Predicting number of bikes rented in each bike stations with a temporal granularity of one hour.

NAME: Jeevitha Sivadasan

INTRODUCTION

The London government has introduced a public bicycle hire scheme which allows public to rent bicycle for hourly bases. This scheme became very successful and even has advantanges like less pollution and less traffic over other transportation facilties. Due to increase in number of bicycles rented in each station the government finds it difficult to balance the bike sharing supply so that the public will get full advantage of this scheme. This project is to predict number of bikes that will be rented in each bike stations with a temporal granularity of one hour.

We are provided with three datasets:

1. **bike_journeys:** This dataset contains 1542844 bike journeys entries with 16 features such as, Journey_Duration, Journey_ID, End_Date, End_Month, End_Year,End_Hour,End_Minute, EndStationID,Start_Date, Start_Month, Start_Year, Start_Hour, Start_Minute, StartStationID. All these features has each journey complete information like journey starting date, time and end date, time.
2. **bike_stations:** This dataset contains 773 stations across London with 5 features such as, Station ID, Capacity - maximum capacity of bikes of the particular station latitude - latitude where the station is located longitude - longitude where the station is located Station_Name - The name of the station
3. **London Census:** This dataset contains 625 observations with 23 features. This dataset has the information of london WardCode, Ward Name, Borough, NESW, AreaSqKm, Latitude and longitude associated with the centre of the ward, **IncomeScor**: This is a proportion of deprived are with low income. The more deprived an area, the higher the score. **LivingEnSc**: This is the quality of the local environment. The higher the score, the more deorived the area. **NoEmployee**: The number of people having an occupation in that particular area.**PopDen**: This is density of the population. Calculated by dividing population and the surface of the area inthe ward. **BornUK**: This is total number of people who are born in UK **NotBornUK**: This is total number of people who are not born in UK **NoCTFToH**: number of properties in the council tax band F-H **NoDwelling**:The total number of properties in the particular ward. **Noflats**: Total number of flats in this particular ward. **NoHouses**: Total number f houses in the particular ward. **NoOwndDwel**: Total number of Owned properties in each ward. **MedHPrice**: This is the median house price in each ward.

Goal

The goal is to predict number of bike rented in bike stations with the temporal granularity of one hour time slot and to understand which are the factors that are associated with the demand of the rented bikes in London.

Challenge

As we are provided with 3 dataset we are need to merge all three dataset to one for the further implementation of the model. But London census dataset does not have a comman feature to merge with outher two dataset whereas, the bike_stations and bike_jouneys dataset has Station_ID and StartStationID column respectively in comman.

Solution

In Bike_Station And london census dataset we have 2 features which provides the location of the station ID and the ward code respectively: Latitude and longitude. With these two features we are finding the distance between the station and the Ward code in the area. The minimum distance between there two location is noted and then the nearest ward code is added to the bike station dataset.

Another challenge in the project is that there are other factors that might affect the number of bikes rented in each area such as climate, and weekdays where the dataset which is provided does not have the information of week of the days.

HYPOTHESIS

Hypothesis 1: The number of bike rented during the weekends will be higher.

Hypothesis 2: The number of bike rented during the peak time is higher.

Hypothesis 3: Higher bikes retend in the less deprived area.

Hypothesis 4: Higer Bikes are rented where there is more green space area.

Hypothesis 5: Higher bikes are rented where there is more population.

Hypothesis 6: Higher bikes are rented where more number of non-born UK people live.

Hypothes 7: Higher bikes are rented where there is higher number of properties councel tax.

Hypothesis 8: Higher bikes are rented where there is more number of flats and houses.

Hypothesis 9: Higher bikes are rented where there is more number of owned properties.

Hypothesis 10: Higher bikes are rented where house price is higher.

Hypothesis 11: Higher bikes are rented where there is more employed people live.

Metrices

1. **weekdays** (Hypothesis 1): This is days of the week when the bike is rented. This feature is extracted using <Start_Date, Start_Month, Start_Year> in the bike_journeys dataset. weekdays() method to get the days of the week and added the weekdays column to the bike_journeys dataset. This column is converted to numerical feature using label encoding method.
2. **peak** (Hypothesis 2): This variable is created by assigning 0 and 1 to off-peak and peak time respectively. Peak team is 6:30 to 9:30 and 16:00 to 19:00. Therefore the duration which falls between this time is set to **1** and other values are **0**. The time bike is rented is extracted using <Start_Hour, Start_Minute> in bike_journey dataset.

3. **IncomeScore**(Hypothesis 3): This is income score of population in particular area. The more deprived a area, higher income score.
4. **LivingEnSc**(Hypothesis 3): This is quality of the environment. If the area is more deprived then this score will be higher.
5. **RatioEmployee**(Hypothesis 11): This is the ratio of employee in the area. It is calculated using **_(bike_j N oE m ployee/ (bik jPopDen*bike_j$AreaSqKm))_**.
6. **GrenSpace**(Hypothesis 4): This is percentage of green space in a particular area in a ward.
7. **PopDen**(Hypothesis 5): This is population divided by the surface of the ward area.
8. **NonUKBorn**(Hypothesis 6): This is ratio of total number of non Uk people living in a area.
9. **NoCTFtoH**(Hypothesis 7): This is number of properties under council tax.
10. **NoFlatHouse**(Hypothesis 8): This is total number of flats and houses in a area. This calculated by adding NoFlats and NoHouses in london_cesus data.
11. **NoOwndDwel**(Hypothesis 9): This is number of owned propertied in a area.
12. **MedHPrice**(Hypothesis 10): This is median house price in a area.

Importing all three datasets

We are importing all the three dataset using read.csv and saving it as bike_journeys, bike_stations and london_census.

Hide

```
##UPLOADING THREE DATASETS

bike_journeys = read.csv("bike_journeys.csv")
bike_stations = read.csv("bike_stations.csv")
london_census = read.csv("London_census.csv")
```

Hide

```
##PRINTING NUMBER OF OBSERVATIONS AND FEATURES IN ALL THREE DATASETS

##Number of observations and features in
bike_journey dim(bike_journeys)
```

```
[1] 1542844      14
```

Hide

```
##Number of observations and features in
bike_stations dim(bike_stations)
```

```
[1] 773   5
```

Hide

```
##Number of observations and features in
london_census dim(london_census)
```

```
[1] 625  20
```

Creating the metrics

MERGING ALL THREE DATASETS

Before merging all three datasets we are creating a time slot of one hour in bike_journeys. To do this the following steps are executed:

1. Creating a new column 'Start_Time' with pasting start_hour and start_minute values together seperated with colon ':'.
2. To create a time slot we are using as.POSIXlt and creating a time slot of one hour in 24hours format.

Hide

```
## creating time slot
# merging hour and minutes
column #library(dplyr)
#cols <- select(bike_journeys, Start_Hour, Start_Minute)
bike_journeys$Start_time<-paste(bike_journeys$Start_Hour, bike_journeys$Start_Minute, sep = ':')

## creating time slot of one hour
a <- as.POSIXlt(bike_journeys$Start_time, '%H:%M', tz = '')
time_slot <- paste(format(a, '%H:00'), format(a+3600, '%H:00'), sep='-')

## binding it to bike journey data
bike_journeys <- cbind(bike_journeys,time_slot)
str(bike_journeys)
```

```
'data.frame':   1542844 obs. of  16 variables:
 $ Journey_Duration: num  2040 1800 1140 420 1200 1320 720 720 540 960 ...
 $ Journey_ID      : int  953 12581 1159 2375 14659 2351 7252 9782 13500 11205 ...
 $ End_Date        : int  19 19 15 14 13 14 17 17 15 15 ...
 $ End_Month       : int  9 9 9 9 9 9999    9 ...
 $ End_Year        : int  17 17 17 17 17 17 17 17 17 17 ...
 $ End_Hour        : int  18 15 17 12 19 14 17 17 13 16 ...
 $ End_Minute      : int  021116    33 53 12 12 42 3 ...
 $ End_Station_ID  : int  478 122 639 755 605 514 484 484 367 350 ...
 $ Start_Date      : int  19 19 15 14 13 14 17 17 15 15 ...
 $ Start_Month     : int  9 9 9 9 9 9999    9 ...
 $ Start_Year      : int  17 17 17 17 17 17 17 17 17 17 ...
 $ Start_Hour      : int  17 14 16 12 19 14 17 17 13 15 ...
 $ Start_Minute    : int  26 51 4291331003347...
 $ Start_Station_ID: int  251 550 212 163 36 589 478 478 153 396 ...
 $ Start_time      : chr  "17:26" "14:51" "16:42" "12:9" ...
 $ time_slot       : Factor w/ 24 levels "00:00-01:00",..: 18 15 17 13 20 15 18 18 14 16 ...
```

Creating weekdays variable

The code below gets the days of the week using weekdays() method and this is added to the bike_journey dataset.

Hide

```
#finding the week days
bike_journeys$Date<-paste(
  bike_journeys$Start_Date,
  bike_journeys$Start_Month,
  bike_journeys$Start_Year,
  sep = '-')
weekdays <- paste(weekdays(as.Date(bike_journeys$Date)))
```

Hide

```
#combining the weekdays column to bike_journey
bike_journeys <- cbind(bike_journeys,weekdays)
str(bike_journeys)
```

```
'data.frame':   1542844 obs. of  18 variables:
 $ Journey_Duration: num  2040 1800 1140 420 1200 1320 720 720 540 960 ...
 $ Journey_ID      : int  953 12581 1159 2375 14659 2351 7252 9782 13500 11205 ...
 $ End_Date        : int  19 19 15 14 13 14 17 17 15 15 ...
 $ End_Month       : int  9 9 9 9 9 9999    9 ...
 $ End_Year        : int  17 17 17 17 17 17 17 17 17 17 ...
 $ End_Hour        : int  18 15 17 12 19 14 17 17 13 16 ...
 $ End_Minute      : int  021116    33 53 12 12 42 3 ...
 $ End_Station_ID  : int  478 122 639 755 605 514 484 484 367 350 ...
 $ Start_Date      : int  19 19 15 14 13 14 17 17 15 15 ...
 $ Start_Month     : int  9 9 9 9 9 9999    9 ...
 $ Start_Year      : int  17 17 17 17 17 17 17 17 17 17 ...
 $ Start_Hour      : int  17 14 16 12 19 14 17 17 13 15 ...
 $ Start_Minute    : int  26 51 4291331003347...
 $ Start_Station_ID: int  251 550 212 163 36 589 478 478 153 396 ...
 $ Start_time      : chr  "17:26" "14:51" "16:42" "12:9" ...
 $ time_slot       : Factor w/ 24 levels "00:00-01:00",..: 18 15 17 13 20 15 18 18 14 16 ...
 $ Date            : chr  "19-9-17" "19-9-17" "15-9-17" "14-9-17" ...
 $ weekdays        : Factor w/ 7 levels "Friday","Monday",..: 6 6 5 7 6 7 4 4 5 5 ...
```

Creating peak variable

peak variable is created by setting peak hours to 1 and other as 0. peak hours: 6:30 to 9:30 off-peak hours: 4:00 to 7:00

Hide

```
#creating a new column with Start_hour and Start_minute separated with
'.' bike_journeys$START_TIME<-paste(bike_journeys$Start_Hour,
                                   bike_journeys$Start_Minute,
                                   sep = '.')

#creating new column peak_hours and giving logical condition with 'AND''OR'.
#if the value statisfies the condition then it is set to true or false
peak_hours <- bike_journeys$START_TIME > 6.30 & bike_journeys$START_TIME < 9.30 |
  bike_journeys$START_TIME > 16 & bike_journeys$START_TIME < 19

#Converting to numeric value
peak <- as.numeric(peak_hours)
```

Hide

```
#adding the column to bike_journeys
bike_journeys <- cbind(bike_journeys,peak)
colnames(bike_journeys)
```

```
 [1] "Journey_Duration" "Journey_ID"       "End_Date"         "End_Month"        "End_Year"
"End_Hour"
 [7] "End_Minute"       "End_Station_ID"   "Start_Date"       "Start_Month"      "Start_Year"
"Start_Hour"
[13] "Start_Minute"     "Start_Station_ID" "Start_time"       "time_slot"        "Date"
"weekdays"
[19] "START_TIME"       "peak"
```

We are removing few columns from the bike_journeys dataset.

