Bike Renting Prediction

Somyanath Mohanty

9th October 2018

Contents

Chapter 1

Introduction

Problem Statement

The problem statement here is to determine the factors affecting the bike rentals and build a model which can predict the bike rental count on daily basis based on the environmental and seasonal settings.

Data

Our task is to build a regression model which will predict the bike rental count on daily basis. Given below is a sample of the dataset that we are using to predict the bike rental count:

Table 1.1: Bike rent sample Data (Columns 1-9)

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

Table 1.2: Bike rent sample Data (Columns 10-15)

temp	atemp	hum	windspeed	casual	registered
0.344167	0.363625	0.805833	0.160446	331	654
0.363478	0.353739	0.696087	0.248539	131	670
0.196364	0.189405	0.437273	0.248309	120	1229
0.2	0.212122	0.590435	0.160296	108	1454
0.226957	0.22927	0.436957	0.1869	82	1518

Table 1.3: Bike rent sample Data (Column 16)

cnt
985
801
1349
1562
1600

As you can see in the table below we have the following 16 variables. We'll treat the variables *"casual"* and *"registered"* also as target variables because the target variable *"cnt"* is the sum of both these variables. Thus we'll use the remaining 13 variables to create two separate models for *"casual"* and *"registered"* and add their best predictions to predict the our specified target variable *"cnt"*.

Table 1.4: Predictor and Dependant Variables

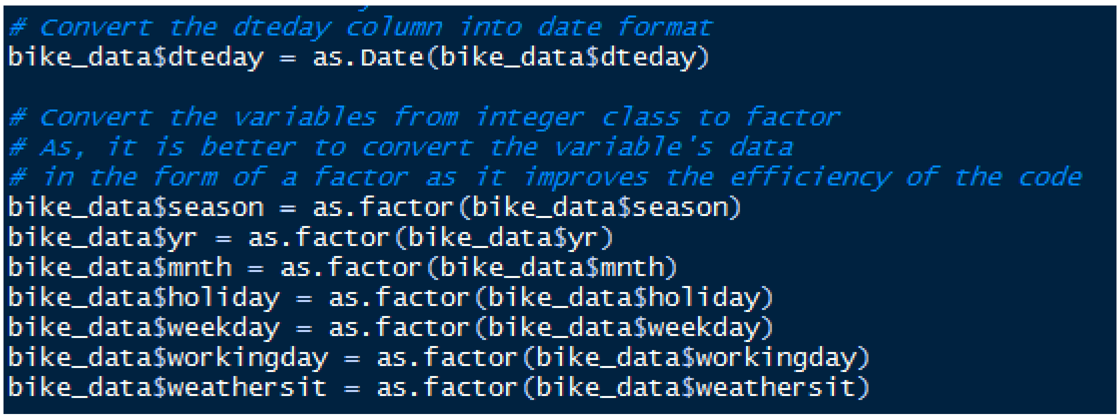
S.No.	Predictor
1	instant
2	dteday
3	season
4	yr
5	mnth
6	holiday
7	weekday
8	workingday
9	weathersit
10	temp
11	atemp
12	hum
13	windspeed
14	casual
15	registered
16	cnt

Chapter 2

Methodology

Pre-Processing

Any predictive modelling requires that we look at the data before we start modelling. However, in data mining terms looking at data refers to so much more than just looking. Looking at data refers to exploring the data, cleaning the data as well as visualizing the data through graphs and plots. This is often called **Exploratory Data Analysis**. To start this process, we will first modify some variables just by observing the dataset.



```
# Convert the dteday column into date format
bike_data$dteday = as.Date(bike_data$dteday)

# Convert the variables from integer class to factor
# As, it is better to convert the variable's data
# in the form of a factor as it improves the efficiency of the code
bike_data$season = as.factor(bike_data$season)
bike_data$yr = as.factor(bike_data$yr)
bike_data$mnth = as.factor(bike_data$mnth)
bike_data$holiday = as.factor(bike_data$holiday)
bike_data$weekday = as.factor(bike_data$weekday)
bike_data$workingday = as.factor(bike_data$workingday)
bike_data$weathersit = as.factor(bike_data$weathersit)
```

Fig 2.1: Code Snippet for "instant" removal and variable modification

Here, we have modified the variables "*season*", "*yr*", "*mnth*", "*holiday*", "*weekday*", "*workingday*" and "*weathersit*" into factor as they have very limited number of values. So instead of treating them as numbers or integers we can convert them into factor and improve the efficiency of the model and the code a bit.

Now, we can check the probability distributions of the variables. Most analysis like regression, require the data to be normally distributed. We can visualize that in a glance by looking at the probability distributions of the variables.

In fig 2.2, the red lines represent the normal distribution. So, as you can see in the figure most of the variables either very closely or somewhat imitate the normal distribution.

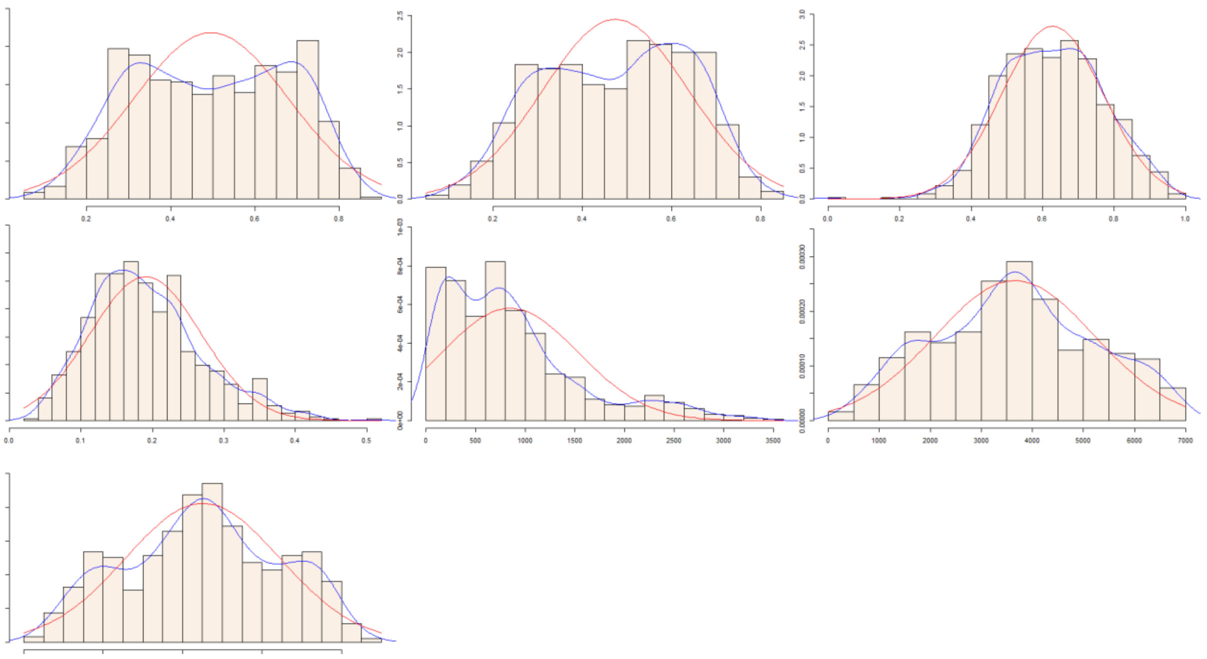


Fig 2.2: Probability Density Functions of Bike Renting data

Outlier Analysis

We can clearly observe from these probability distributions that few of the variables are skewed, for example, *"hum", "windspeed"* and "*casual*". The skew in these distributions can be most likely explained by the presence of outliers and extreme values in the data.

