Bike Renting

Sayantan Adak

15th Nov, 2018

Contents

Chapter 1

Introduction

1.1 Problem Statement

People can rent a bike through membership (mostly regular users) or on demand basis (mostly casual users). The objective of this case is to prediction of bike rental count on daily based on the environmental and seasonal settings.

1.2 Understanding the Data

The data set shows daily rentals data for two years(2011 and 2012). Bike demand by registered, casual users are separately given in the data set and the sum of the both users is given as count. As training and test data are not given separately, we are required to split the data in **train** and **test,** and define our machine learning model over train data to predict the number of bike rental counts in test data.

Independent Variables:

instant: Record index
dteday: Date
season: Season (1:springer, 2:summer, 3:fall, 4:winter)
yr: Year (0: 2011, 1:2012)
mnth: Month (1 to 12)
hr: Hour (0 to 23)
holiday: weather day is holiday or not (extracted fromHoliday Schedule)
weekday: Day of the week
workingday: If day is neither weekend nor holiday is 1, otherwise is 0.
weathersit: (extracted fromFreemeteo)
1: Clear, Few clouds, Partly cloudy, Partly cloudy
2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

temp: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39 (only in hourly scale)

atemp: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_maxt_ min), t_min=-16, t_max=+50 (only in hourly scale)
hum: Normalized humidity. The values are divided to 100 (max)
windspeed: Normalized wind speed. The values are divided to 67 (max)

Dependent Variables:

casual: count of casual users
registered: count of registered users
cnt: count of total rental bikes including both casual and registered

Given below is the top 5 observation of the data set:

table 1.2.1: Data(column 1-8)

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

table 1.2.2: Data(column 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	0.200000	0.212122	0.590435	0.160296	108	1454	1562
1	0.226957	0.229270	0.436957	0.186900	82	1518	1600

Chapter 2

Methodology

2.1 Exploratory Data Analysis

Data exploration is a crucial stage for any predictive model. The quality of the input decides the quality of the output.

Train Data: The predictive model is always built on train data set. An intuitive way to identify the train data is, that it always has the 'response variable' included.

Test Data: Once the model is built, it's accuracy is 'tested' on test data. This data always contains less number of observations than train data set. Also, it does not include 'response variable'.

Need for Data Cleaning or Data Preparation:

- Dataset might contain discrepancies in the names or codes.
- Dataset might contain outliers or errors.
- Dataset lacks your attributes of interest for analysis.
- All in all the dataset is not qualitative but is just quantitative.

From the experiences of many data scientists it is said that the data exploration, cleaning and preparation can take upto 70% time of the total project.

2.1.1 Variable type Identification

If we see the structure of the data set then we can observe there are different kinds of variable: few are numerical and few are categorical. So we have to treat different kinds of variables in different ways.

```
data.frame':   731 obs. of  16 variables:
```

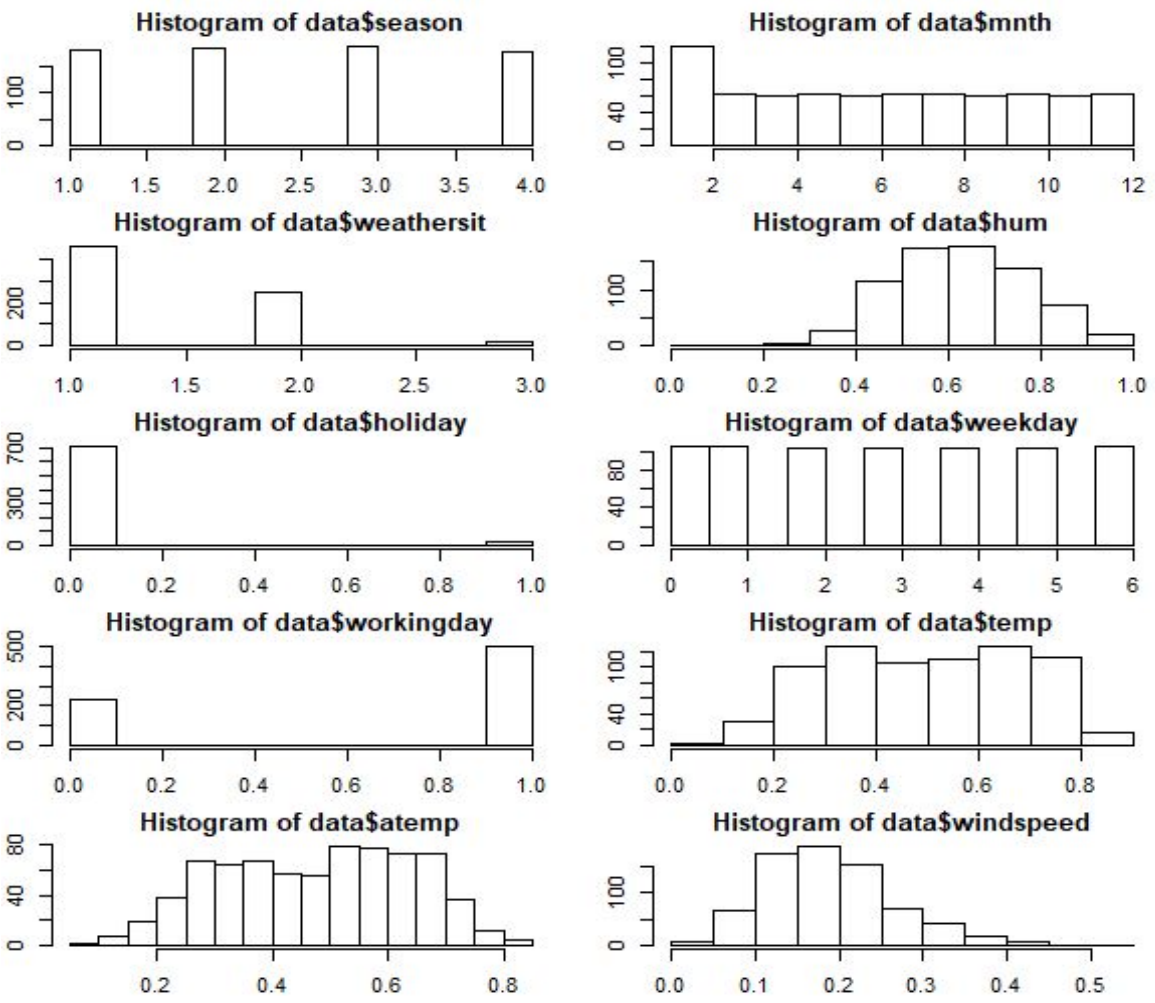
```
 $ instant   : int  1 2 3 4 5 6 7 8 9 10 ...
 $ dteday    : Factor w/ 731 levels "2011-01-01","2011-01-02",..: 1 2 3 4 5
6 7 8 9 10 ...
 $ season    : int  1 1 1 1 1 1 1 1 1 1 ...
 $ yr        : int  0 0 0 0 0 0 0 0 0 0 ...
 $ mnth      : int  1 1 1 1 1 1 1 1 1 1 ...
 $ holiday   : int  0 0 0 0 0 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 ...
```

Intuitively we can tell the number of instants and **dteday** are not our concern for predicting bike rentals. So we can exclude these two variables for further analysis.

2.1.2 Distribution of numerical variables

Understand the distribution of numerical variables and generate a frequency table for numeric variables. Although **yr** is categorized as numerical, it has two unique values(0 and 1). So we can treat it as categorical variable. Now, I'll test and plot a histogram for each numerical variables and analyze the distribution.

Fig 2.1.2: Distribution of numeric variables



Few inferences can be drawn by looking at the these histograms:

- **Season** has four categories of almost equal distribution
- **mnth** and **weekday** are showing similar inferences
- **weathersit** has one higher contribution
- **temp, atemp,hum** and **windspeed** looks naturally distributed

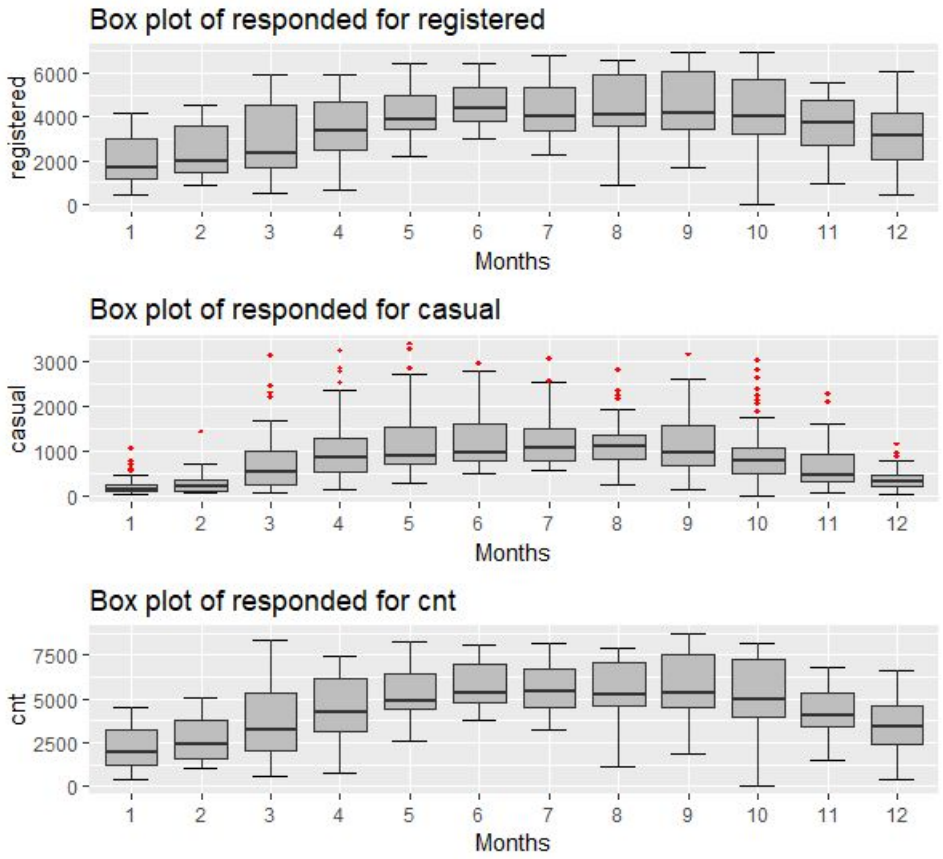
From the above figure we can see the variables **season,mnth,weathersit,holiday,weekday,working day** are discrete. So we can treat them as categorical variables to reduce complexity of the model.