Hide

```
bike_journeys$Journey_ID = NULL
bike_journeys$End_Date = NULL
bike_journeys$End_Month = NULL
bike_journeys$End_Year = NULL
bike_journeys$End_Hour = NULL
bike_journeys$End_Minute = NULL
bike_journeys$End_Station_ID = NULL
bike_journeys$START_TIME = NULL
bike_journeys$Start_time = NULL
bike_journeys$time_slot = NULL
colnames(bike_journeys)
```

```
 [1] "Journey_Duration" "Start_Date"      "Start_Month"     "Start_Year"      "Start_Hour"
"Start_Minute"
 [7] "Start_Station_ID" "Date"            "weekdays"        "peak"
```

Merging bike_journeys and bike_stations

Now, we are merging bike_journeys and bike_stations dataset by “Staion_ID”. In bike_journeys dataset the Start_Station_ID is converted to Station_ID so that we can merge with the Station_ID column in the bike_stations dataset.

Hide

```
#renaming Start_station_id as "Station_ID"
colnames(bike_journeys)[7] <- "Station_ID"

## merging bike station and bike journeys datasets
bike <- merge(bike_journeys, bike_stations, by = "Station_ID", all.x = TRUE)
colnames(bike)
```

```
 [1] "Station_ID"       "Journey_Duration" "Start_Date"       "Start_Month"      "Start_Year"
"Start_Hour"
 [7] "Start_Minute"     "Date"             "weekdays"         "peak"             "Capacity"
"Latitude"
[13] "Longitude"        "Station_Name"
```

Merging bike dataset and london census

1. Creating two new dataframe: a and b, which has {WardCode,Lat, lon} and {Station_ID, latitude, longitude} respectively.
2. By using distm() function from grosphere library we are calculating the distance between two locations and saving it as a matrix named ‘distance’. We have used haversine distance formula to calculate the distance.
3. By using matrixStats rowMins functions we are extracting the minimum distance row-wise from the distance matrix.
4. We are storing the index value of each minimum value.
5. Now we are creating a new data frame ‘code_id’ which has the Station_ID and the WardCode which is in the repective index value that are stores earlier in the index matrix.
6. Merging the code_id and london _census by the “WardCode” variable.
7. Finally, the census dataset and the bike dataset is merged by the Station_ID and the final dataframe is stored as bike_j

Hide

```
## creating new data frame calculating distance

#selecting Wardcode, lat, lon from the london census data
set a <- select(london_census, "WardCode", "lat", "lon")

#selecting station_Id, Latitude and longitude from the Bike_station Dataset.
b <- select(bike_stations, Station_ID, Latitude, Longitude)
```

Calculating the distance

Hide

```
library(geosphere)
#calculating distance between the two location
distance<- distm(cbind((b$Longitude),(b$Latitude)),
                 cbind((a$lon),(a$lat)),
                 fun = distHaversine)

#gets minimum value row-wise
min <- matrixStats::rowMins(distance)

#stores index of minimum value
index <-(distance == min) %*% 1:ncol(distance)

#Creating new dataframe
code_id <- data.frame(Station_ID = b$Station_ID, WardCode = a$WardCode[index])

## merging
census <- merge(code_id, london_census, by = "WardCode", all.x = TRUE)

## all the data

bike_j <- merge(bike, census, by= "Station_ID", all.x = TRUE)
colnames(bike_j)
```

```
 [1] "Station_ID"      "Journey_Duration" "Start_Date"      "Start_Month"     "Start_Year"
"Start_Hour"
 [7] "Start_Minute"    "Date"             "weekdays"        "peak"            "Capacity"
"Latitude"
[13] "Longitude"       "Station_Name"     "WardCode"        "WardName"        "borough"
"NESW"
[19] "AreaSqKm"        "lon"              "lat"             "IncomeScor"      "LivingEnSc"
"NoEmployee"
[25] "GrenSpace"       "PopDen"           "BornUK"          "NotBornUK"       "NoCTFtoH"
"NoDwelling"
[31] "NoFlats"         "NoHouses"         "NoOwndDwel"      "MedHPrice"
```

Finally, all the three dataset has been merged successfully.

Adding new metrics

Hide

```
##Adding new metrics

bike_j$RatioEmployee= (bike_j$NoEmployee/(bike_j$PopDen*bike_j$AreaSqKm)) #(Hypothesis 11)
bike_j$RatioNotBornUK= (bike_j$NotBornUK/(bike_j$BornUK+bike_j$NotBornUK))#(Hypothesis 6)
bike_j$NoFlatsHouse = ((bike_j$NoFlats) + (bike_j$NoHouses))
colnames(bike_j)
```

```
 [1] "Station_ID"       "Journey_Duration" "Start_Date"       "Start_Month"     "Start_Year"
"Start_Hour"
 [7] "Start_Minute"     "Date"             "weekdays"         "peak"            "Capacity"
"Latitude"
[13] "Longitude"        "Station_Name"     "WardCode"         "WardName"        "borough"
"NESW"
[19] "AreaSqKm"         "lon"              "lat"              "IncomeScor"      "LivingEnSc"
"NoEmployee"
[25] "GrenSpace"        "PopDen"           "BornUK"           "NotBornUK"       "NoCTFtoH"
"NoDwelling"
[31] "NoFlats"          "NoHouses"         "NoOwndDwel"       "MedHPrice"       "RatioEmployee"
"RatioNotBornUK"
[37] "NoFlatsHouse"
```

Hide

```
## Removing unwanted
features bike_j$Station_ID =
NULL bike_j$Latitude = NULL
bike_j$Longitude = NULL
bike_j$lon = NULL bike_j$lat
= NULL bike_j$Station_Name =
NULL bike_j$WardCode = NULL
bike_j$WardName = NULL
bike_j$borough = NULL
bike_j$AreaSqKm = NULL
bike_j$Capacity = NULL
bike_j$Start_Date = NULL
bike_j$Start_Minute = NULL
bike_j$Start_time = NULL
bike_j$BornUK = NULL
bike_j$NotBornUK = NULL
bike_j$NoEmployee = NULL
bike_j$Date = NULL
bike_j$NESW = NULL
bike_j$NoFlats = NULL
bike_j$NoHouses = NULL
str(bike_j)
```

```
'data.frame':   1542844 obs. of  17 variables:
 $ Journey_Duration: num  1385 1080 532  4081200 ...
 $ Start_Month     : int  8 9 8 8 9 8 9  8 88 ...
 $ Start_Year      : int  17 17  17 17 17 1717 17 17 17 ...
 $ Start_Hour      : int  108681810813       88...
 $ weekdays        : Factor w/ 7 levels "Friday","Monday",..: 2 6 7 6 1 2 2 4 3 2 ...
 $ peak            : num  0 1 1 1 1 0 1  0 11 ...
 $ IncomeScor      : num  0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 0.21 ...
 $ LivingEnSc      : num  51 51 51 51 51 ...
 $ GrenSpace       : num  9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 9.3 ...
 $ PopDen          : num  12778  12778 1277812778 12778 ...
 $ NoCTFtoH        : num  24 24  24 24 24 2424 24 24 24 ...
 $ NoDwelling      : int  5663 5663 5663 5663 5663 5663 5663 5663 5663 5663 ...
 $ NoOwndDwel      : int  1565 1565 1565 1565 1565 1565 1565 1565 1565 1565 ...
 $ MedHPrice       : int  455000 455000 455000 455000 455000 455000 455000 455000 455000 455000
...
 $ RatioEmployee   : num  3.82 3.82 3.82 3.82 3.82 ...
 $ RatioNotBornUK  : num  0.385  0.385 0.3850.385 0.385 ...
 $ NoFlatsHouse    : int  5733 5733 5733 5733 5733 5733 5733 5733 5733 5733 ...
```

All the metrices are grouped according to the start_hour and a new column ‘Count’ is created. Count is out dependent variable.

Hide

```
library(dplyr)
BIKE = bike_j %>%
  group_by(Start_Hour,
           weekdays,
           peak,
           IncomeScor,
           LivingEnSc,
           RatioEmployee,
           GrenSpace,
           PopDen,
           RatioNotBornUK,
           NoCTFtoH,
           NoFlatsHouse,
           NoOwndDwel,
           MedHPrice) %>%
  dplyr::summarise(Count=n())
BIKE = as.data.frame(BIKE)
str(BIKE)
```

```
'data.frame':   18641 obs. of   14 variables:
 $ Start_Hour    : int  0000    0 0 0 0 00...
 $ weekdays      : Factor w/  7 levels "Friday","Monday",..: 1 1       1111111 1 ...
 $ peak          : num  0000    0 0 0 0 00...
 $ IncomeScor    : num  0.01  0.03 0.05 0.05  0.05 0.05 0.05 0.05 0.06 0.06 ...
 $ LivingEnSc    : num  43.9 49  37.5 48 49.8 ...
 $ RatioEmployee : num  3.011 1.184 0.658 0.933 0.676 ...
 $ GrenSpace     : num  69.1 5.8 34.5 27.9 43.1 13.3 6.1 10.8 18.6 2.1 ...
 $ PopDen        : num  2583 16333 7842 9056 8056 ...
 $ RatioNotBornUK: num  0.596 0.632 0.314 0.528 0.448 ...
 $ NoCTFtoH      : num  95.7  81.2 44.6 76.6  74.1 71.5 66.1 81.1 33.7 57.1 ...
 $ NoFlatsHouse  : int  5438  5645 6851 4550  4166 5105 6527 5840 5530 7492 ...
 $ NoOwndDwel    : int  1813  2050 3685 1884  1698 1792 1759 1618 1868 2054 ...
 $ MedHPrice     : int  1750000  1000000 525000 1100000 895000 863750 680000 1200000 465000 67500
0 ...
 $ Count         : int  50133   15 15 30 20  30 58 31 ...
```