One of the other steps of **pre-processing** apart from checking for normality is the presence of outliers. In this case we use a classic approach of imputing outliers using **KNN Imputation**. We can visualize the outliers using the *boxplots*.

In fig 2.3 - 2.8 we have plotted the boxplots of the 6 numerical variables with respect to the variable most responsible for their variability. The important thing to notice is that only the variables *"hum", "windspeed"* and "*casual*" have outliers. Also, the outliers in *"hum"* and *"windspeed"* variables must be due to extreme weather conditions. The outliers in *"casual"* variable can be due to many factors one of which could be bulk rentals due to some cycling expedition or group bookings. There can be countless reasons for the outliers.

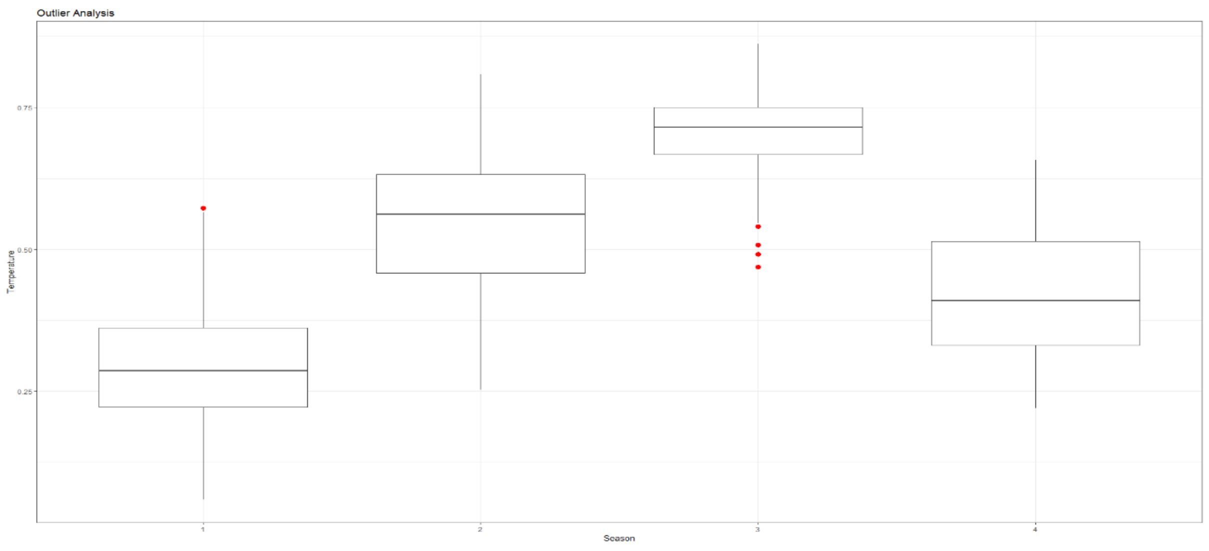


Fig 2.3: Boxplot of Temperature w.r.t Season

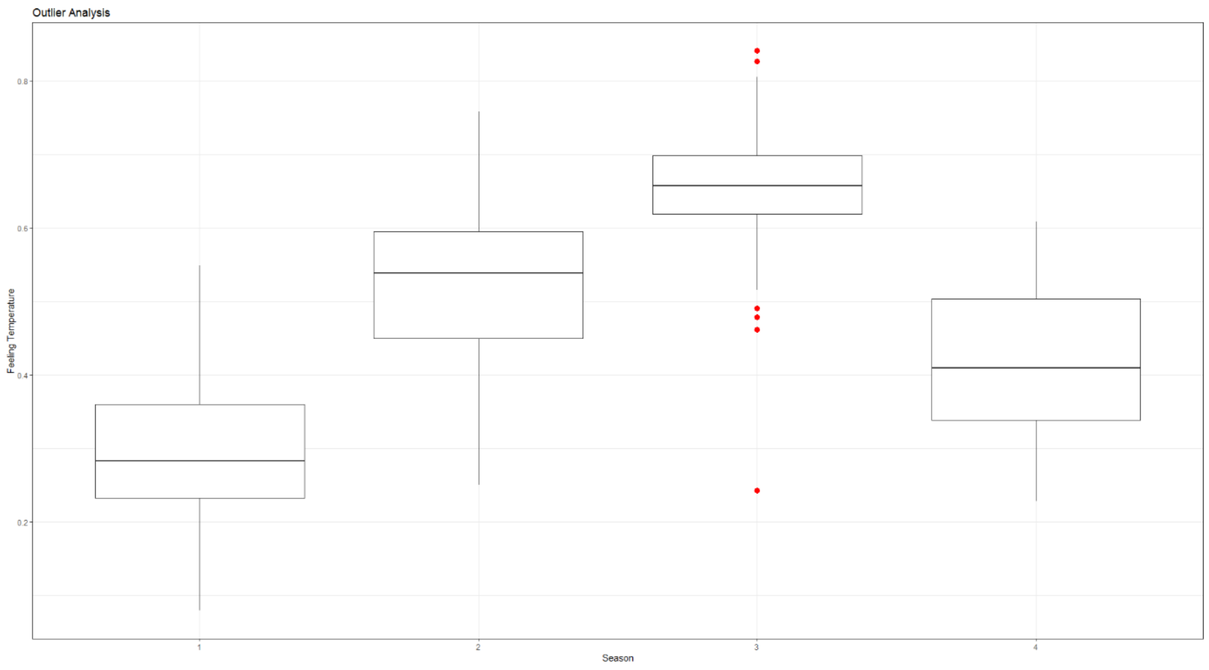


Fig 2.4: Boxplot of Feeling Temperature w.r.t Season

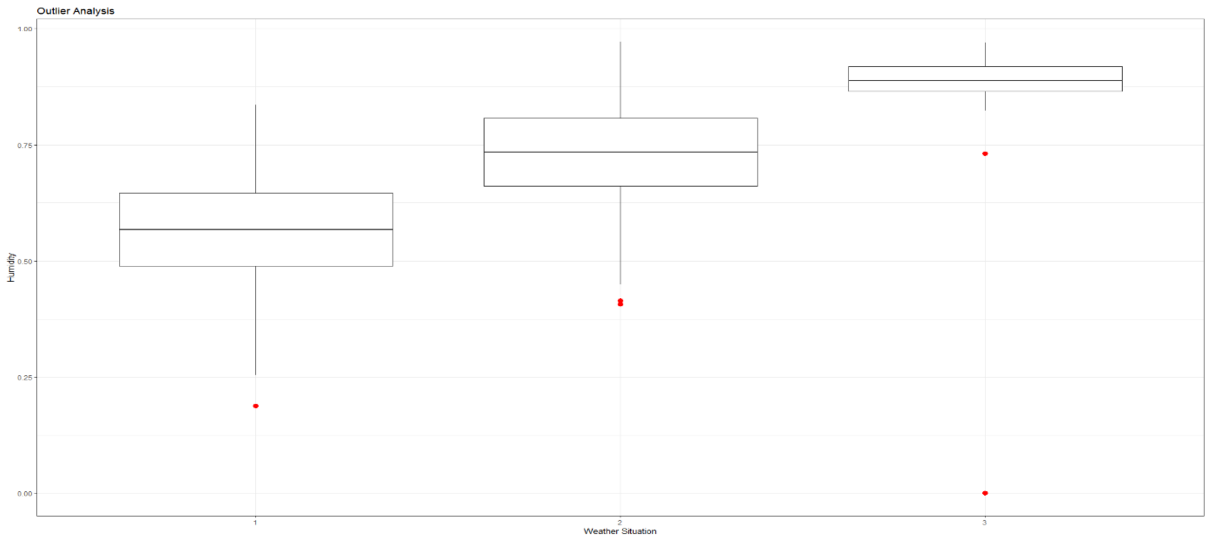


Fig 2.5: Boxplot of Humidity w.r.t Weather Situation

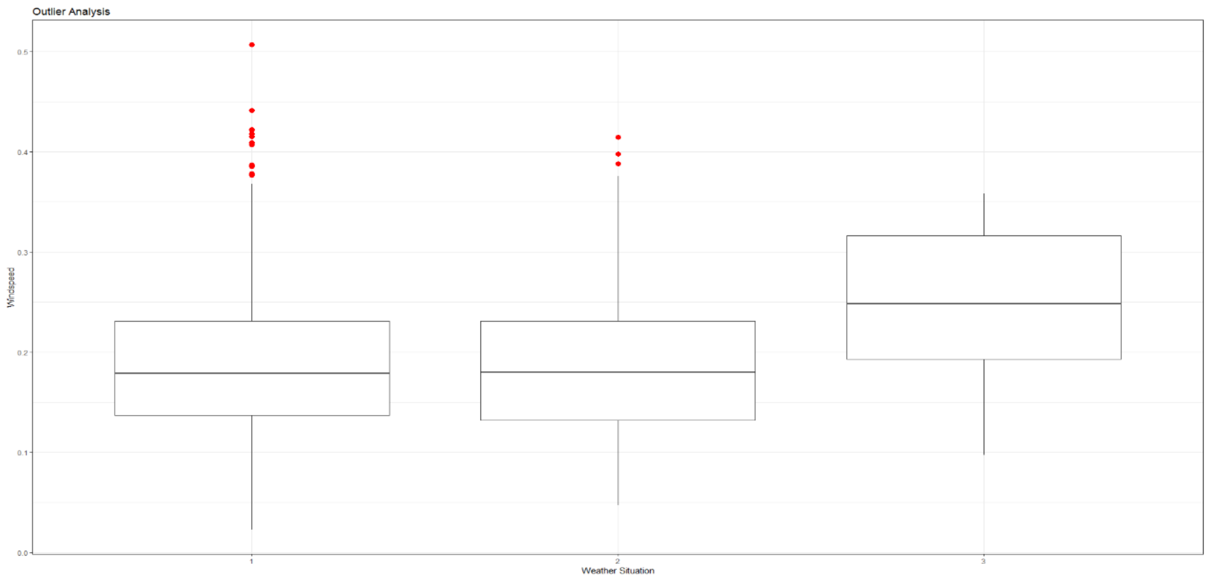


Fig 2.6: Boxplot of Windspeed w.r.t Weather Situation

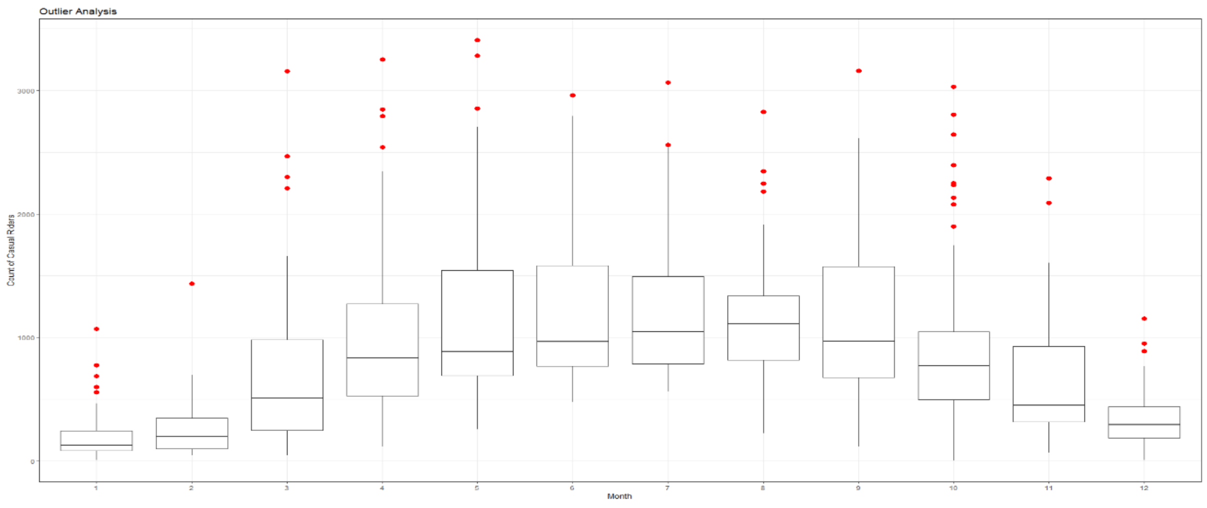


Fig 2.7: Boxplot of Casual Riders w.r.t Month

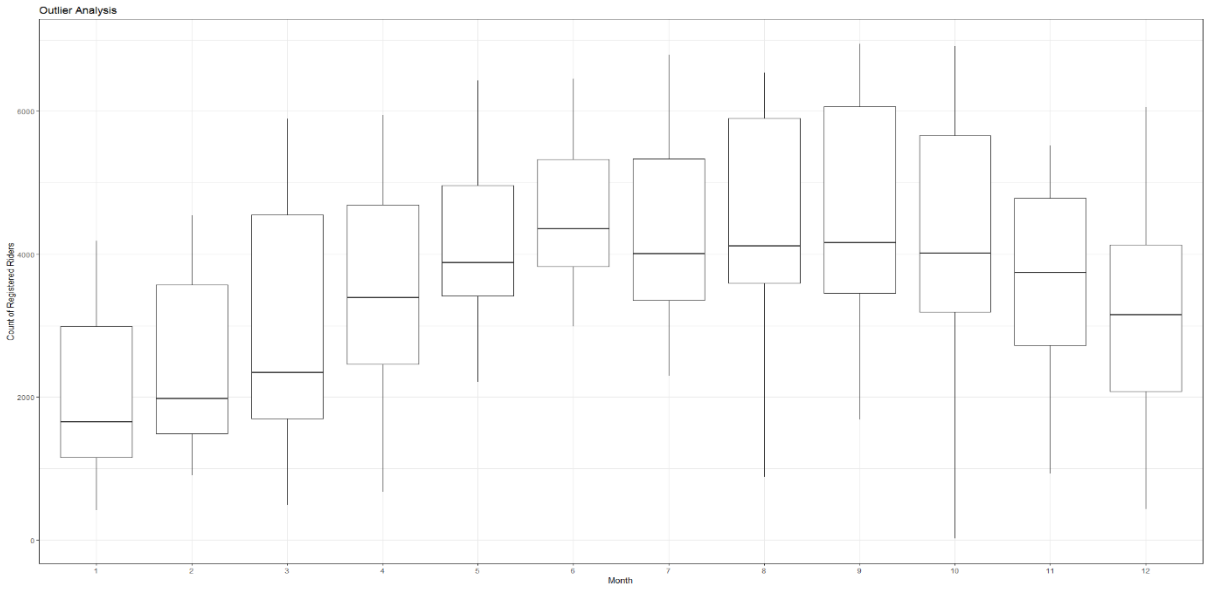


Fig 2.8: Boxplot of Registered w.r.t Month

In fig 2.9, 2.10 and 2.11 we have plotted the boxplots for the variables *"hum"*, *"windspeed"* and "*casual*" after outlier removal using KNN Imputation.