2.1.3 Multivariate analysis

So far we have fair understanding of the distribution of the data. Now we will plot count of users on the basis of each predictor variable and analyse the trends of the bike rentals.

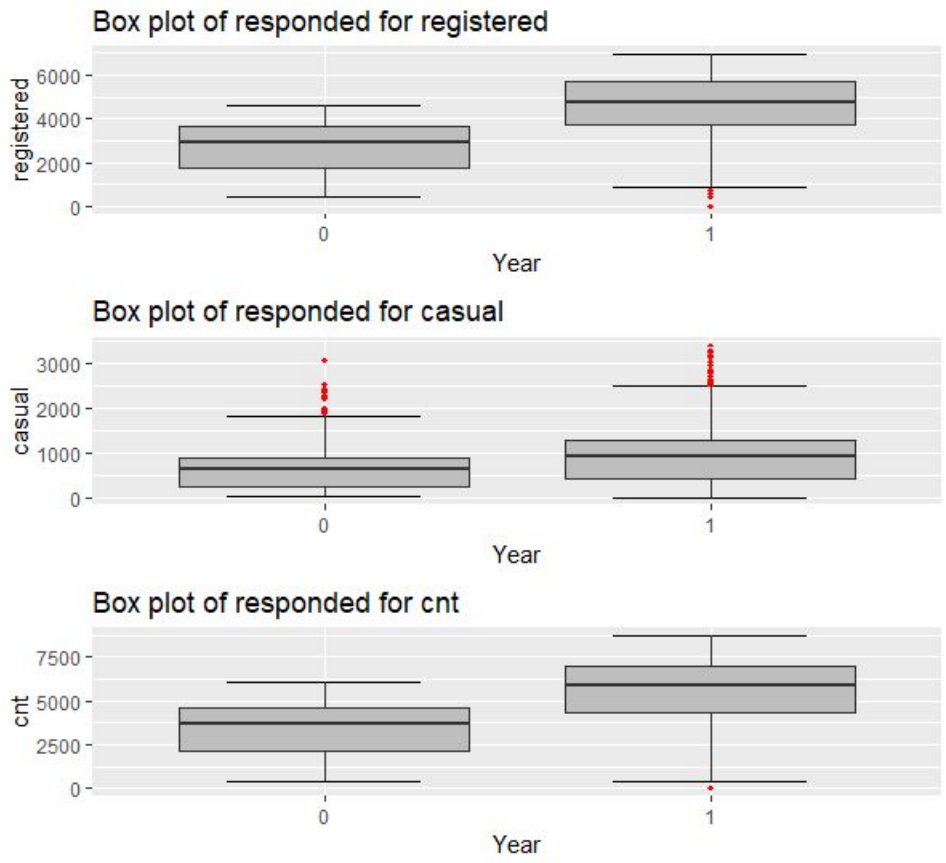
- **Monthly trends –** Let's check the boxplot of registered, casual and cnt with respect to each month.

Fig 2.1.3.1: boxplot of each users monthly basis



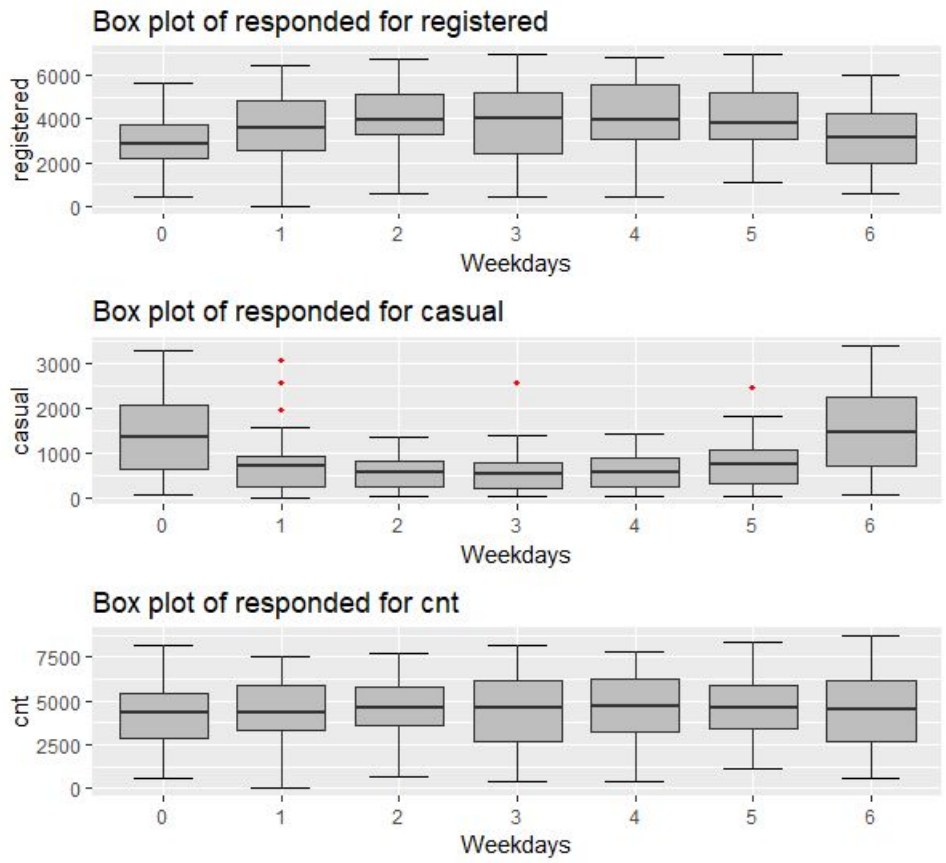
- **Yearwise trends –** Now I will check yearwise boxplot of each types of users

Fig 2.1.3.2: Yearly trends



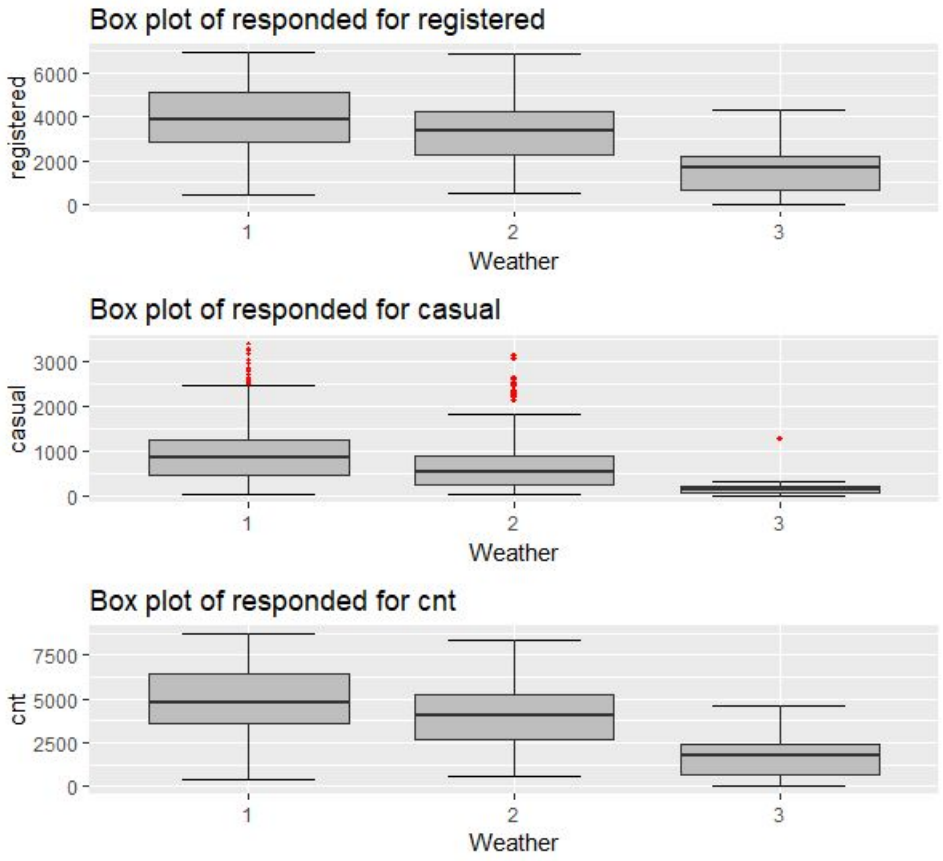
Here **0** refers to year 2011 and **1** refers to year 2012. We can clearly see from the boxplot that the number of each users increases in 2012 as compared to 2011

- **Daily trend –** Let's check daily trend of users



Here I have plotted the boxplot of each users over weekdays(from 0-6 i.e. Sunday to Saturday). There is not much difference of total count of users over the week but the count of casual users increases over the weekends.