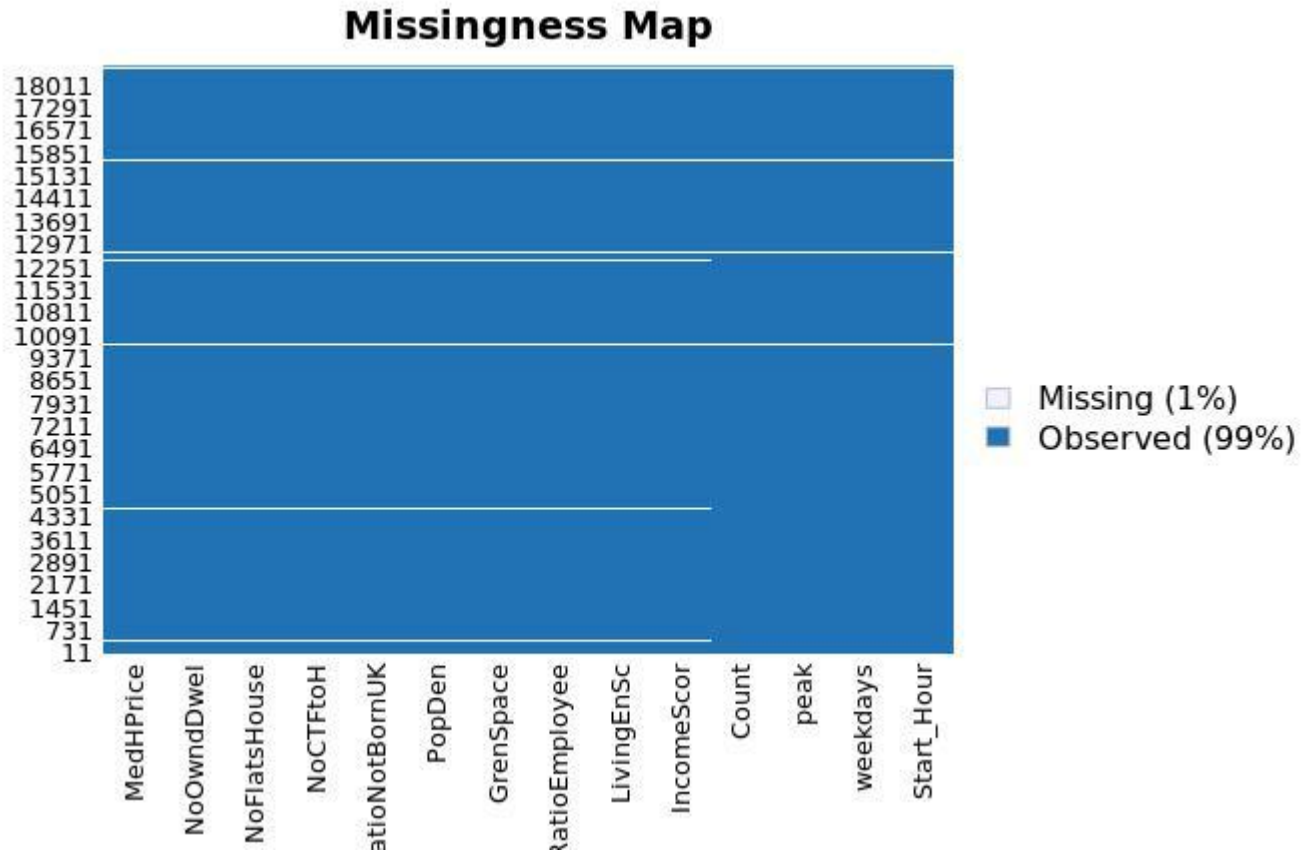
We have created the metrices data frame.

Checking for missing values

To check whether there is missing value in the dataset we are using missmap() function from Amelia.

Hide

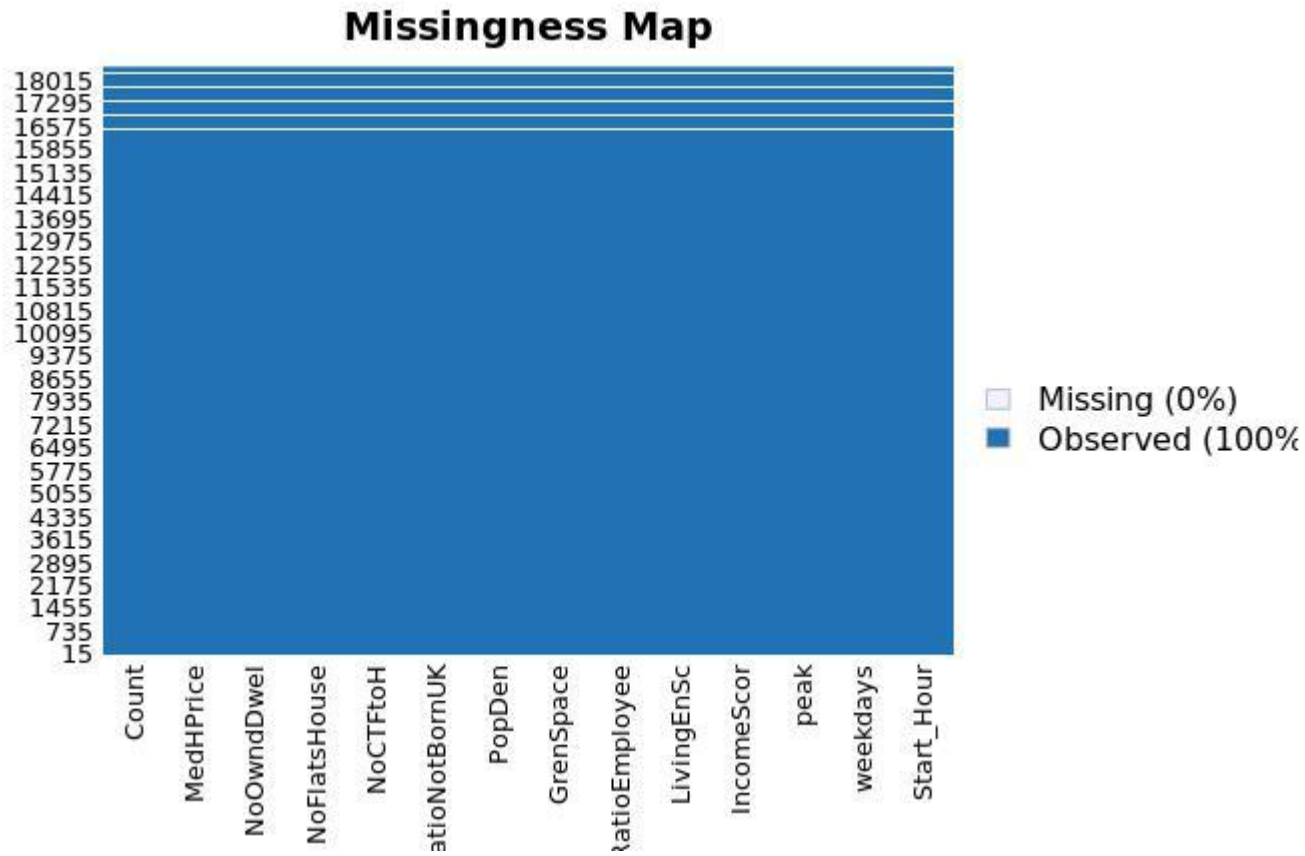
```
library(Amelia)
missmap(BIKE)
```



The above map shows that there is 1% missing value in the dataset. Now we are going to remove the missing values using na.omit() function.

Hide

```
#Removing the na values
BIKE_1 <- na.omit(BIKE)
missmap(BIKE_1)
```



All the missing values are removed. Lets check the number of rows removed in the dataset.

Checking the dimensions of the datasets before and after removing the na values

Hide

```
print(dim(BIKE))
```

```
[1] 18641    14
```

Hide

```
print(dim(BIKE_1))
```

```
[1] 18465    14
```

We can see that 176 rows has been excluded from the dataset.

Data exploration

Number of bikes rented in days of the weeks

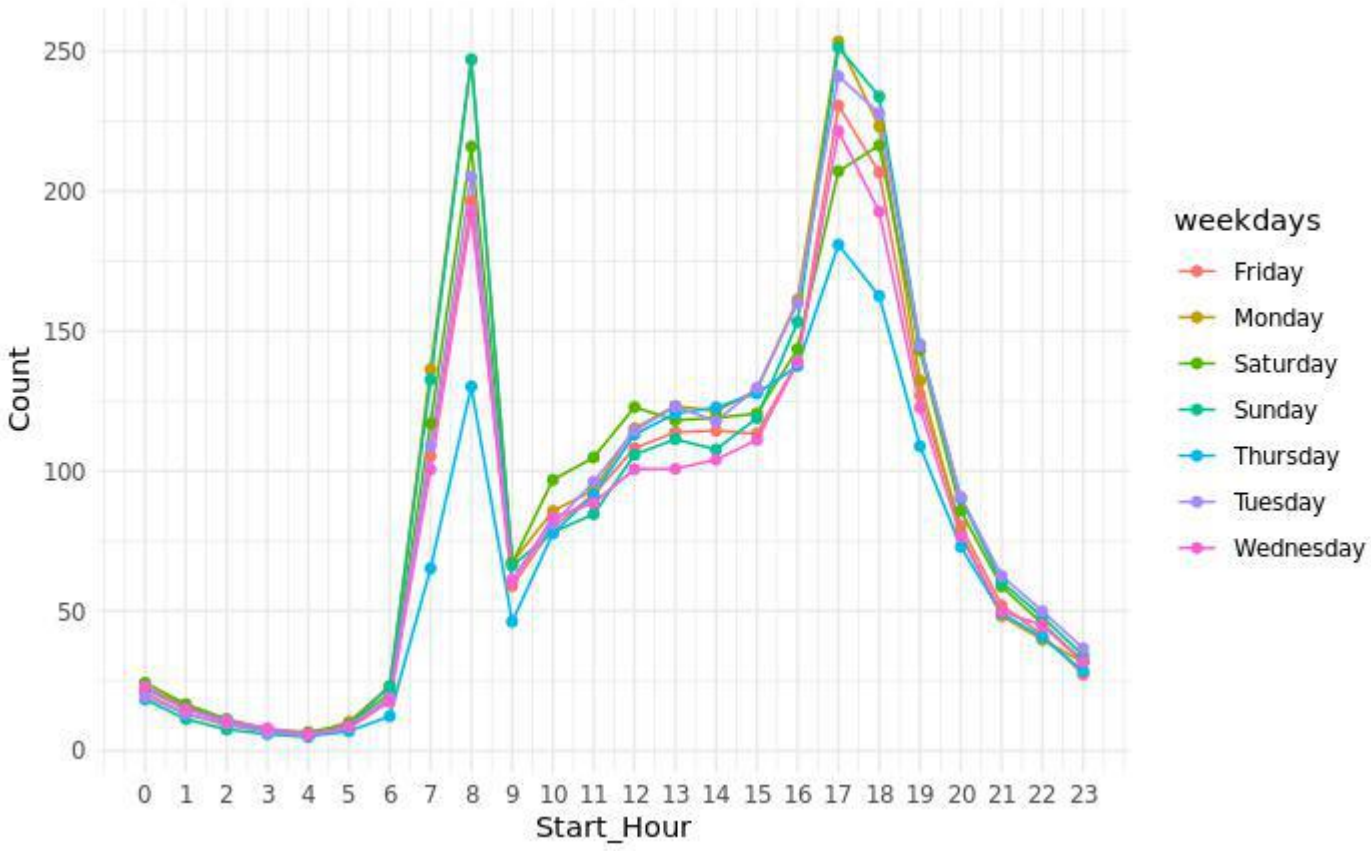
Hide

```
#library(ggplot2)
#library(lubridate)
#library(scales)
#library(plyr)
#library(readr)
#creating a summary of weekdays
weekdays_summary <- ddply(BIKE_1,.(weekdays,Start_Hour),summarise, Count = mean(Count))

#Creating a line plot
weekplot <- ggplot(BIKE_1, aes(x = Start_Hour, y = Count, colour =
  weekdays))+ geom_line(weekdays_summary, mapping = aes(group =
  factor(weekdays)))+ geom_point(weekdays_summary, mapping = aes(group =
  factor(weekdays))) + scale_x_continuous("Start_Hour", breaks = 0:23) +
  theme_minimal()
```

Hide

```
##week plot
weekplot
```



From the above plot we can see that only on thursday the number of bikes rented is less whereas on the other days we can see a similar pattern and on weekends we can see more number of bikes rented.

