the users as per season:

f,(ax1,ax2,ax3,ax4) = plt.subplots(4, figsize = (14,14))

registered_season_2011 = sns.boxplot(x = 'season', y = 'registered', data = df_2011, ax = ax1).set(ylabel = 'registered_2011', title = 'Distribution of users by season')

casual_season_2011 = sns.boxplot(x = 'season', y = 'casual', data = df_2011, ax = ax2).set(ylabel = 'casual_2011')

registered_season_2012 = sns.boxplot(x = 'season', y = 'registered', data = df_2012, ax = ax3).set(ylabel = 'registered_2012')

casual_season_2012 = sns.boxplot(x = 'season', y = 'casual', data = df_2012, ax = ax4).set(ylabel = 'casual_2012')

##

#In the year 2011 registered users were high in seasons 2 and 3 followed by 4 and low in 1

#casual users in the year 2011 were high in season 3 and equally distributed in 2 and 4 and low in 1

#In the year 2012 registered users were high in 3 and 4 and follwed by 2 and lowest in 1.

#casual users in the year 2012 were high in 2 and 3 low in 1.

f,(ax1,ax2) = plt.subplots(2, figsize = (14,14))

temp_2011 = sns.boxplot(x = 'season', y = 'temp', data = df_2011, ax = ax1).set(ylabel = 'temp_2011', title = 'Distribution of temp by season')

temp_2012 = sns.boxplot(x = 'season', y = 'temp', data = df_2012, ax = ax2).set(ylabel = 'temp_2012')

##

#temperature is high in the season 3 and followed by 2 and 4 low in season 1........

#thus the users prefer to rent bicycles in the seasons 2 and 3 more where temperatures are high and doesnt prefer to rent while temp is low season 1

f,(ax1,ax2) = plt.subplots(2, figsize = (14,14))

temp_2011 = sns.boxplot(x = 'season', y = 'atemp', data = df_2011, ax = ax1).set(ylabel = 'atemp_2011', title = 'Distribution of atemp by season')

temp_2012 = sns.boxplot(x = 'season', y = 'atemp', data = df_2012, ax = ax2).set(ylabel = 'atemp_2012')

f,(ax1,ax2) = plt.subplots(2, figsize = (14,14))

```
atemp_2011 = sns.boxplot(x = 'season', y = 'windspeed', data = df_2011, ax = ax1).set(ylabel = 'windspeed_2011', title = 'Distribution of windspeed by season')

atemp_2012 = sns.boxplot(x = 'season', y = 'windspeed', data = df_2012, ax = ax2).set(ylabel = 'windspeed_2012')

###########################################################

#from this plot and plots obtained from before plots we can know that in the seasons 2 and 3 windspeeds are low than other,

#in the seasons 2 and 3 no of bicycle rented are high

#windspeed is inversly propotional to no of users or count of rented bicycles

f,(ax1,ax2) = plt.subplots(2, figsize = (14,14))

humidity_2011 = sns.boxplot(x = 'season', y = 'hum', data = df_2011, ax = ax1).set(ylabel = 'humidity_2011', title = 'Distribution of humidity by season')

humidity_2012 = sns.boxplot(x = 'season', y = 'hum', data = df_2012, ax = ax2).set(ylabel = 'humidity_2012')

###########################################################

#similar to windspeed , humidity is also less in season 2 and 3 where no of rented bikes are high thus we can interpret

#lesser the humidity higher the users.

##########looking for the distributions of all the numerical variable:

for i,col in enumerate(df.columns):

    if df.iloc[:,i].dtypes == 'int64'  or df.iloc[:,i].dtypes == 'float64':

        print(i, col)

        plt.hist(df.iloc[:,i])

        plt.show()

#lets check for the correlation:

continious_variables = ['instant','temp','atemp','hum','windspeed','casual','registered','cnt']

correlation = df.loc[:,continious_variables]

f,ax = plt.subplots(figsize = (10,10))

corr = correlation.corr()

sns.heatmap(corr,

         mask = np.zeros_like(corr, dtype = np.bool),

        cmap = sns.diverging_palette(220,10,as_cmap = True),

        square = True, ax = ax)

#according to the plot temp and atemp are highly correlated thus we shall remove atemp from modelling

df  = df.drop(columns = ['atemp','dteday','instant'])

dummy_var = pd.get_dummies(df)
```

```
dummy_var.columns

dummy_var_final = dummy_var.drop(columns =
['season_4','yr_1','holiday_1','weekday_6','workingday_1','weathersit_3','mnth_12'])

#lets split the data into train and test split:

from sklearn.cross_validation import train_test_split

x = dummy_var_final.drop(columns = ['registered','casual','cnt'])

y  = dummy_var_final[['registered','casual','cnt']]

x_train,x_test,y_train,y_test = train_test_split(x, y, test_size = 0.20, random_state = 15)

from sklearn.ensemble import RandomForestRegressor

RF_model = RandomForestRegressor(n_estimators = 700)

#fitting and predicting the model to registered users seperately:

register = RF_model.fit(x_train, y_train['registered'])

pred_register = register.predict(x_test)

#fitting and predicting the model for casual users seperately:

casual = RF_model.fit(x_train, y_train['casual'])

pred_casual = casual.predict(x_test)

count = pred_register + pred_casual

count

y_test_cnt_arry = np.array(y_test['cnt'])

y_test_cnt_arry

def MAPE( y_true, y_pred):

   mape = np.mean(np.abs((np.array(y_true) - np.array(y_pred))/y_true))*100

   return mape

MAPE(y_test_cnt_arry, count)

#11.1503

from sklearn.tree import DecisionTreeRegressor

fit_reg = DecisionTreeRegressor(max_depth = 7).fit(x_train, y_train['registered'])

fit_cas = DecisionTreeRegressor(max_depth = 7).fit(x_train, y_train['casual'])

predict_regress_decision = fit_reg.predict(x_test)

predict_casual_decision = fit_cas.predict(x_test)

count_decision = predict_regress_decision + predict_casual_decision

count_decision = count_decision.astype('int')

def MAPE( y_true, y_pred):
```

```
    mape = np.mean(np.abs((np.array(y_true) - np.array(y_pred))/y_true))*100
    return mape
MAPE(y_test['cnt'], count_decision)
#14.4723
#trying linear regression using stats model:
import statsmodels.api as sm
model_reg = sm.OLS(y_train['registered'], x_train).fit()
model_reg.summary()
model_cas = sm.OLS(y_train['casual'], x_train).fit()
model_cas.summary()
pred_reg_linear = model_reg.predict(x_test)
pred_cas_linear = model_cas.predict(x_test)
count_linear_sm = pred_reg_linear + pred_cas_linear
count_linear_sm = count_linear_sm.astype('int')
count_linear_sm
def MAPE( y_true, y_pred):
    mape = np.mean(np.abs((np.array(y_true) - np.array(y_pred))/y_true))*100
    return mape
MAPE(y_test['cnt'], count_linear_sm)
#14.2173
```