- **Dependency on weather –** Now check if the number of users are dependent on weathersit given in data data.



It is observed that the number of users is high mostly on weather type 1 i.e. clear wether.

- **Temperature,windspeed** and **Humidity –** As these are continuous variables we can check the correlation of each users with respect to these variables.

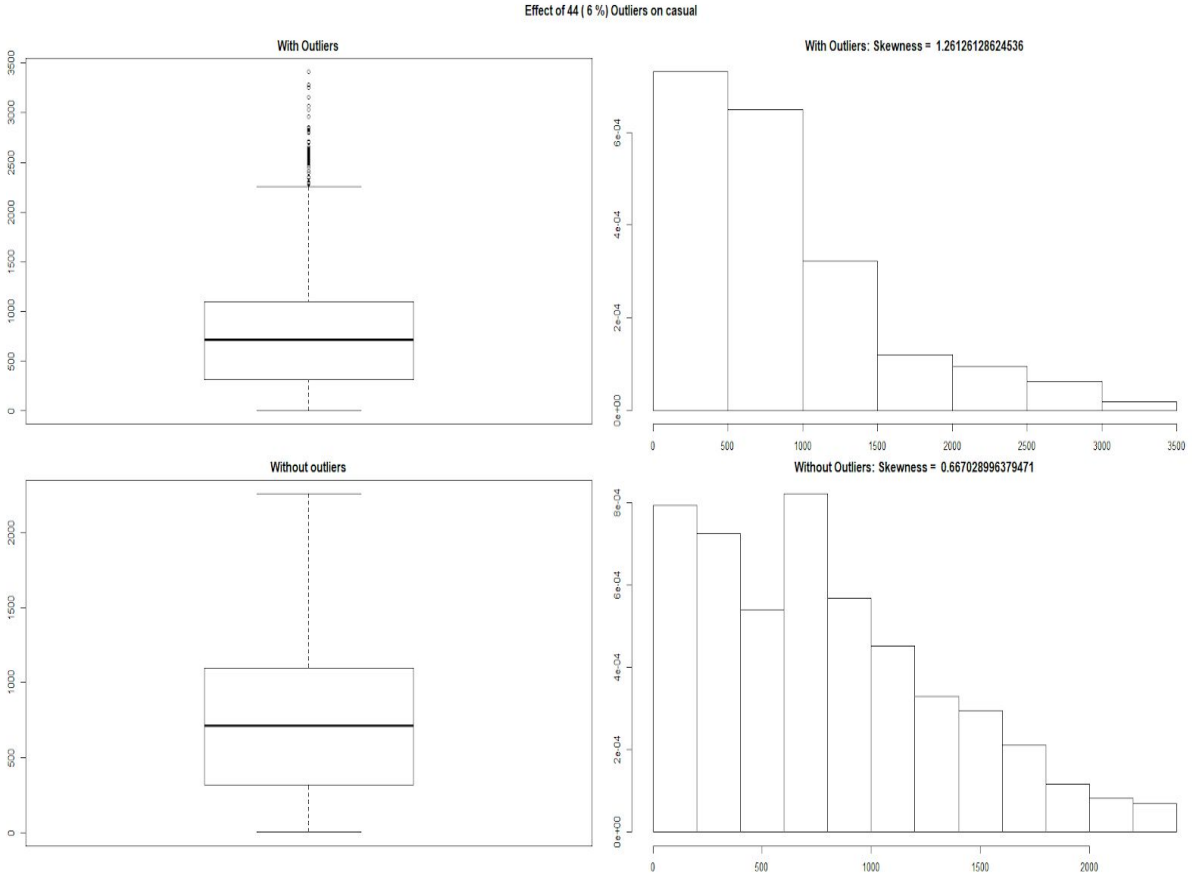
	temp	atemp	hum	windspeec	casual	registered	cnt
temp	1	0.991702	0.126963	-0.15794	0.543285	0.540012	0.627494
atemp	0.991702	1	0.139988	-0.18364	0.543864	0.544192	0.631066
hum	0.126963	0.139988	1	-0.24849	-0.07701	-0.09109	-0.10066
windspeed	-0.15794	-0.18364	-0.24849	1	-0.16761	-0.21745	-0.23454
casual	0.543285	0.543864	-0.07701	-0.16761	1	0.395282	0.672804
registered	0.540012	0.544192	-0.09109	-0.21745	0.395282	1	0.945517
cnt	0.627494	0.631066	-0.10066	-0.23454	0.672804	0.945517	1

It is observed that count of users(both for casual and registered) are moderately high positive correlated with **temperature** and **atem** and slightly negative correlated with **humidity** and **windspeed.**

2.1.4 Outlier analysis

We have observed few outliers for casual users. If we don't treat them properly then it will affect our overall predictions.

We have used boxplot method to visualize the outliers and will treat them as missing values. Let's have a look of outlier effects on the distribution of casual users.



We have succeeded to reduce the skewness of the distribution of **casual** by removing ouliers. First we have replaced the outliers with **missing values**. Then we have imputed missing values using **knnImputation** method.

2.1.5 Feature Scaling

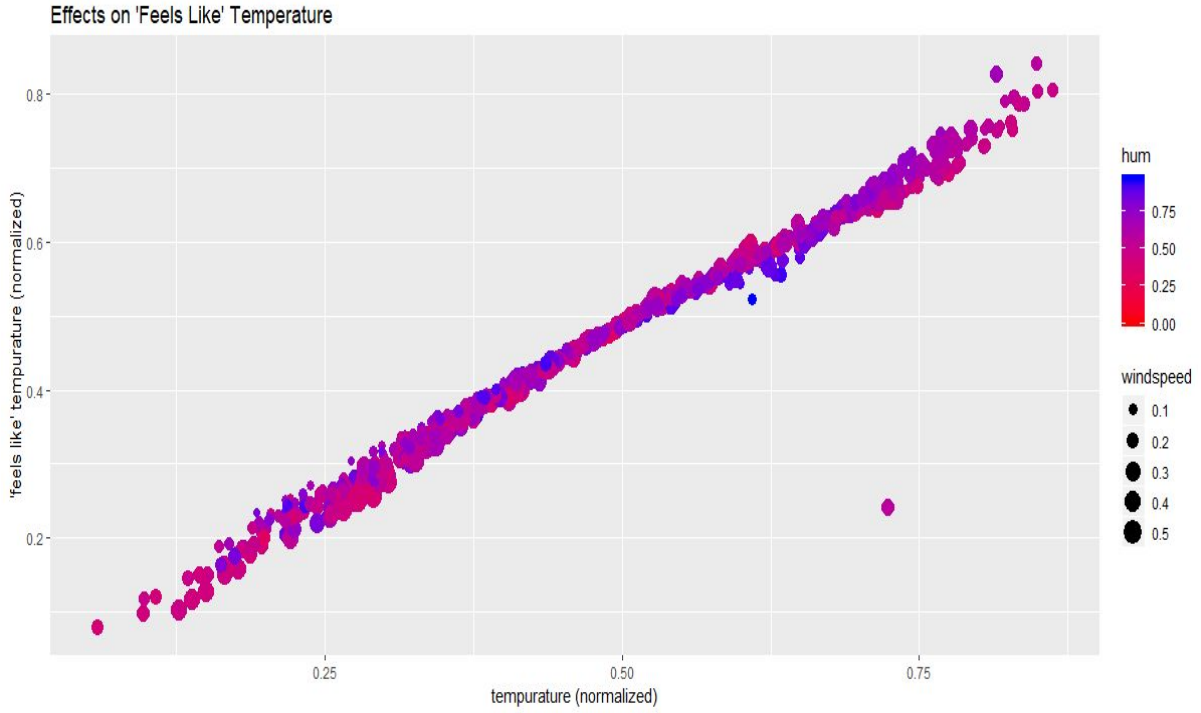
In the raw data there are different feature of different scale in their magnitude. If we feed them directly into the machine learning model without scaling, object function will not work out properly and the result will be biased. That's why we have gone for feature scaling as a data pre-processing technique to reduce variation either within or between variables. We have normalized all the numeric predictors to a fixed range using the formula:

Valuenew = (Value - minValue)/(maxValue - minValue)

This changes the range of each numerical predictor variables to 0 to 1.

2.1.6 Other necessary pre-processing

Let's look at the variable **atemp.** The atemp variable in the data set is a 'feels like' temperature based on weather factors. The plot below illustrates this relationship.



This effect shows that windspeed is low for lower temperature and humidity is quite high for higher temperature. Although 'Feels like' temperature is linearly dependent on temperature, there are few outliers can be seen in the diagram.