Next we are plotting peak hours count line graph,

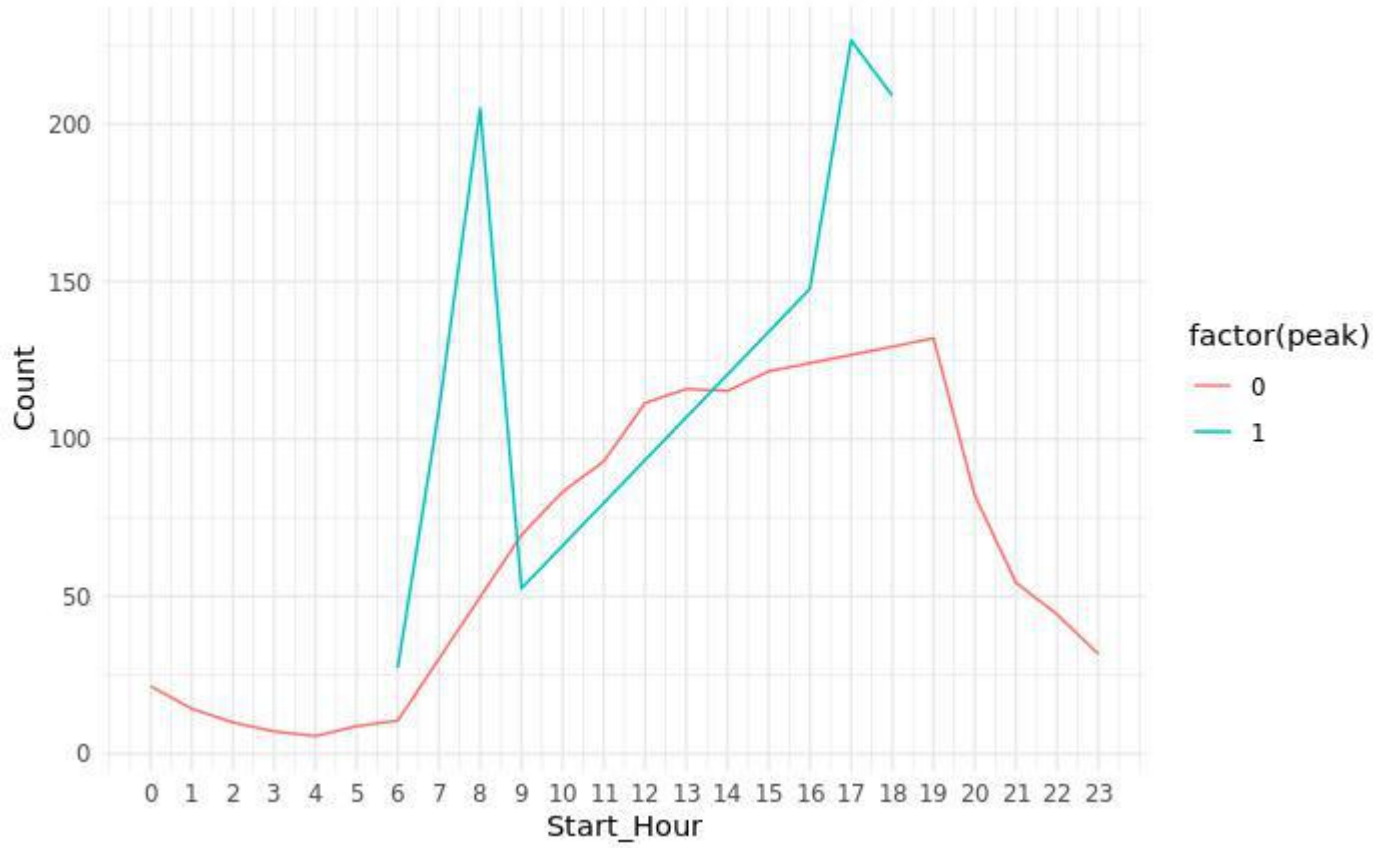
Hide

```
##creating summary for peak
peak_summary <- ddply(BIKE_1,.(peak,Start_Hour),summarise, Count = mean(Count))

peak_plot <- ggplot(BIKE_1, aes(x = Start_Hour, y = Count, color =
  factor(peak)))+ geom_line(peak_summary, mapping = aes(group = peak))+
  scale_x_continuous("Start_Hour", breaks = 0:23) +
  theme_minimal()
```

Hide

```
peak_plot
```



Above graph shows that in the peak hours more number of bikes are rented as compared to the off-peak hours. 0 is the off-peak hour and 1 is peak hour.

Ploting median house price with the count of bikes rented

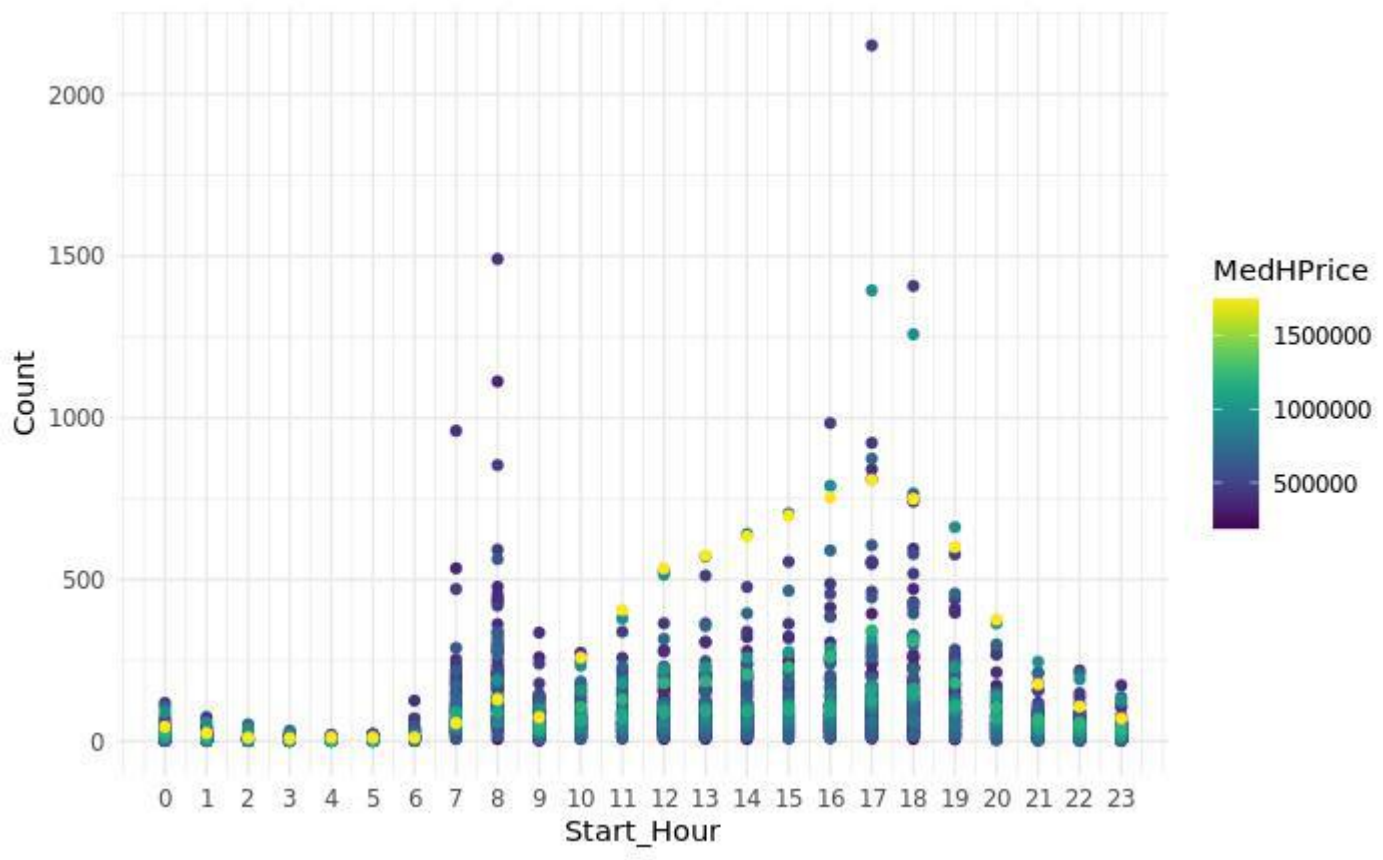
Hide

```
##Creating a summary of median house price with respect to start_hour & calculating the
count #library(viridis)
     mhouse_summary <- ddply(BIKE_1,.(MedHPrice,Start_Hour),
                             summarise, Count = mean(Count))

mhouse_plot <- ggplot(BIKE_1, aes(x = Start_Hour, y = Count, color =
  MedHPrice))+ geom_point(mhouse_summary, mapping = aes(group = MedHPrice))+
  scale_color_viridis(option = "D")+
  scale_x_continuous("Start_Hour", breaks = 0:23) +
  theme_minimal()
```

Hide

```
mhouse_plot
```



From the plotting we can see that bikes are rented more in the area where median house price is less.

Pre-processing

We are applying label Encoder to the weekends in the dataset. This converts the categorical variable to a unique interger based on the provided order. We are doing this by creating a new character vector with days of the week in order. This is the order that we are going to assign for the encoding.

Then, we are using factor() fuction to store the vector of integer values with corresponding to the weeks character. And with the factor() method we are setting the level parameters as ‘weeks’ and converting the factor to number using as.numeric(). Finally the values are stored in the BIKE_1 dataset.

Hide

```
#days of the week order
weeks <- c("Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday",
"Sunday") #creating factor
factors <- factor(weeks)
as.numeric(factors)
```

```
[1]2675134
```

Hide

```
#label encoding storing the converted numbers in the BIKE_1 weekdays
column BIKE_1$weekdays <- factor(BIKE_1$weekdays, level = weeks)
BIKE_1$weekdays <- as.numeric(BIKE_1$weekdays)
```

Splitting dataset using train-test split

The dataset is splitted into train and test data. 75% of the data is training data and 25% is testing data.

Hide

```
#spliting dataset into train and test.
set.seed(123)
n <- nrow(BIKE_1)
BIKE_2 <- BIKE_1[sample(n),]
trainIndex = sample(1:nrow(BIKE_1), 0.75*nrow(BIKE_2))
train = BIKE_2[trainIndex,]
test= BIKE_2[-trainIndex,]
head(train)
```

	Start_Hour <int>	weekd… <dbl>	p… <dbl>	IncomeS… <dbl>	LivingEnSc <dbl>	RatioEmployee <dbl>	GrenSp… <dbl>	PopDen <dbl>
14310	18	5	1	0.22	33.15	1.1872463	37.0	8366.7
1168	1	4	0	0.24	47.25	1.9925955	27.0	10384.6
7709	9	7	1	0.25	48.95	0.8083646	20.7	11958.3
9009	10	3	0	0.07	43.77	0.8342876	8.5	14583.3
9897	12	5	0	0.21	52.69	2.2352904	9.4	10818.2
14597	18	7	1	0.14	61.93	0.5082873	10.0	18100.0

6 rows | 1-9 of 14 columns

Splitting the independent variable and dependent variable.