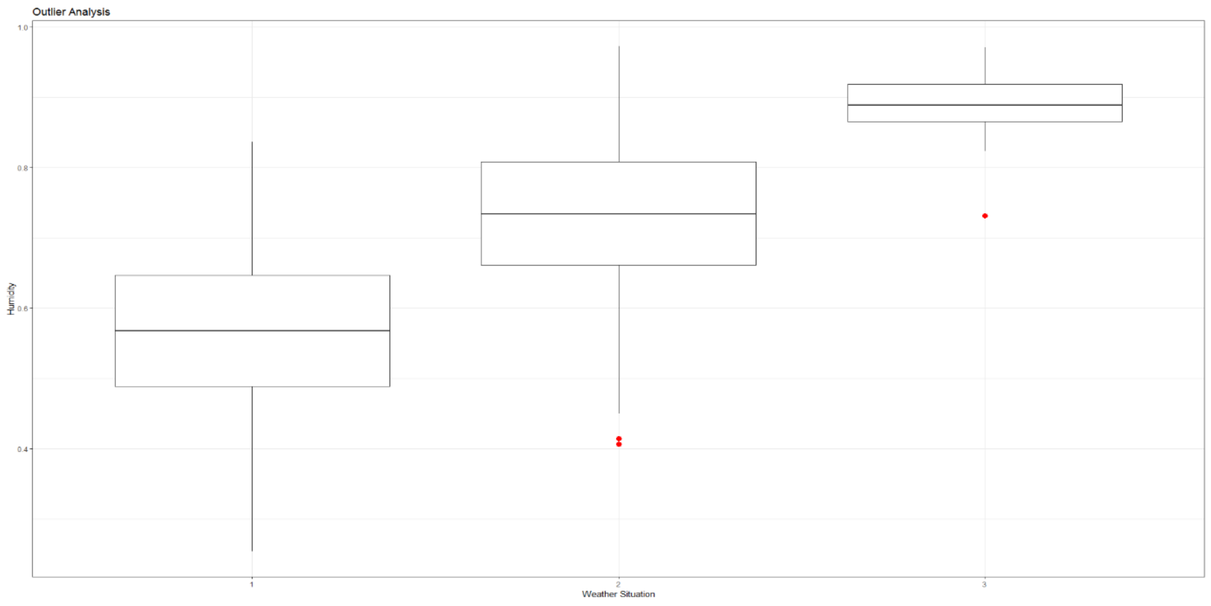


Fig 2.9: Boxplot of Humidity w.r.t Weather Situation After Outlier Analysis

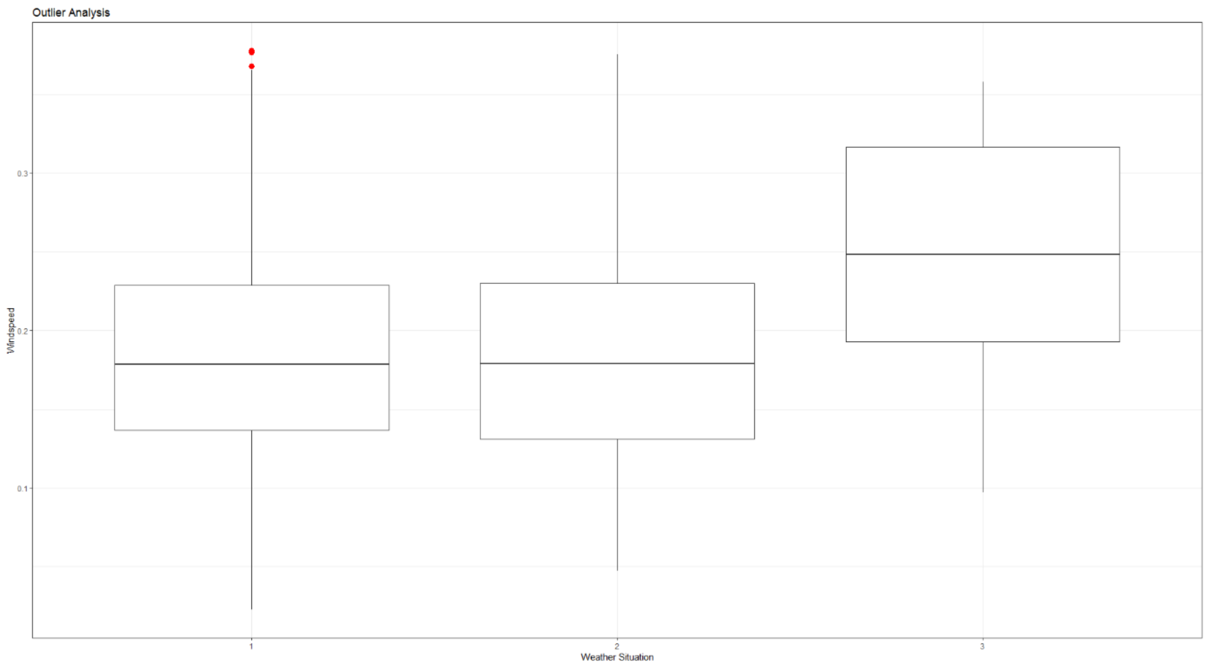


Fig 2.10: Boxplot of Windspeed w.r.t Weather Situation After Outlier Analysis

In fig 2.12 we have plotted the probability distribution of the numerical variables after performing outlier analysis. As we can see that most of the outliers were in the *"casual"* variable and those have been removed and imputed with the most suitable value using the KNN Imputation algorithm.

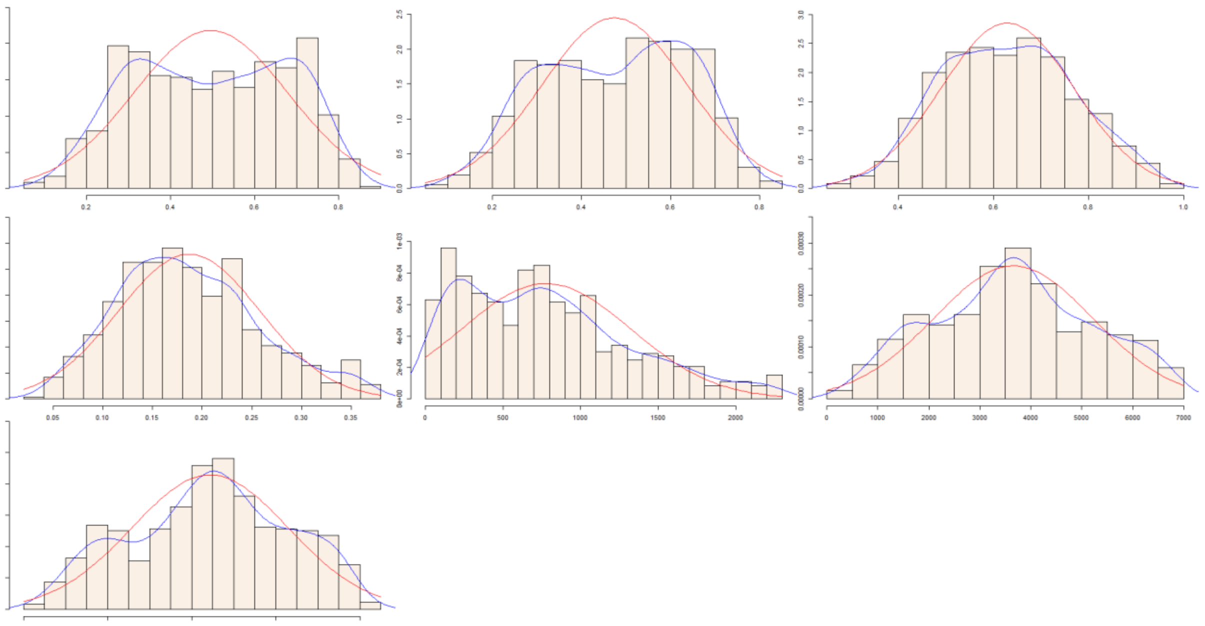


Fig 2.12: Probability Density Functions of Bike Renting data After Outlier Removal

We can see that the distribution of the variables *"hum"* and *"windspeed"* have become less skewed and is concentrated around the centre of the curve.

Feature Selection

Before performing any type of modelling, we need to assess the importance of each predictor variable in our analysis. There is a possibility that many variables in our analysis are not important at all to the problem of prediction. There are several methods of doing that, we can opt for correlational analysis for the numerical variables and chi-square test for the categorical variables.