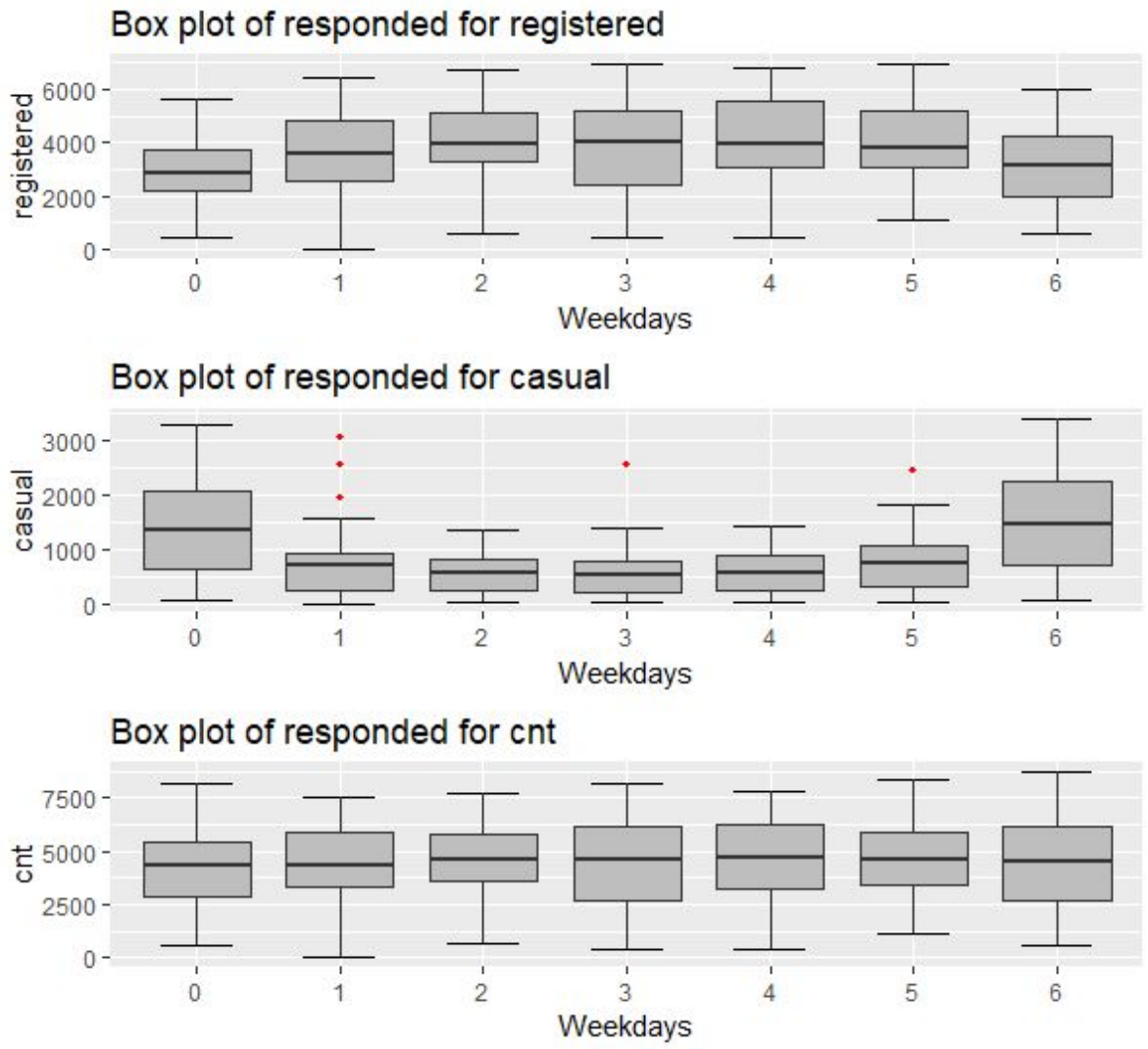
For simplicity we can say **temp** and **atemp** have similar affect on counts. So we can exclude either of these.

2.2 Feature Engineering

For reducing complexity of the model and getting better prediction we may need to exclude few variables that are not important for prediction.

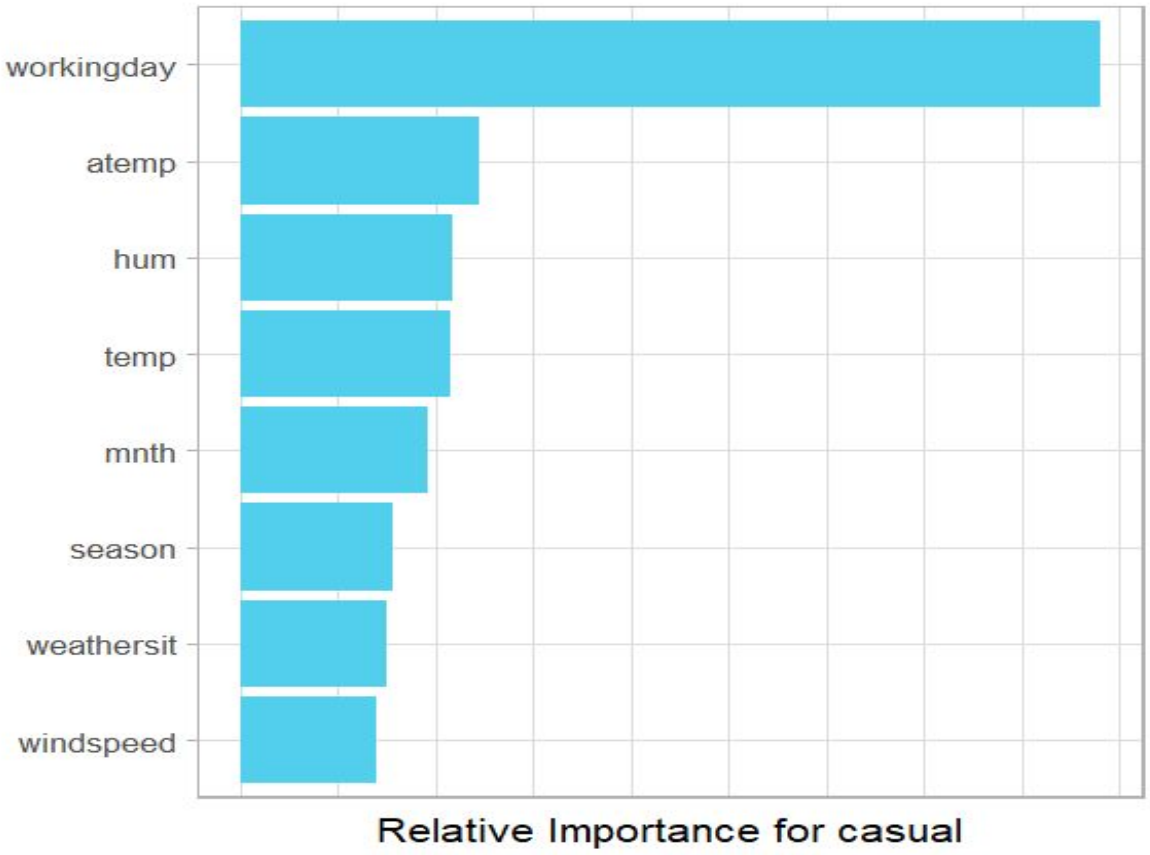
We have already excluded two variables(**instant** and **dteday**) from raw data before pre-processing. We have already got an inference from the **yr** variable that in 2012 the number of counts is higher than that of in 2011. So we don't need to check feature importance of **yr** again.

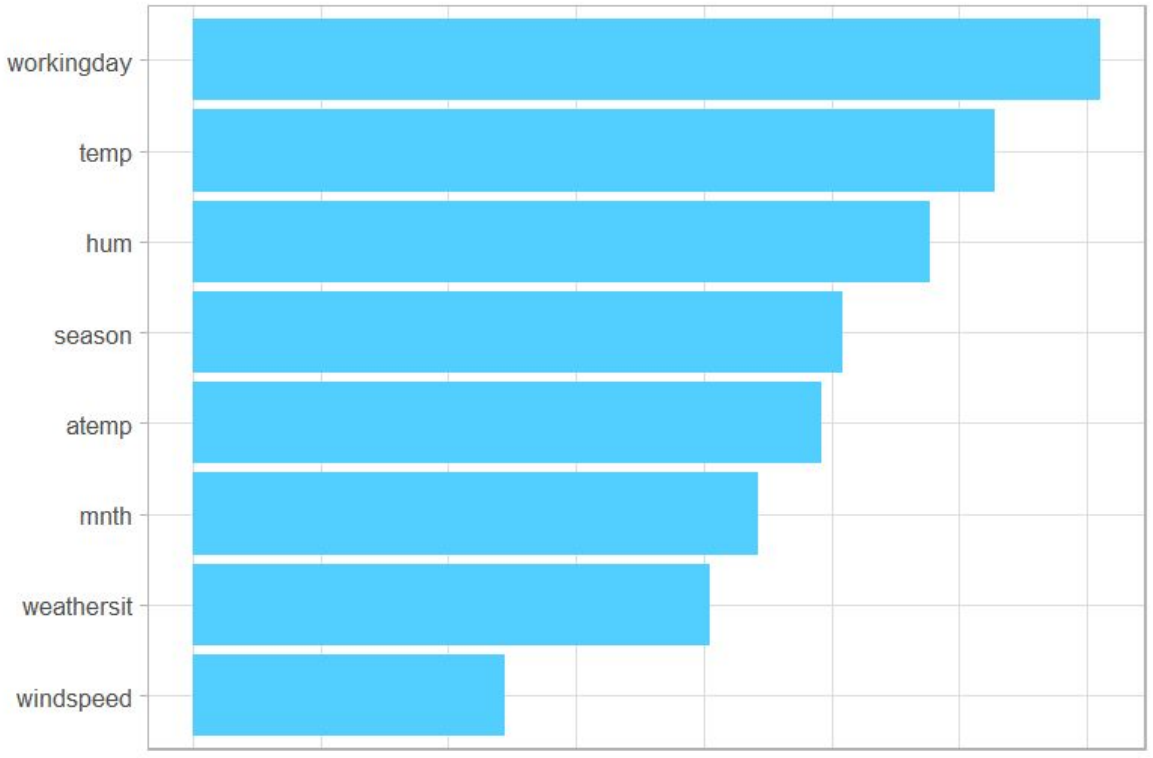
We can see from the boxplot of count of users on daily basis, that there is not much difference in the median value. Less number of registered users and more number of casual users can be observed in the weekends.