Hide

```
train_tibble = as_tibble(train)
test_tibble = as_tibble(test)
x_train <- as.matrix( train_tibble[, 1:13])
y_train <- as.matrix(train_tibble %>% select(Count))
x_test <- as.matrix( test_tibble[, 1:13])
y_test <- as.matrix(test_tibble %>% select(Count))
print('Independent Variable:')
```

```
[1] "Independent Variable:"
```

Hide

```
print(colnames(x_train))
```

```
 [1] "Start_Hour"     "weekdays"       "peak"           "IncomeScor"                    "Ratio
Employee"  "GrenSpace"                                                  "LivingEnSc
 [8] "PopDen"         "RatioNotBornUK" "NoCTFtoH"       "                               "MedHP
rice"                                                                  "NoFlatsHouse"
                                                                       "NoOwndDwel
                                                                       "
```

Hide

```
print('Dependent Variable')
```

```
[1] "Dependent Variable"
```

Hide

```
print(colnames(y_train))
```

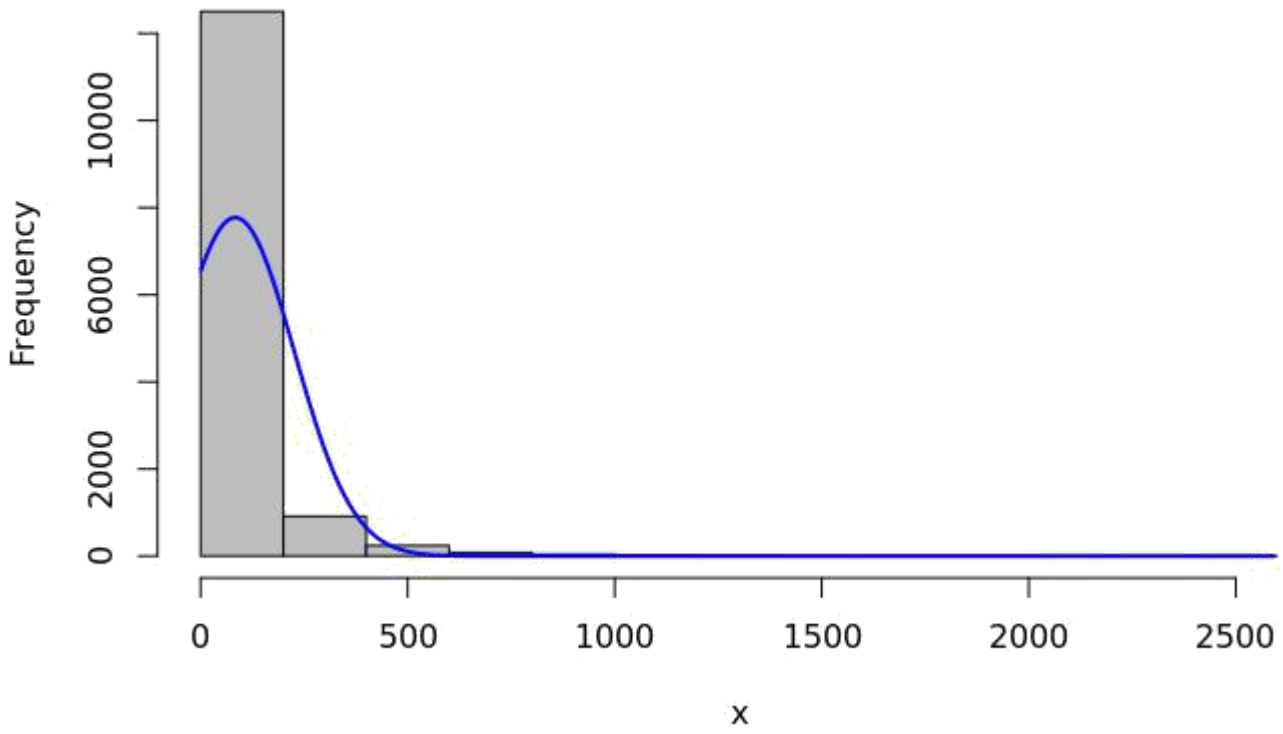
```
[1] "Count"
```

Looking at the dependent Variable

We are plotting a histogram to understand the distribution of the independent variable. We are using plotNormalHistogram() function from rcompanion package.

Hide

```
library(rcompanion)
plotNormalHistogram(y_train)
```



The above plot shows that the variable is skewed and not normally distributed.Multivariate normality is one of the assumption of regression problem. The skewness has impact on the model like, the skewed data will affect the parameter estimation because it is based on the minimization of squared error. Therefore, we are scaling the y_train dataset.

We are applying Log transformation because: 1. When we apply log transformation it returns a positive value. This makes sense because the count of bike rent should be an positive value but not an negative. 2. When the variables are skewed there is high chance that the model will overfit. By normalizing the value to more normally-shaped bell curve we reduce overfitting issue in the model.

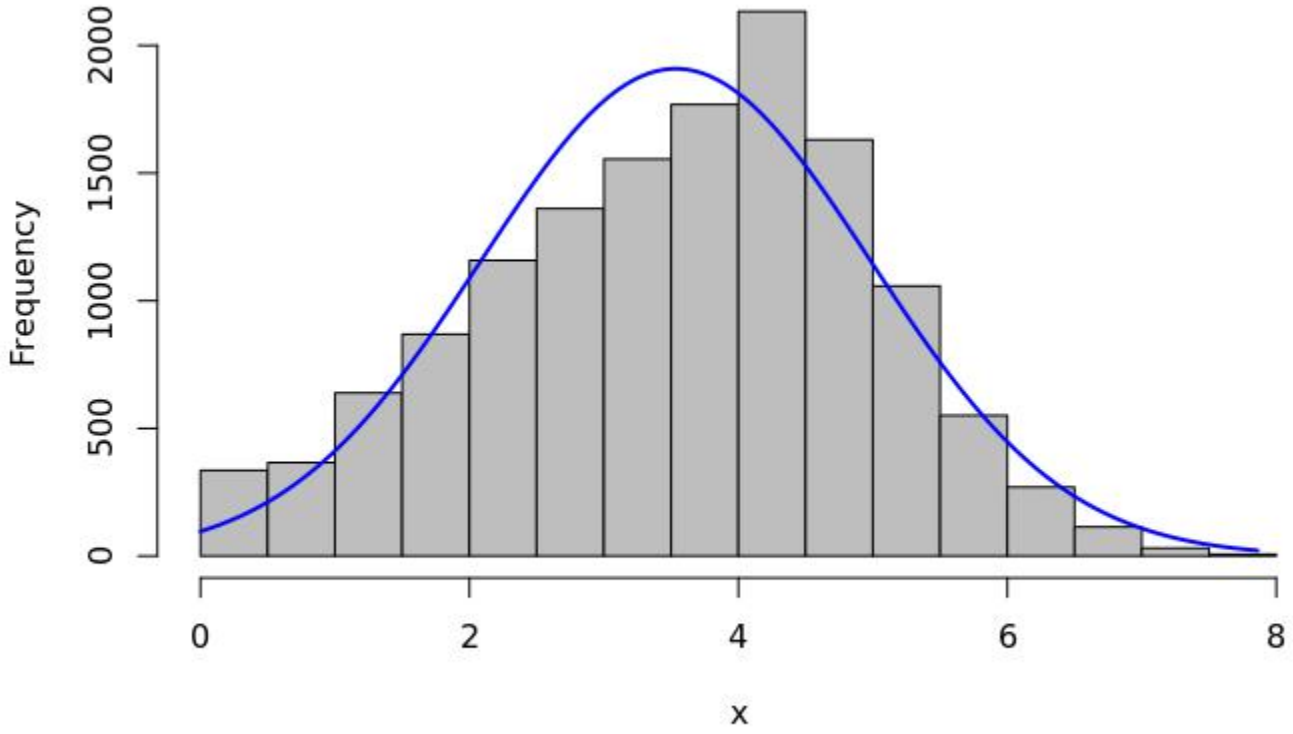
log transformation - y_train and y_test

Hide

```
y_train_scale <- as.data.frame(log(y_train))
y_test_scale <- as.data.frame(log(y_test))
```

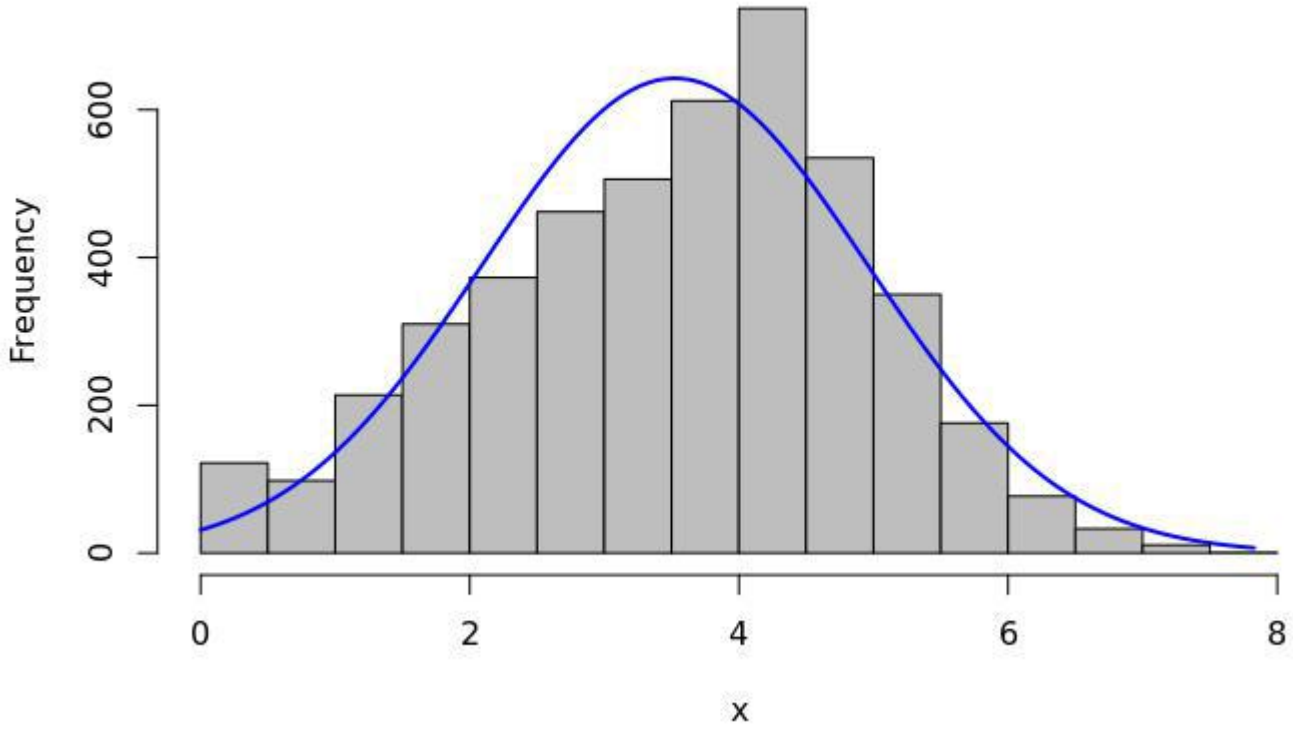
Hide

```
plotNormalHistogram(y_train_scale)
```



Hide

```
plotNormalHistogram(y_test_scale)
```



Now we can see that y_train and y_test is normally distributed.

Looking at the independent Variable

The assumptions in regression problem is: 1. Independent variable must have less collinearity or no correlation at all. 2. Multicollinearity is a problem because it produces very unstable coefficient and the interpretation of the model will be more difficult. 3. There must be a bell curve distribution of the variables.