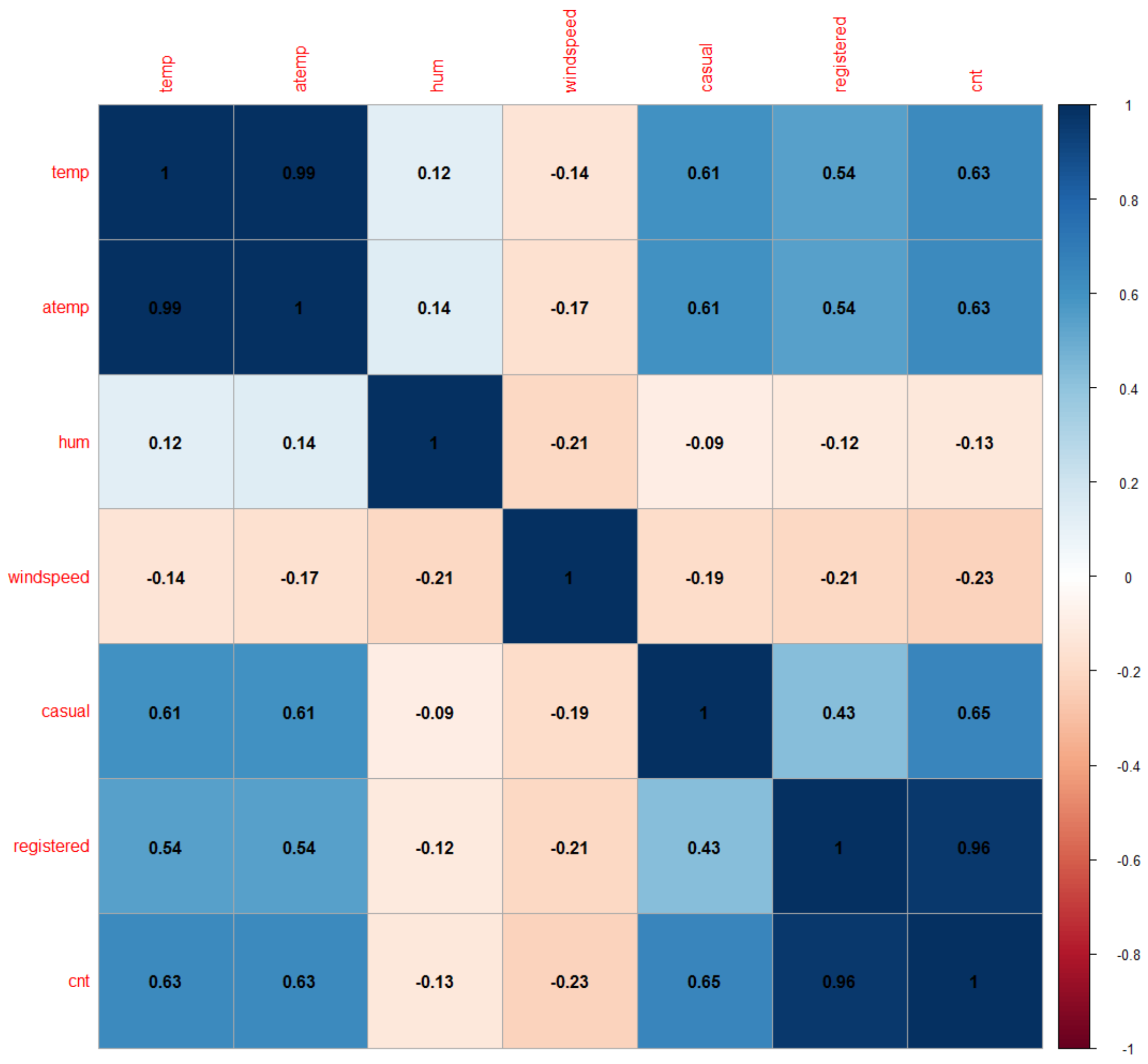
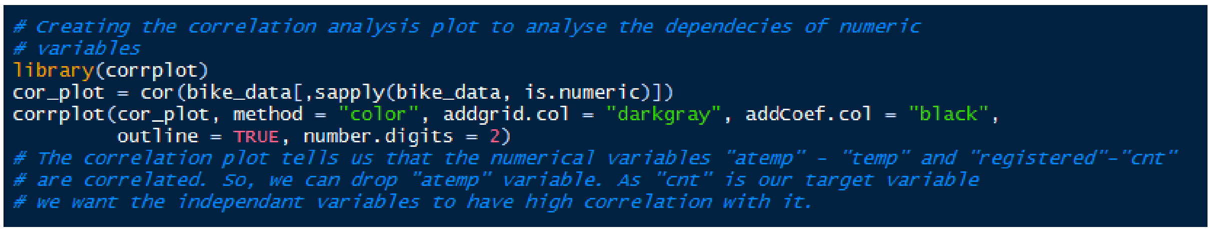
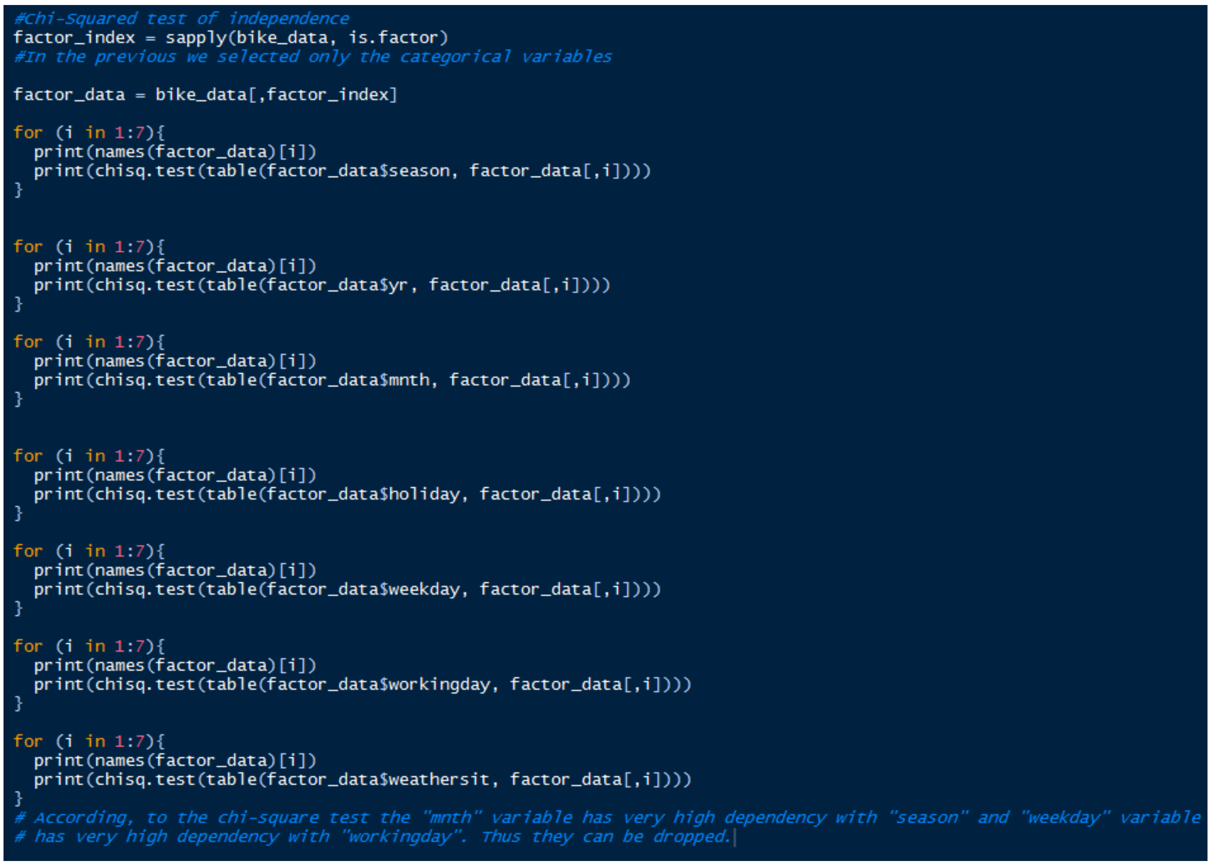