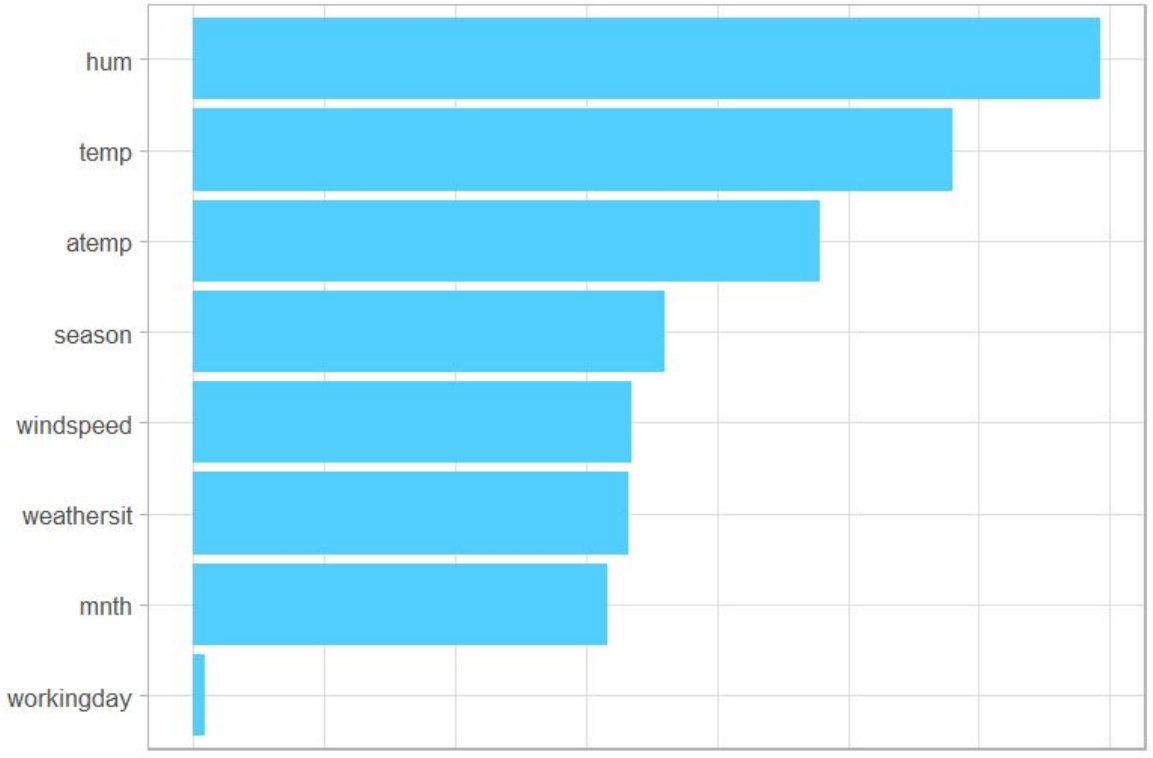
If we closely look into the data set then we can observe **holiday** and **weekends** are considered as non-working days. **holiday** and **weekends** can be described by **workingday** variable. We are not going to consider **weekday** and **holiday** for prediction of counts.

Now take a look over relative importance of each predictor variables(excluding the above). We have used **randomForest** machine learning algorithm for checking the variable importance.


workingday
atemp
hum
temp
mnth
season
weathersit
windspeed
Relative Importance for casual



Relative Importance for registered



Relative Importance for total count

As we can see **workingday** takes most important role for registered and casual users but not so important for total count.

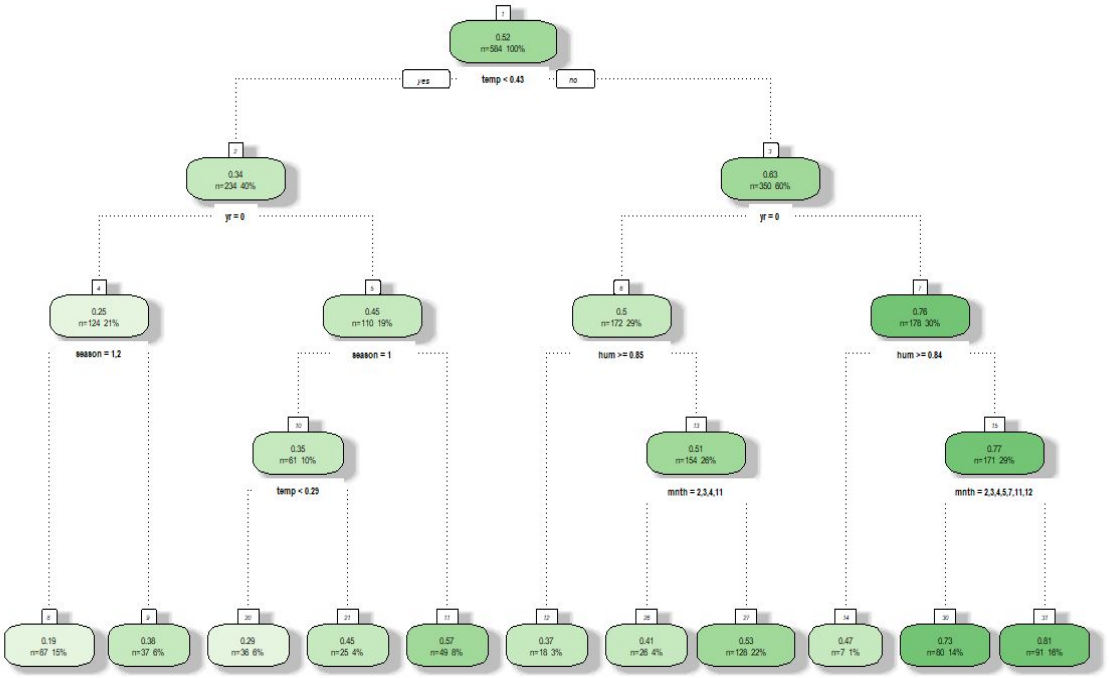
2.3 Model building

2.3.1 Model selection

Our target is to predict the count of bike rentals for registered, casual and total users. As the target variable is numeric we can use statistical method like **linear regression** or **Decision Tree Regression** model for prediction. We have used **Decision Tree Regression** for our prediction.

2.3.2 Regression Tree

Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with **decision nodes** and **leaf nodes**. A decision node has two or more branches each representing values for the attribute tested. Leaf node represents a decision on the numerical target. The topmost decision node in a tree which corresponds to the best predictor called root node. Decision trees can handle both categorical and numerical data. **Decision Tree Regression** is used to handle numerical response variable.



This is our regression tree model for **cnt** . In each node it makes a decision. Based on the decision the branching occurs until the leaf node arrives.

2.3.3 Multiple Linear Regression

multiple linear regression analysis is the task of fitting a single line through a scatter plot. More specifically the multiple linear regression fits a line through a multi-dimensional space of data points. The simplest form has one dependent and two independent variables. The dependent variable may also be referred to as the outcome variable or regressand. The independent variables may also be referred to as the predictor variables or regressors.

There are 3 major uses for multiple linear regression analysis. First, it might be used to identify the strength of the effect that the independent variables have on a dependent variable.

Second, it can be used to forecast effects or impacts of changes. That is, multiple linear regression analysis helps us to understand how much will the dependent variable change when we change the independent variables. For instance, a multiple linear regression can tell you how much GPA is expected to increase (or decrease) for every one point increase (or decrease) in IQ.

Third, multiple linear regression analysis predicts trends and future values. The multiple linear regression analysis can be used to get point estimates. An example question may be "what will the price of gold be 6 month from now?"

When selecting the model for the multiple linear regression analysis, another important consideration is the model fit. Adding independent variables to a multiple linear regression model will always increase the amount of explained variance in the dependent variable (typically expressed as R^2). Therefore, adding too many independent variables without any theoretical justification may result in an over-fit model.

Challenge for categorical independent variable

Regression analysis requires numerical variables. So, when a researcher wishes to include a categorical variable in a regression model, supplementary steps are required to make the results interpretable.

In these steps, the categorical variables are recoded into a set of separate binary variables. This recoding is called "**dummy coding**" and leads to the creation of a table called ***contrast matrix***. This is done automatically by statistical software, such as R.

Nominal variables, or variables that describe a characteristic using two or more categories, are commonplace in quantitative research, but are not always useable in their categorical form. A common workaround for using these variables in a regression analysis is dummy coding, but there is often a lot of confusion (sometimes even among dissertation committees!) about what dummy variables are, how they work, and why we use them. With this in mind, it is important that the researcher knows how and why to use dummy coding so they can defend their correct (and in many cases, necessary) use.

Dummy coding is a way of incorporating nominal variables into regression analysis, and the reason why is pretty intuitive once you understand the regression model. Regressions are most commonly known for their use in using continuous variables (for instance, hours spent studying) to predict an

outcome value (such as grade point average, or GPA). In this example, we might find that increased study time corresponds with increased GPAs.

Now, what if we wanted to also know if favourite class (e.g., science, math, and language) corresponded with an increased GPA. Let's say we coded this so that science = 1, math = 2, and language = 3. Looking at the nominal favourite class variable, we can see that there is no such thing as an increase in favorite class – math is not higher than science, and is not lower than language either. This is sometimes referred to as directionality, and knowing that a high versus low score means something is an integral part of regression analysis. Luckily, there is a way around this! Enter: dummy coding.