Looking at the summary of the variables

Hide

```
summary(x_train)
```

```
    Start_Hour      weekdays          peak           IncomeScor      LivingEnSc   RatioEmployee
GrenSpace
 Min.    : 0.00  Min.   :1.000   Min.   :0.0000   Min.   :0.0100  Min.   :22.05   Min.    : 0.13
21   Min.    : 0.00
 1st Qu.: 6.00   1st Qu.:2.000   1st Qu.:0.0000   1st Qu.:0.1100  1st Qu.:41.64   1st Qu.: 0.33
33   1st Qu.: 8.90
 Median :11.00   Median :4.000   Median :0.0000   Median :0.1900   Median :46.51   Median : 0.53
16   Median :13.50
 Mean    :11.42  Mean   :3.992   Mean   :0.2725   Mean   :0.1975   Mean   :46.94   Mean    : 1.97
48   Mean    :18.22
 3rd  Qu.:17.00  3rd Qu.:6.000   3rd Qu.:1.0000   3rd Qu.:0.2900  3rd Qu.:51.91   3rd Qu.: 1.36
17    3rd Qu.:26.90
 Max.    :23.00  Max.   :7.000   Max.   :1.0000   Max.   :0.4400   Max.   :68.06   Max.    :50.55
40   Max.    :69.10
     PopDen      RatioNotBornUK      NoCTFtoH      NoFlatsHouse    NoOwndDwel     MedHPrice
 Min.    : 2312  Min.   :0.2888   Min.    : 0.2   Min.    : 3430  Min.    : 563  Min.    : 188000
 1st Qu.: 9767   1st Qu.:0.3904   1st Qu.: 8.4    1st Qu.: 5202   1st Qu.:1418  1st Qu.: 331500
 Median :12636   Median :0.4379   Median :24.5   Median  : 5745  Median  :1709  Median : 432500
 Mean    :12873  Mean   :0.4495   Mean   :28.1   Mean    : 5987  Mean    :1761  Mean    : 503690
 3rd  Qu.:16333  3rd Qu.:0.5091   3rd Qu.:41.7    3rd Qu.: 6592   3rd Qu.:2050  3rd Qu.: 615000
 Max.    :29375  Max.   :0.6457   Max.   :95.7   Max.    :12035  Max.    :3685  Max.    :1750000
```

From the above summary we can see that the range of the values differs a lot and that that might result to different magnitudes. Therefore we are standardizing the data using scale() function.

Hide

```
#Standardizing the training and testing data
x_train_scale <- as.data.frame(scale(x_train))
x_test_scale <- as.data.frame(scale(x_test))
```

Checking the summary after Standardizing

Hide

```
summary(x_train_scale)
```

```
   Start_Hour         weekdays            peak          IncomeScor        LivingEnSc      Rat
ioEmployee
 Min.   :-1.70150   Min.   :-1.49500   Min.   :-0.612   Min.   :-1.81299   Min.   :-3.1202   Mi
n.   :-0.3211
 1st Qu.:-0.80740   1st Qu.:-0.99538   1st Qu.:-0.612   1st Qu.:-0.84607   1st Qu.:-0.6639   1st
Qu.:-0.2861
 Median :-0.06232   Median :  0.00386  Median :-0.612   Median :-0.07253   Median :-0.0533   Med
ian :-0.2515
 Mean    : 0.00000  Mean   :  0.00000  Mean    : 0.000  Mean   : 0.00000   Mean    : 0.0000  Mea
n   : 0.0000
 3rd Qu.:  0.83178  3rd Qu.:  1.00310  3rd Qu.:  1.634  3rd Qu.:  0.89439  3rd Qu.:  0.6238  3rd
Qu.:-0.1069
 Max.    : 1.72589  Max.   :  1.50272  Max.    : 1.634  Max.   :  2.34477  Max.    : 2.6487  Ma
x.   : 8.4667
   GrenSpace            PopDen        RatioNotBornUK       NoCTFtoH        NoFlatsHouse       No
OwndDwel
 Min.    :-1.3689  Min.    :-2.25968  Min.    :-1.9085  Min.    :-1.2306  Min.    :-2.1017  Min.
:-2.05547
 1st Qu.:-0.7004   1st Qu.:-0.66467   1st Qu.:-0.7023   1st Qu.:-0.8688   1st Qu.:-0.6455   1st
Qu.:-0.58836
 Median :-0.3548   Median :-0.05063   Median :-0.1374   Median :-0.1586   Median :-0.1992   Medi
an :-0.08903
 Mean    : 0.0000  Mean    : 0.00000  Mean    : 0.0000  Mean    : 0.0000  Mean    : 0.0000  Mean
: 0.00000
 3rd Qu.: 0.6518   3rd Qu.: 0.74041   3rd Qu.: 0.7070   3rd Qu.: 0.6002   3rd Qu.: 0.4968   3rd
Qu.: 0.49610
 Max.    : 3.8218  Max.    : 3.53099  Max.    : 2.3289  Max.    : 2.9823  Max.    : 4.9699  Max.
: 3.30162
   MedHPrice
 Min.   :-1.2380
 1st Qu.:-0.6752
 Median :-0.2792
 Mean   : 0.0000
 3rd Qu.: 0.4365
 Max.   : 4.8874
```

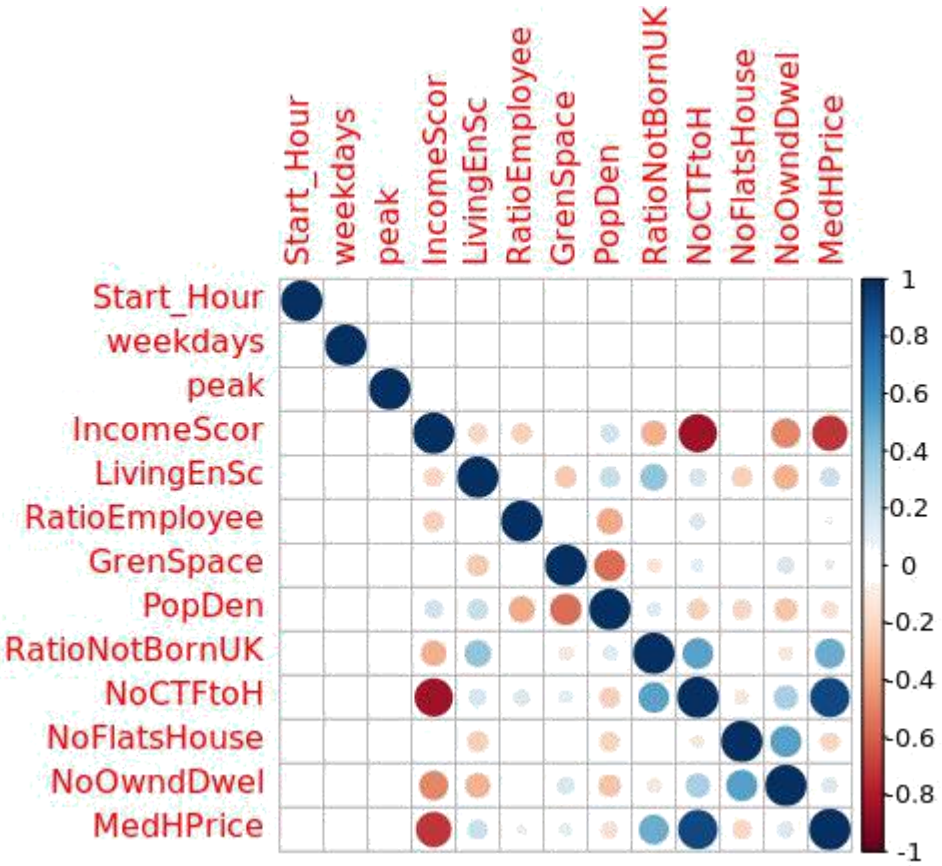
Now we can see that the variables are standardized and the mean value are zero and the variance at 1.

Checking correlation between the variables

We are looking at the correlation among the features and removing one of the two feature which are highly correlated because these features does not give any extra information to the model.

Hide

```
library(corrplot)
corrplot(cor(x_train_scale))
```



From the above plot we can see that certain features are correlated with wach other: 1. MedHPrice, IncomeScor, NoCTFtoH are correlated with each other

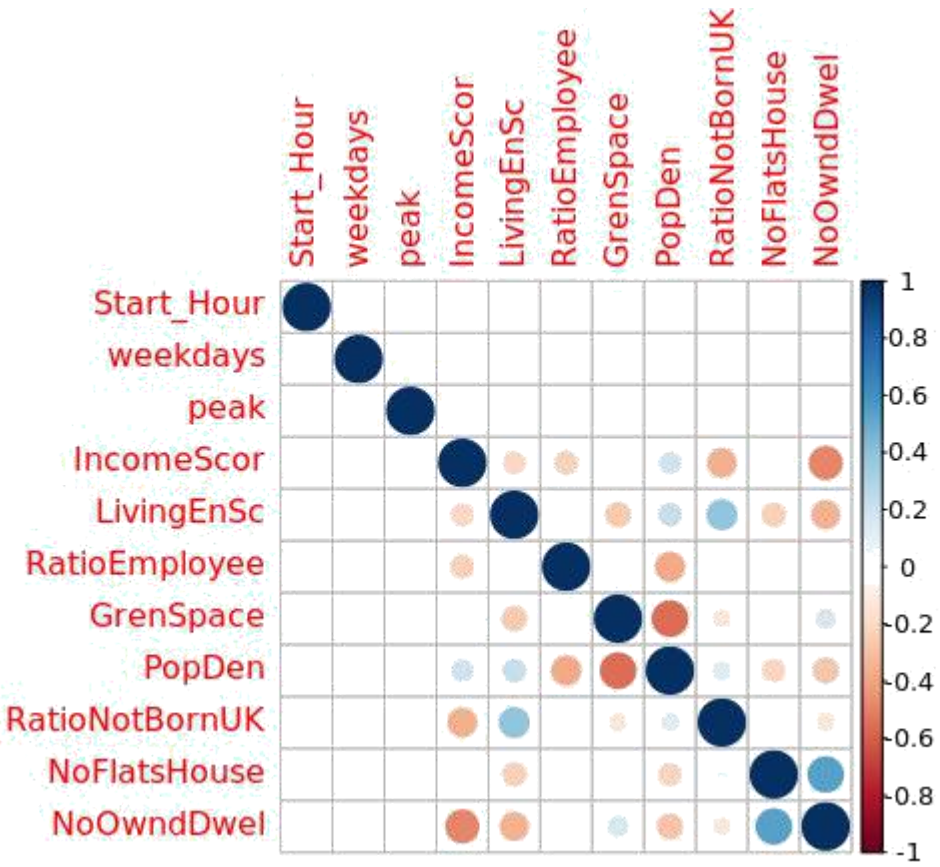
Therefore we are removing 4 features namely, NoDwelling, MedHprice, NoCTFtoH, RatioBornUK

Hide

```
x_train_scale$NoCTFtoH = NULL
x_train_scale$MedHPrice = NULL
x_test_scale$NoCTFtoH = NULL
x_test_scale$MedHPrice = NULL
```