Fig 2.13: Correlation Analysis Plot



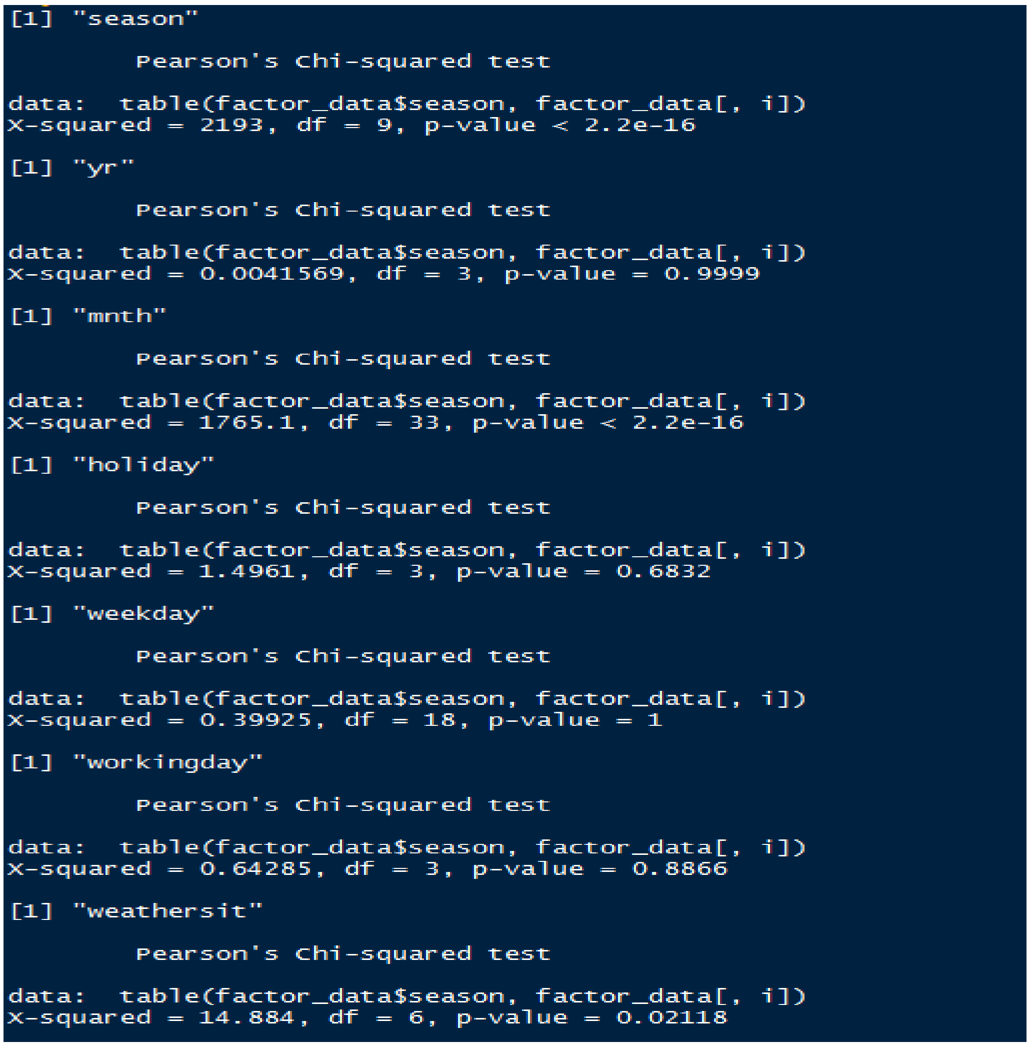
```
# Creating the correlation analysis plot to analyse the dependecies of numeric
# variables
library(corrplot)
cor_plot = cor(bike_data[,sapply(bike_data, is.numeric)])
corrplot(cor_plot, method = "color", addgrid.col = "darkgray", addCoef.col = "black",
         outline = TRUE, number.digits = 2)
# The correlation plot tells us that the numerical variables "atemp" - "temp" and "registered"-"cnt"
# are correlated. So, we can drop "atemp" variable. As "cnt" is our target variable
# we want the independant variables to have high correlation with it.
```

Fig 2.14: Code Snippet of Correlation Analysis



```r
#Chi-Squared test of independence
factor_index = sapply(bike_data, is.factor)
#In the previous we selected only the categorical variables

factor_data = bike_data[,factor_index]

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$season, factor_data[,i])))
}


for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$yr, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$mnth, factor_data[,i])))
}


for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$holiday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$weekday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$workingday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$weathersit, factor_data[,i])))
}
# According, to the chi-square test the "mnth" variable has very high dependency with "season" and "weekday" variable
# has very high dependency with "workingday". Thus they can be dropped.
```

Fig 2.15: Code Snippet for Chi-Square Analysis



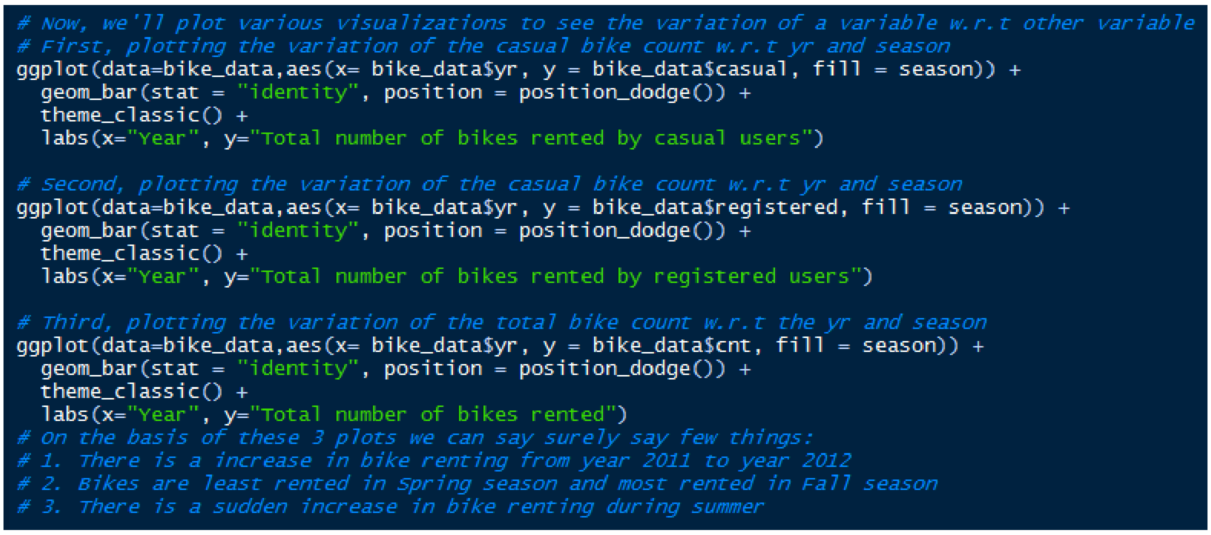
```
[1] "season"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 2193, df = 9, p-value < 2.2e-16

[1] "yr"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 0.0041569, df = 3, p-value = 0.9999

[1] "mnth"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 1765.1, df = 33, p-value < 2.2e-16

[1] "holiday"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 1.4961, df = 3, p-value = 0.6832

[1] "weekday"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 0.39925, df = 18, p-value = 1

[1] "workingday"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 0.64285, df = 3, p-value = 0.8866

[1] "weathersit"

        Pearson's Chi-squared test

data:  table(factor_data$season, factor_data[, i])
X-squared = 14.884, df = 6, p-value = 0.02118
```

Fig 2.16: Segment of Chi-Square Test result

In figure 2.13 we have plotted the correlation analysis plot of all the numerical variables. The plot shows that there is very high dependency between *"temp"*-*"atemp" and "casual"*-*"registered"*-*"cnt"*. So, we can drop *"atemp"*. Here we will avoid dropping *"registered"* or *"casual"* as they have high dependency with *"cnt"*, which is target variable. So, it is better for the model to have variables which have high dependency with target variable.