Dummy coding allows us to turn categories into something a regression can treat as having a high (1) and low (0) score. Any binary variable can be thought of as having directionality, because if it is higher, it is category 1, but if it is lower, it is category 0. This allows the regression look at directionality by comparing two sides, rather than expecting each unit to correspond with some kind of increase. Let's go back to the favorite class variable. Remember, we originally coded this as science = 1, math = 2, and language = 3. To give the regression something to work with, we can make a separate column, or variable, for each category. These columns will each show whether each category was a student's favorite; if a student has a (1), the high (or yes) score, in the science column, science is their favorite, but if they have a (0), the low (or no) score, science did not make the cut. The same goes for each of the dummy variables, as they are called. Below is an example of how this ends up working out:

Dummy variables
Student Favorite class Science Math Language

		Dummy variables		
Student	Favorite class	Science	Math	Language
1	Science	**1**	0	0
2	Science	**1**	0	0
3	Language	0	0	**1**
4	Math	0	**1**	0
5	Language	0	0	**1**
6	Math	0	**1**	0

Now, looking at this you can see that knowing the values for two of the variables tell us what value the final variable has to be. Let's look at student 1; we know they can only have one favorite class. If we know science = 1 and math = 0, we know that language has to be 0 as well. The same goes for student 5; we know that science is not their favorite, nor is math, so language has to have a yes (or 1).

For this reason, we do not use all three categories in a regression. Doing so would give the regression redundant information, result in multicollinearity, and break the model. This means we have to leave one category out, and we call this missing category the reference category.

Let's take a look at our Linear Regression Model:

```
Residuals:
    Min      1Q  Median      3Q     Max
-3956.5  -394.3    57.5   469.5  3268.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  1710.90     276.14   6.196 1.12e-09 ***
season2      1093.76     207.02   5.283 1.82e-07 ***
season3       924.44     243.46   3.797 0.000162 ***
season4      1695.63     202.48   8.374 4.45e-16 ***
yr1          1958.75      67.90  28.849  < 2e-16 ***
mnth2         189.85     165.08   1.150 0.250622
mnth3         562.69     191.10   2.945 0.003368 **
mnth4         296.07     283.30   1.045 0.296438
mnth5         581.24     303.41   1.916 0.055913 .
mnth6         408.35     319.17   1.279 0.201278
mnth7         -92.16     357.94  -0.257 0.796903
mnth8         288.01     344.09   0.837 0.402940
mnth9        1051.50     298.09   3.527 0.000454 ***
mnth10        478.23     272.20   1.757 0.079485 .
mnth11       -132.52     260.86  -0.508 0.611645
mnth12        -79.26     205.72  -0.385 0.700163
workingday1   174.40      72.46   2.407 0.016408 *
weathersit2  -409.64      89.99  -4.552 6.51e-06 ***
weathersit3 -1864.96     235.84  -7.908 1.40e-14 ***
temp         4563.96     475.13   9.606  < 2e-16 ***
hum         -1708.34     352.92  -4.841 1.68e-06 ***
```

```
windspeed   -2806.62     495.25  -5.667 2.32e-08 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 794.7 on 562 degrees of freedom
Multiple R-squared:  0.8381,   Adjusted R-squared:  0.832
F-statistic: 138.5 on 21 and 562 DF,  p-value: < 2.2e-16
```

As we can see the categorical variables(with n categories) are internally divided into n-1 variables by **dummy coding.**

From the adjusted **R-squared** value we can explain about 83% of the data using our multiple linear regression model. Looking at the **F-statistics** and **p-value** we can reject the null hypothesis.

Chapter 3

Conclusion

3.1 Model Evaluation

Now we have few models for predicting target variable, we need to decide which model to choose. The performance of any classification model does not only depend upon its prediction accuracy. Depending upon business understanding we need to consider different kinds of metrics to evaluate

our models. The choice of metric completely depends on the type of model and the implementation plan of the model.

There are different types of metrics depend on what kind of problem you are trying to solve:

For Regression Problem :

a. RMSE
b. MSE
c. MAPE

Regression error metrics: We want to know how well the model predicts new data, not how well it fits the data it was trained with. Key component of most regression measures are difference between actual y and predicted y ("error")

- **MSE:** The mean squared error (MSE) or mean squared deviation (MSD) of an estimator measures the average of the squares of the errors or deviations that is, the difference between the estimator and what is estimated.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$$

- **RMSE:** It stands for Root Mean Squared Error/Deviation. Square the errors, find their average, take the square root

$$\mathrm{RMSD} = \sqrt{\frac{\sum_{t=1}^{n}(\hat{y}_t - y_t)^2}{n}}.$$

- **MAE:** The mean absolute error (MAE) is a quantity used to measure how close forecasts or predictions are to the eventual outcomes. The mean absolute error is given by

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}|f_i - y_i|$$

- **MAPE:** Stands for Mean absolute percentage error. Measures accuracy as a percentage

Of error.

$$\mathrm{M} = \frac{1}{n}\sum_{t=1}^{n}\left|\frac{A_t - F_t}{A_t}\right|$$

We have only calculated the **MAE** and **MAPE** for evaluating our models.

For regression tree(predicting cnt) :

```
   mae        mape
0.07638475 0.17638277
```

Accuracy = 100 – mape * 100 =82%

For Linear Regression(predicting cnt):

```
   mae        mape
0.06802317 0.16341551
```

Accuracy = 100 – mape * 100 = 84%

3.2 Model Selection

As we can see **The Decision tree** gives 82% accuracy in predicting total count of bike rentals and **Linear Regression** Model gives slightly better(84% accuracy) result. So we can prefer **Linear Regression** for this purpose but we need to be careful about the categorical predictor variables.