Hide

```
corrplot(cor(x_train_scale))
```



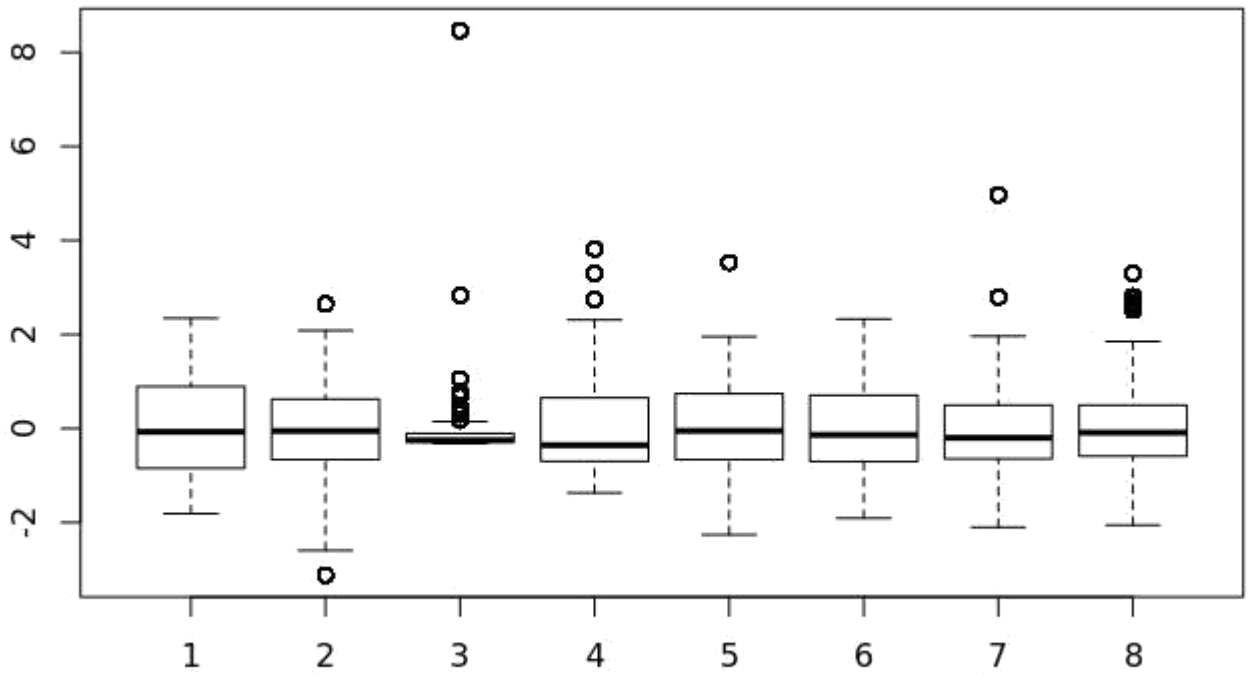
Now we can see that the features are not correlated with each other.

Outlier detection

We are plotting a boxplot graph to check whether there is outlier. If outlier is not checked we are likely to introduce bias in the model.

Hide

```
## Boxplot to check outlier
boxplot(x_train_scale$IncomeScor, x_train_scale$LivingEnSc,
        x_train_scale$RatioEmployee, x_train_scale$GrenSpace,
        x_train_scale$PopDen, x_train_scale$RatioNotBornUK,
        x_train_scale$NoFlatsHouse, x_train_scale$NoOwndDwel)
```



We can see that RatioEmployee has outliers. We can do any one of the following method to deal outliers.

1. We can remove the feature.
2. Assign median value.
3. Assign mean value of the remaining values.

Algorithm

Linear regression

In this project we have to predict the number of bike rented therefore this ia a regression problem. In this project we are going to use linear regression. This model uses a straight line to show the relationship between the variables. The model fit the best line by searching for the value of coefficients that minimize the error of the model. We are combining the x and y variables to fit and predict the linear regression model.

Hide

```
#Train and test dataset
TRAIN <- cbind(x_train_scale, y_train_scale)
TEST <- cbind(x_test_scale, y_test_scale)
```

Hide

```
##Linear regression
##Model is initiated using lm() function
lr = lm(Count ~ ., data=TRAIN)
```

Hide

```
##predicting the train and test
train_preds = predict(lr, TRAIN)
test_preds = predict(lr, TEST)


library(caret)
mae_train <- MAE(train_preds, TRAIN$Count)
mse_train <- DescTools::MSE(train_preds, TRAIN$Count)
rmse_train <-  RMSE(train_preds, TRAIN$Count)
r2_train <- R2(train_preds, TRAIN$Count, form = "traditional")

mae_test <- MAE(test_preds, TEST$Count)
mse_test <- DescTools::MSE(test_preds, TEST$Count)
rmse_test <-  RMSE(test_preds, TEST$Count)
r2_test <- R2(test_preds, TEST$Count, form = "traditional")

Dataset <- c('Training dataset', 'Testing dataset')
R2 <- c(r2_train, r2_test)
MAE <- c(mae_train, mae_test)
MSE <- c(mse_train, mse_test)
RMSE<- c(rmse_train, rmse_test)

measures <- data.frame(Dataset, R2, MAE, MSE, RMSE)
measures
```

Dataset <fctr>	**R2** <dbl>	**MAE** <dbl>	**MSE** <dbl>	**RMSE** <dbl>
Training dataset	0.3630801	0.9314220	1.333902	1.154947
Testing dataset	0.3498564	0.9278249	1.335601	1.155682

2 rows

Evaluating the model

1. R-Square R - square is the goodness of fit measure for linear regression model. This value range between 0 and 1. If the value is closer to 1 then the model fits the data very well. We can see that both train and test R-square values are similar. Thus it is clear that there no overfitting in the model.

2. Mean Absolute Error: MAE describes the magnitude of the residuals even though it does not show the underperformance or overperformance of the model it shows whether the model is perfect predictor of the outputes. If the MAE value is closer to zero or zero then the model is perfect predictor. In our model mae score is closer to 1 which shows that the model is not a perfect predictor.

3. Mean Square Error: MSE is the square difference between the actual and predicted values before summing them all. This value will always be bigger than MAE because of the square. The MSE value ranges from 0 to infinity so there is a chance that the MSE value can be bigger and as it become larger the interpretation of the model will be harder.From the above output we can see that the MSE value is very close to 1 and that means the model is not that good predictor of the output.

4. Root Mean Square Error: To overcome the interpretation problem in the MSE we can use this metric.We can see that the MSE and RMSE values is higher than the MAE value This is due to the outliers in the data. MSE and RMSE is affected by the outliers. In this model we did not deal the outliers which affects RMSE measure.

Looking at the Summary of the model

We are using summary() function to get the summary of the model. This function return function, residuals, coefficients, p-value, residual standard error, multiple r squared, f-statistics.

Hide

```
summary(lr)
```

```
Call:
lm(formula = Count ~ ., data = TRAIN)

Residuals:
    Min      1Q   Median      3Q     Max 
-4.3906 -0.7527  0.0736  0.8389  3.4150 

Coefficients:
               Estimate Std. Error t value  Pr(>|t|)    
(Intercept)     3.534502   0.009819 359.974  < 2e-16***
Start_Hour      0.614347   0.009824  62.535  < 2e-16***
weekdays        0.006612   0.009820   0.673  0.50076    
peak            0.427184   0.009822  43.490  < 2e-16***
IncomeScor     -0.168157   0.017663  -9.520  < 2e-16***
LivingEnSc      0.038240   0.012346   3.097  0.00196**
RatioEmployee   0.295728   0.012027  24.589  < 2e-16***
GrenSpace      -0.022990   0.012540  -1.833   0.06678 .
PopDen          0.068883   0.013949   4.938 7.98e-07***
RatioNotBornUK  0.068919   0.013078   5.270 1.39e-07***
NoFlatsHouse    0.227262   0.014937  15.215  < 2e-16***
NoOwndDwel     -0.276900   0.019840 -13.957  < 2e-16***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 1.155 on 13836 degrees of freedom
Multiple R-squared:  0.3631,    Adjusted R-squared:  0.3626 
F-statistic:   717 on 11 and 13836 DF,  p-value: < 2.2e-16
```

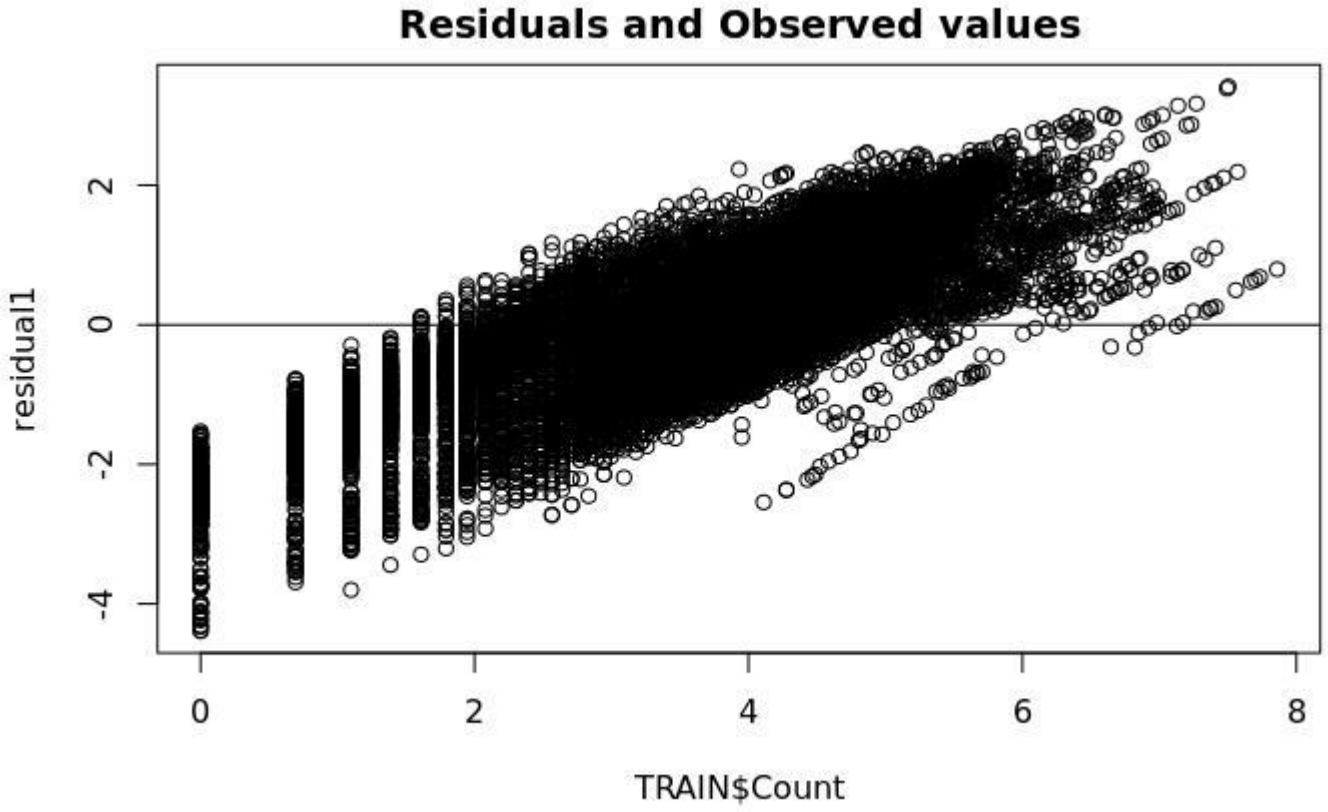
SUMMARY INTERPRETATION:

1. Residuals: Residuals are the difference between the actual and predicted values. In the summary, we get 5 summary details. From this we can understand the distribution of the residuals. The Median value is 0.0736, which indicates that the residuals are symmentrically distributed. Therefore, the model predicted certain points that fall closer to the actual values. We can further plot the residuals distribution as shown below.