Appendix A : Extra Figures

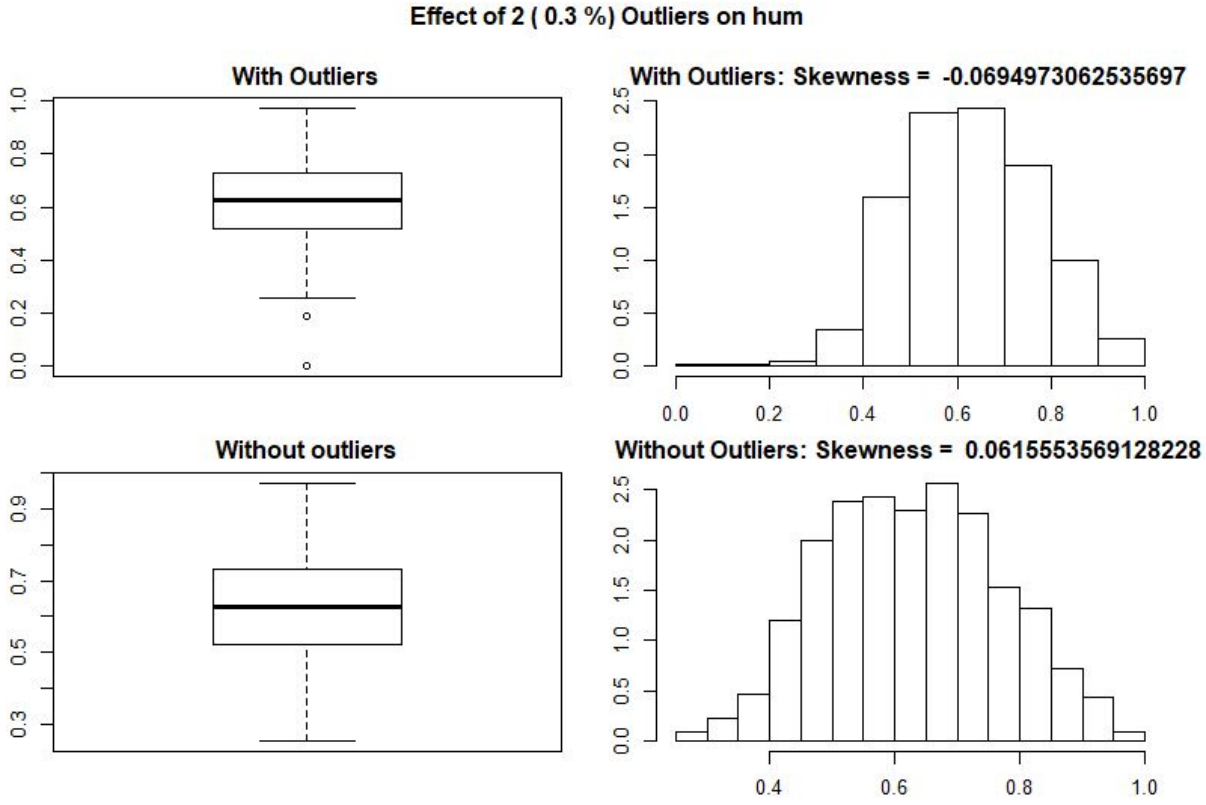


Figure 4.1: Outlier effect on humidity

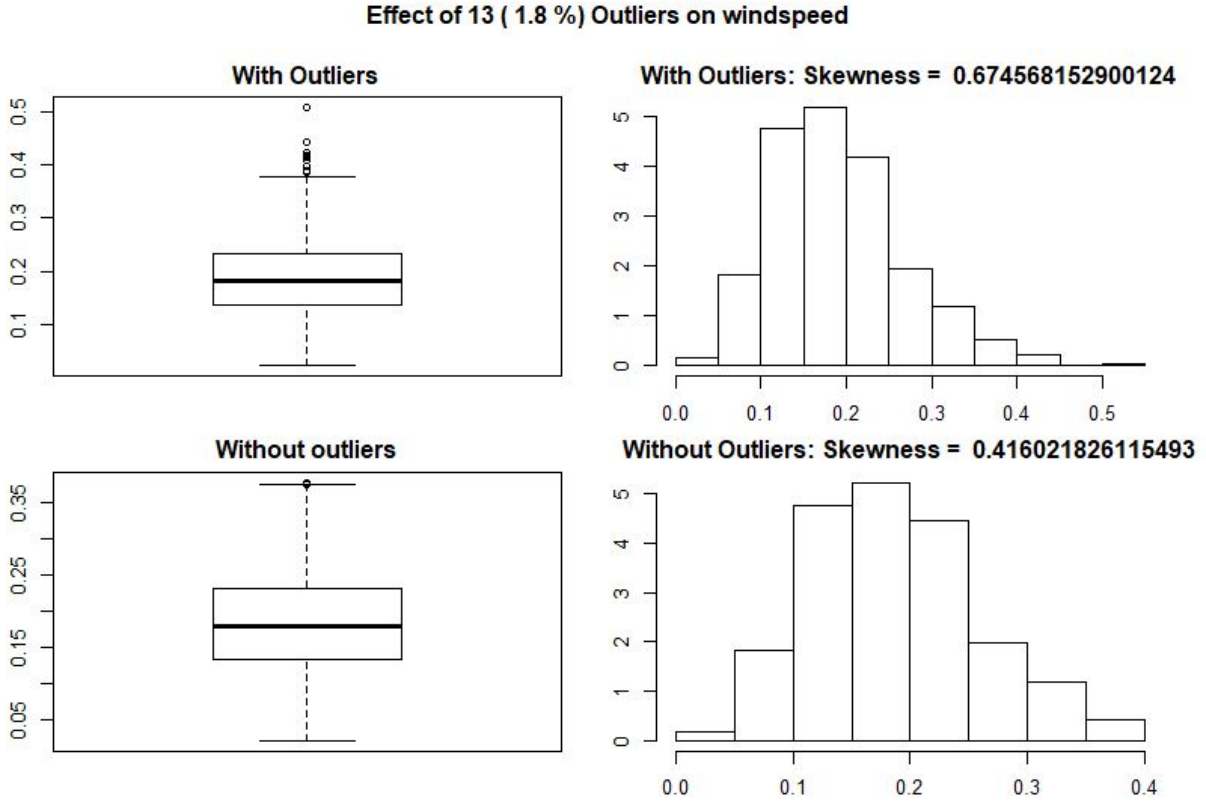


Figure 4.2: Outlier effect on windspeed

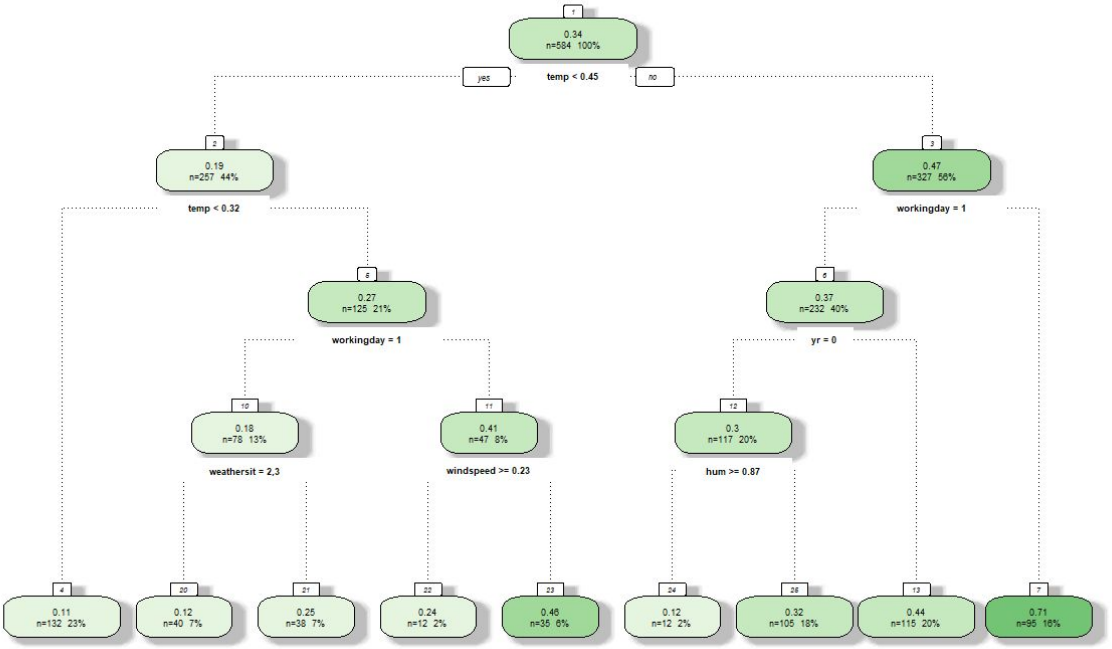


Figure 4.3 Regression tree for casual users

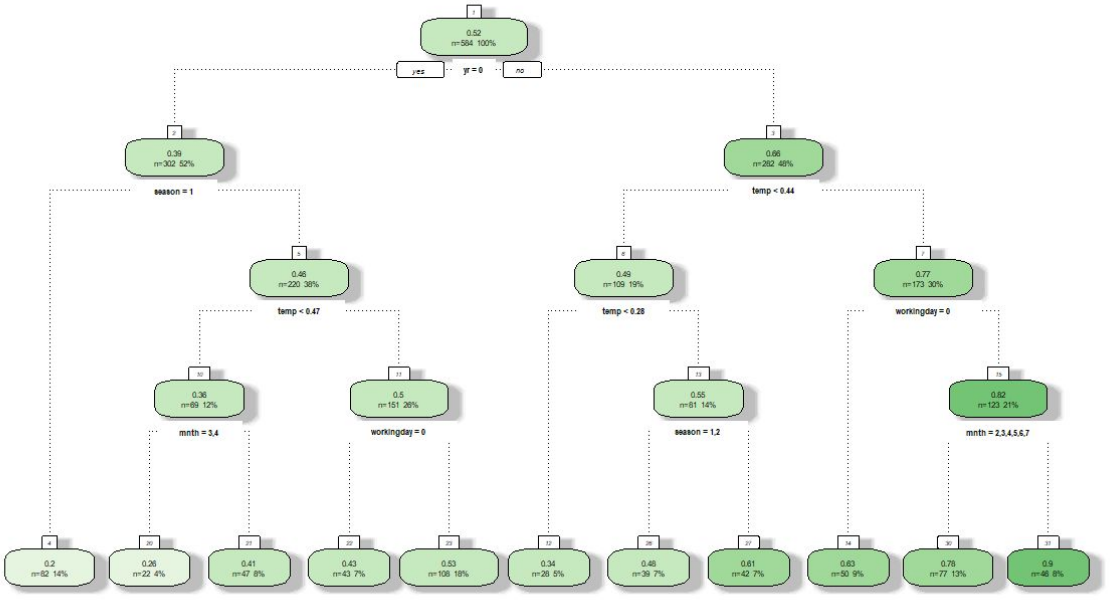


Figure 4.4 Regression tree for Registered users

Appendix B: R code

Distribution of predictors:

```
par(mfrow=c(5,2))
par(mar = rep(2, 4))
hist(data$season)
hist(data$mnth)
hist(data$weathersit)
hist(data$hum)
hist(data$holiday)
hist(data$weekday)
hist(data$workingday)
hist(data$temp)
hist(data$atemp)
```

```
hist(data$windspeed)
```

Multivariate Analysis(only for workingday basis):

```
response_data <- subset(data,select=c(registered,casual,cnt))
cnames <- colnames(response_data)
library(ggplot2)
for (i in 1:length(cnames)){
      assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]),x=data$workingday), data = response_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='Working Day')+
        ggtitle(paste("Box plot of responded for",cnames[i])))
}
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=1)
```

Correlation matrix:

```
data.frame(cor(numeric_data))
```

Effects of Outliers:

```
outToNa <- function(vl, df) {
 for (i in vl) {
   outlier <- boxplot.stats(df[, i])$out
   df[, i] <- ifelse(df[, i] %in% outlier,
                  NA, df[, i])
 }
 return(df)
}
```

```
outliereff <- function(i, df) {
 total = length(df[, i])
 par(mfrow = c(2, 2), oma = c(0, 0, 3,
                        0))
 boxplot(df[, i], main = "With Outliers")
 hist(df[, i], main = paste("With Outliers: Skewness = ",skewness(df[,i])),
     xlab = NA, ylab = NA, prob = TRUE)
 #skewness(df[,i],)
 df <- outToNa(i, df)
 out <- sum(is.na(df[, i]))
 per <- round((out)/total * 100, 1)
 df <- knnImputation(df)
 boxplot(df[, i], main ="Without outliers")
 hist(df[, i], main = paste("Without Outliers: Skewness = ",skewness(df[,i])),
     xlab = NA, ylab = NA, prob = TRUE)

 title(paste("Effect of", out, "(", per,
           "%)", "Outliers on", colnames(df)[i],
           sep = " "), outer = TRUE)



}

var_list <- list(10,11,12)
for(vl in var_list){
 outliereff(vl,data)
}
```

Normalization:

```
normalized_data <- data
numeric_data <- subset(data,select=c(casual,registered,cnt))
```

```
cnames <- colnames(numeric_data)
for (i in cnames){
 normalized_data[,i] = (normalized_data[,i] - min(normalized_data[,i]))/
   (max(normalized_data[,i])-min(normalized_data[,i]))

}
```

Effects on 'feels like' temperature:

```
gg <- ggplot(data, aes(temp, atemp, color=hum, size=windspeed)) +
 geom_point() +
 scale_colour_gradient(low="red",high="blue") +
 labs(title="Effects on 'Feels Like' Temperature") +
 labs(x="tempurature (normalized)") +
 labs(y="'feels like' tempurature (normalized)")
print(gg)
```

Feature Engineering:

```
feature <- select(data, season,mnth,workingday,weathersit,temp,atemp,hum,windspeed)
rf <- randomForest(feature, data1$cnt, ntree = 100, importance = T)
imp <- importance(rf, type = 1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])

ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +
 geom_bar(stat="identity", fill="#53cfff") +
 coord_flip() +
 theme_light(base_size=16) +
 xlab("") +
 ylab("Relative Importance for total count") +
 theme(plot.title   = element_text(size=18),
      strip.text.x = element_blank(),
      axis.text.x  = element_blank(),
      axis.ticks.x = element_blank())
```

Visual plot of Decision Tree:

```
train_index <- sample(1:nrow(data1), 0.8 * nrow(data1))
train <- data1[train_index,]
test <- data1[-train_index,]

fit <- rpart(cnt ~ ., data=train, method = "anova")
fancyRpartPlot(fit)
```

Complete R code:

```
#### Loading necessary library
library(ggplot2)
library(e1071)
library(DMwR)
library(randomForest)
library(dplyr)
library(rpart)
library(MASS)
library(rattle)
library(rpart.plot)
library(RColorBrewer)

#### Loading Data
raw_data <- read.csv('day.csv')
str(raw_data)

data <- subset(raw_data,select=-c(instant,dteday))

#### Distribution of numeric data
par(mfrow=c(5,2))
par(mar = rep(2, 4))
```

```
hist(data$season)
hist(data$mnth)
hist(data$weathersit)
hist(data$hum)
hist(data$holiday)
hist(data$weekday)
hist(data$workingday)
hist(data$temp)
hist(data$atemp)
hist(data$windspeed)

#### Data type conversion
data$season <- as.factor(data$season)
data$yr <- as.factor(data$yr)
data$mnth <- as.factor(data$mnth)
data$holiday <- as.factor(data$holiday)
data$weekday <- as.factor(data$weekday)
data$workingday <- as.factor(data$workingday)
data$weathersit <- as.factor(data$weathersit)

#### Multivariate analysis for each types of users(shown only w.r.t weather)
response_data <- subset(data,select=c(registered,casual,cnt))
cnames <- colnames(response_data)
for (i in 1:length(cnames)){
 assign(paste0("gn",i), ggplot(aes_string(y = (cnames[i]),x=data$weather), data = response_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='Weather')+
        ggtitle(paste("Box plot of responded for",cnames[i])))
```

```
}
gridExtra::grid.arrange(gn1,gn2,gn3,ncol=1)

#### Correlation Matrix
numeric_index <- sapply(data,is.numeric)
numeric_data <- data[,numeric_index]
cnames <- colnames(numeric_data)
data.frame(cor(numeric_data))

#### Outlier analysis
outToNa <- function(vl, df) {
 for (i in vl) {
  outlier <- boxplot.stats(df[, i])$out
  df[, i] <- ifelse(df[, i] %in% outlier,
                    NA, df[, i])
 }
 return(df)
}
library(e1071)
library(DMwR)
outliereff <- function(i, df) {
 total = length(df[, i])
 par(mfrow = c(2, 2), oma = c(0, 0, 3,
                              0))
 boxplot(df[, i], main = "With Outliers")
 hist(df[, i], main = paste("With Outliers: Skewness = ",skewness(df[,i])),
     xlab = NA, ylab = NA, prob = TRUE)
 df <- outToNa(i, df)
 out <- sum(is.na(df[, i]))
 per <- round((out)/total * 100, 1)
 df <- knnImputation(df)
```

```
 boxplot(df[, i], main ="Without outliers")
 hist(df[, i], main = paste("Without Outliers: Skewness = ",skewness(df[,i])),
      xlab = NA, ylab = NA, prob = TRUE)

 title(paste("Effect of", out, "(", per,
             "%)", "Outliers on", colnames(df)[i],
             sep = " "), outer = TRUE)



}
var_list <- list(10,11,12)
for(vl in var_list){
 outliereff(vl,data)
}

data2 <- outToNa(var_list,data)
data2 <- knnImputation(data2)

#### Feature Scaling
normalized_data <- data2
numeric_data <- subset(data,select=c(casual,registered,cnt))
cnames <- colnames(numeric_data)
for (i in cnames){
 normalized_data[,i] = (normalized_data[,i] - min(normalized_data[,i]))/
   (max(normalized_data[,i])-min(normalized_data[,i]))
 }

### effect on 'feels like' temperature
gg <- ggplot(data, aes(temp, atemp, color=hum, size=windspeed)) +
 geom_point() +
 scale_colour_gradient(low="red",high="blue") +
```

```
 labs(title="Effects on 'Feels Like' Temperature") +
 labs(x="tempurature (normalized)") +
 labs(y="'feels like' tempurature (normalized)")
print(gg)
```

Variable Importance

```
feature <- select(data, season,mnth,workingday,weathersit,temp,atemp,hum,windspeed)
rf <- randomForest(feature, data1$cnt, ntree = 100, importance = T)
imp <- importance(rf, type = 1)
featureImportance <- data.frame(Feature=row.names(imp), Importance=imp[,1])

ggplot(featureImportance, aes(x=reorder(Feature, Importance), y=Importance)) +
 geom_bar(stat="identity", fill="#53cfff") +
 coord_flip() +
 theme_light(base_size=16) +
 xlab("") +
 ylab("Relative Importance for total count") +
 theme(plot.title   = element_text(size=18),
       strip.text.x = element_blank(),
       axis.text.x  = element_blank(),
       axis.ticks.x = element_blank())
```

Regression Tree model

```
data1 = subset(normalized_data,select=-c(holiday,weekday,atemp,casual,registered))
train_index <- sample(1:nrow(data1), 0.8 * nrow(data1))
train <- data1[train_index,]
test <- data1[-train_index,]

dt_model <- rpart(cnt ~ ., data=train, method = "anova")
fancyRpartPlot(dt_model)
prediction_DT <- predict(dt_model, test[,-9])
```

```
regr.eval(test[,9],prediction_DT,stats = c('mae','mape'))


#### Multiple Linear Regression
lm_model <-  lm(cnt ~ . ,data = train)
prediction_LM <- predict(lm_model, test[,-9])
regr.eval(test[,9],prediction_LM,stats = c('mae','mape'))
```

Appendix C - Python code

```
####Import necessary library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import seaborn as sns
from sklearn.cross_validation import train_test_split
import sklearn.tree
import fancyimpute
```

```
import statsmodels.api as sm
#### Loading data
os.getcwd()
df = pd.read_csv('C:/Users/User/Desktop/project2/day.csv')

#### Histogram of numeric variable
for col in df.columns:
    if df[col].dtypes == np.int64 or df[col].dtypes == np.float64:
        df.hist(column=col,facecolor='red')
#### Data type conversion
df['yr'] = df['yr'].astype('category')
df.weathersit = df.weathersit.astype('category')
df.season = df.season.astype('category')
df.holiday = df.holiday.astype('category')
df.workingday = df.workingday.astype('category')
df.weekday = df.weekday.astype('category')
df.mnth = df.mnth.astype('category')

#### Removing outliers for casual users
q75,q25 = np.percentile(df.casual,[75,25])
iqr = q75 - q25
minimum = q25 - (iqr * 1.5)
maximum = q75 + (iqr * 1.5)

df.loc[df.casual < minimum,:'casual'] = np.nan
df.loc[df.casual > maximum,:'casual'] = np.nan
df_subset = df.drop(['instant','dteday'], axis=1)
df_subset = pd.DataFrame(fancyimpute.KNN(k=3).fit_transform(df_subset),
columns=df_subset.columns)

#### Correlation Matrix
```

```
df.corr()


#### selecting significant features
df_subset = df_subset.drop(['atemp','holiday','weekday'], axis=1)
casual = df_subset.drop(['cnt','registered'], axis=1)
registered = df_subset.drop(['cnt','casual'], axis=1)
cnt = df_subset.drop(['casual','registered'], axis=1)
#### Splitting data into train and test by simple random sampling
train, test = train_test_split(cnt, test_size = 0.2)

#### Decision tree Regression Model
fit_DT = sklearn.tree.DecisionTreeRegressor(max_depth=5).fit(train.iloc[:,0:8],train.iloc[:,8])
prediction_DT = fit_DT.predict(test.iloc[:,0:8])
def MAPE(y_true, y_pred):
   mape = np.mean(np.abs((y_true-y_pred)/y_true))
   return mape
MAPE(test.iloc[:,8],prediction_DT)

#### Multiple Linear regression
lm_model = sm.OLS(train.iloc[:,8],train.iloc[:,:8]).fit()
prediction_LM = lm_model.predict(test.iloc[:,0:8])
MAPE(test.iloc[:,8],prediction_LM)
```

References

James, Gareth, Daniela Witten, Trevor Hastie, and Robert Tibshirani. 2013. *An Introduction to Statistical Learning*. Vol. 6. Springer.

Wickham, Hadley. 2009. *Ggplot2: Elegant Graphics for Data Analysis*. Springer Science & Business Media

"The Comprehensive R Archive Network". Retrieved 2018-08-06.
Chatfield, C. (1995). *Problem Solving: A Statistician's Guide* (2nd ed.). Chapman and Hall. ISBN 0412606305