Plotting residuals

Hide

```
#We are using resid() to get the residuals of the
model residual1 <- resid(lr)
plot(TRAIN$Count, residual1, main="Residuals and Observed values")
abline(0,0)
```



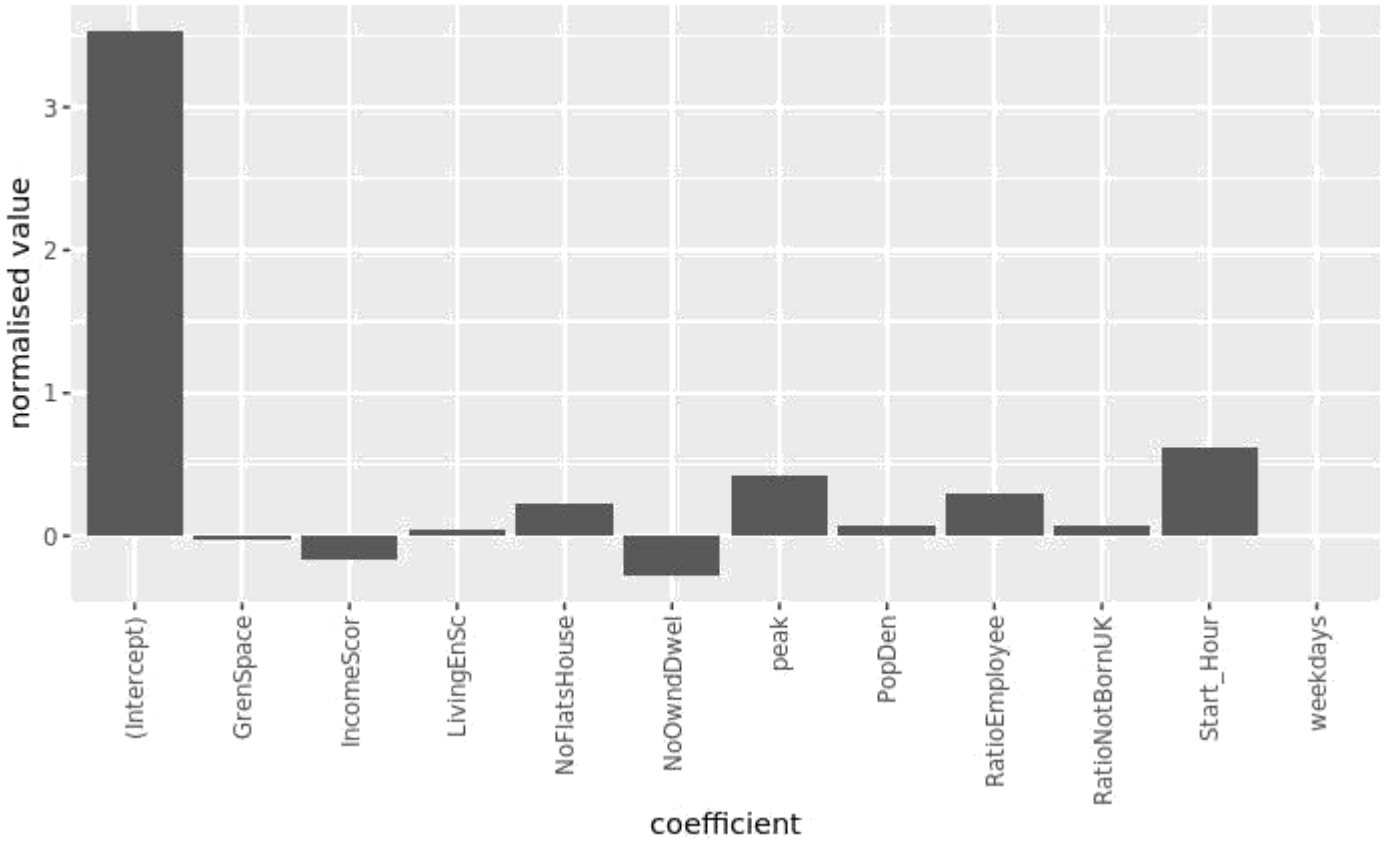
From the above graph we can see that the residuals are symetrically distributed.

2. **Coefficient** - We can use the Coefficient measures to deciden accept or reject the hypothesis. We can plot the coefficient to understand the measures easily.

Hide

```
##Ploting Coefficient
library(ggplot2)

ggplot(, aes(x = names(lr$coefficients),y=lr$coefficients, ))+
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust = 0.5)) +
  xlab("coefficient") +
  ylab("normalised value")
```



The above graph shows that weekdays variable has 0 coefficient. This is beacuse there is very less variation of values which means that it only has very less variation on the dependent variables. Therefore, the p value of the variable is zero. We can look at the other measures for better understanding for this variable.

T-value

From the t-value we can check the relationship between the features. If the t-value is far away from 0 then we can accept the hypothesis. In our case the t-value of the weekdays is 0.579 which means that it is very close to 0 and there is no relationship between weekdays and dependent variable. peak has higher value (far from 0) which shows that the hypothesis can be accepted.

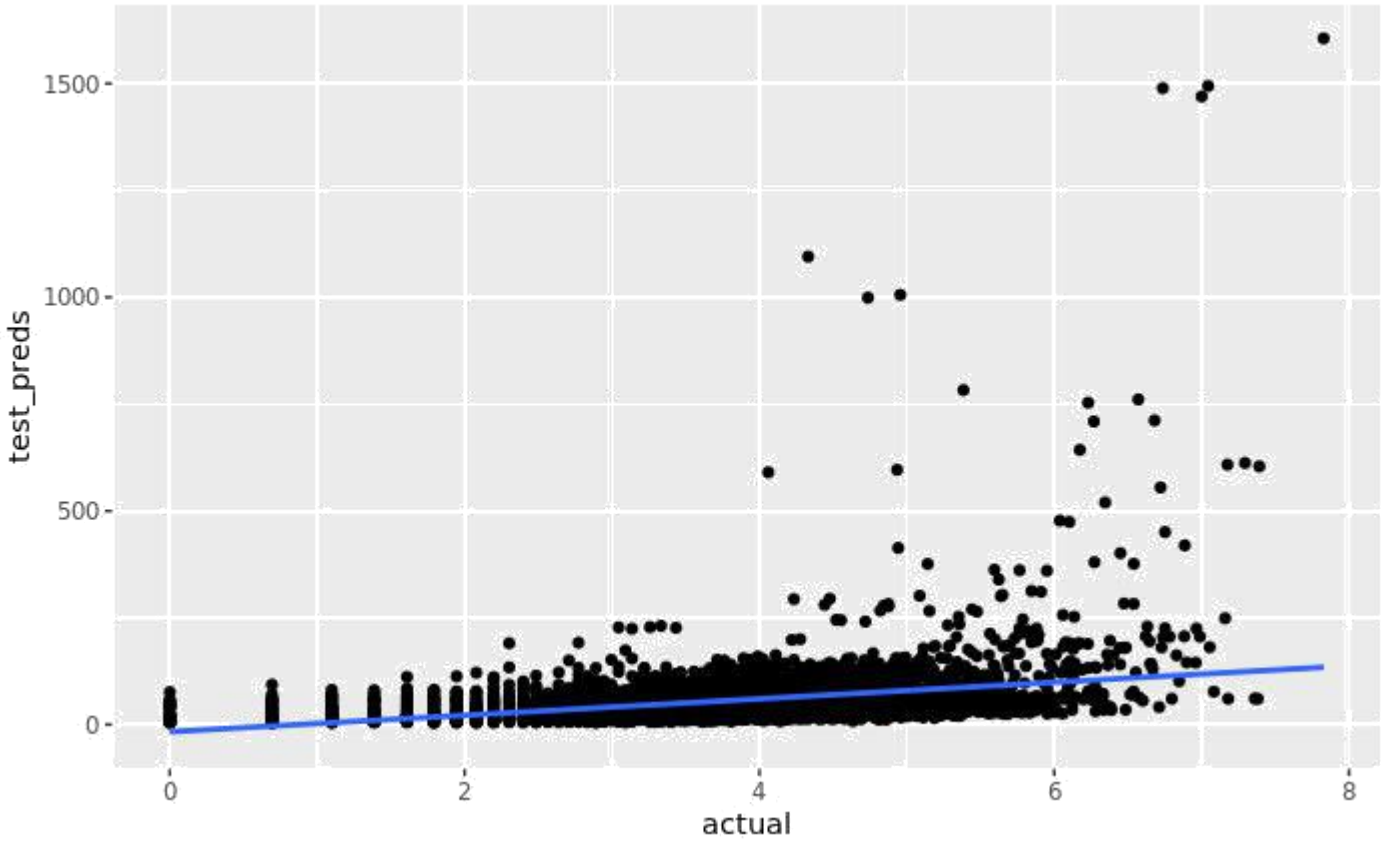
R square In our model the R square value is 0.36 which means that 36% of the variance found in the dependent variable can be explained by the independent variables.

F-Statistic F-statistic is also a good measure to show the relationship between the variables. In our model, f-statistic score is 717. If this value is far from zero then there is relationship between the independent varibles and dependent variable.

Plotting the actual and predicted values for the test data

Hide

```
##Storing the results in a dataf frame
test_results <- data.frame(test_preds = exp(test_preds), actual = TEST$Count)
ggplot(test_results, aes(x=actual, y=test_preds)) +
 geom_point() +
 geom_smooth(method = "lm")
```



From the graph we can see that most of the values falls near the regression line.

Final Results

Finalizing whether the hypothesis is true or false:

Hypothesis 1: The number of bike rented during the weekends will be higher. This hypothesis is **FALSE**. This is because the coefficient and the t-statistic value is 0 and 0.579 respectively. these measures shows there is no relationship between this feature and the dependent feature. (FALSE HYPOTHESIS)

Hypothesis 2: The number of bike rented during the peak time is higher. This hypothesis is **TRUE**. The coefficient of this variable is high. And the graphs shown before also proves that there is more bikes rented during the peak time. (TRUE HYPOTHSIS)

Hypothesis 3: Higher bikes retend in the less deprived area. This hypothesis can be said as **Partially TRUE** because we have taken two metrics to prove this hypothesis: Income Score and living environment(IncomeScor, LivingEnSc). If the values for both feature is high means then the area is more deprived. Therefore the coefficient of the IncomeScor is negative which means that the less income score area rents more bike. And livingEnSc is zero which indicates that there is no relationship between the feature and dependent variable.(PARTIALLY TRUE)

Hypothesis 4: Higer Bikes are rented where there is more green space area. This hypothesis is **FALSE**. The coefficents shows negative value. (FALSE)

Hypothesis 5: Higher bikes are rented where there is more population. This hypothesis is **FALSE** because there is negative coefficient value (FALSE)

Hypothesis 6: Higher bikes are rented where more number of non-born UK people live. This is also **FALSE** hypothesis. Because the p value is zero.

Hypothesis 7: Higher bikes are rented where there is higher number of properties councel tax. This hypothesis is **FALSE** because it is highly correlated with INCOMESCORE variable. (FALSE HYPOTHESIS)

Hypothesis 8: Higher bikes are rented where there is more number of flats. This hypothesis is **TRUE** because it has higher coefficient value. (TRUE HYPOTHESIS)

Hypothesis 9: Higher bikes are rented where there is more number of owned properties. This hypothesis is **TRUE** because it was correlated with NoFlats feature. (TRUE HYPOTHESIS)

Hypothesis 10: Higher bikes are rented where there house price is higher.

Limitations

1. In this model the data is splitted using train and test split method where 75% of data is train and 25% is test. But better choice of splitting the dataset into train and test is using **K-fold cross validation**.

2. Dealing Outliers: There is steps taken to remove or handle the ouliers in the model. The outliers will affect the prediction power of the model and also the measures like MSE and RMSE.

3. From the R-square value we can see that only 36% of the variation of the outcome is explained by the model. This shows that there are many other imporatant features for predicting the bike rent is not included.