Also, according to chi-square test we can reject the *"mnth"* and *"weekday"* variable as they have high dependency with *"season"* and *"workingday"* and there should be very less dependencies between the independent variables.

We can also check the variation of the data between the variables to gauge out more information regarding the variation of the data and make better assumptions to further improve our model.



```
# Now, we'll plot various visualizations to see the variation of a variable w.r.t other variable
# First, plotting the variation of the casual bike count w.r.t yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$casual, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented by casual users")

# Second, plotting the variation of the casual bike count w.r.t yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$registered, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented by registered users")

# Third, plotting the variation of the total bike count w.r.t the yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$cnt, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented")
# On the basis of these 3 plots we can say surely say few things:
# 1. There is a increase in bike renting from year 2011 to year 2012
# 2. Bikes are least rented in Spring season and most rented in Fall season
# 3. There is a sudden increase in bike renting during summer
```

Fig 2.17: Code Snippet for Variation of count for casual, registered and total rents w.r.t year

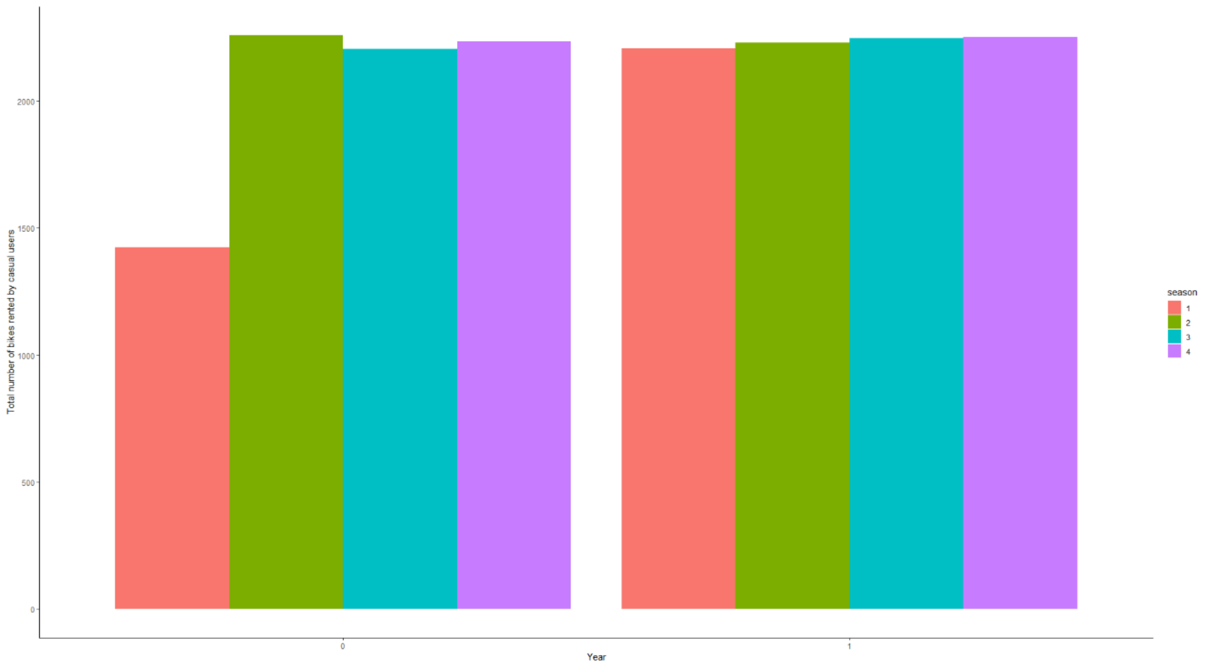


Fig 2.18: Variation of count for casual rents w.r.t year

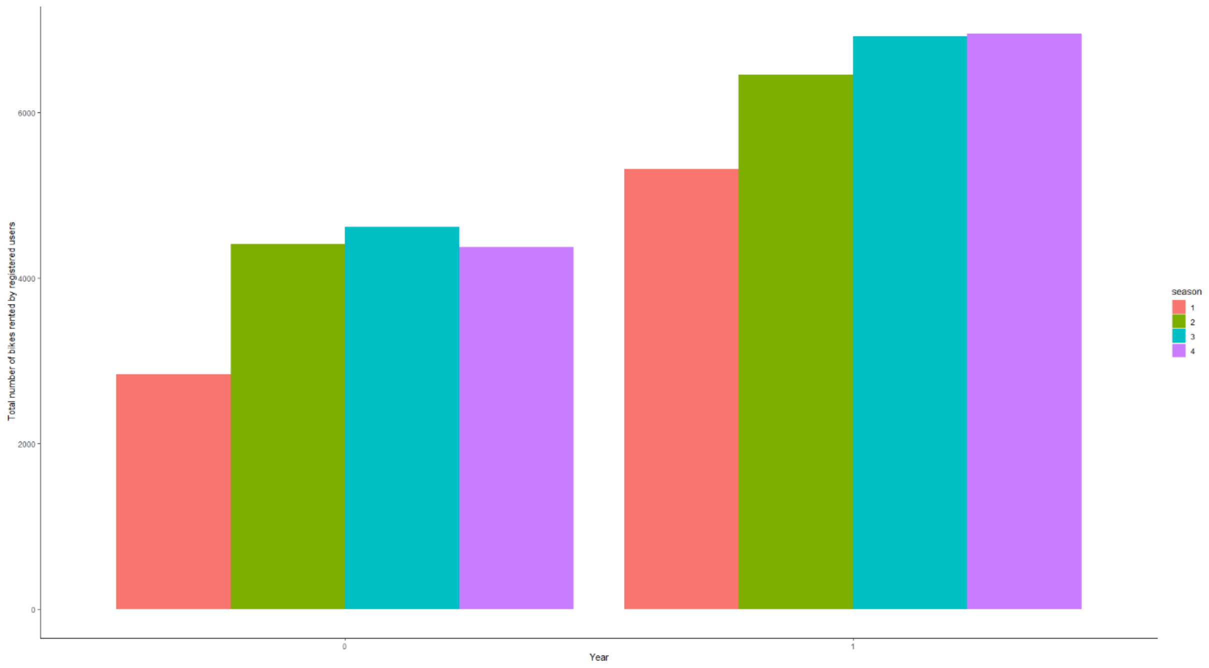


Fig 2.19: Variation of registered rents w.r.t year

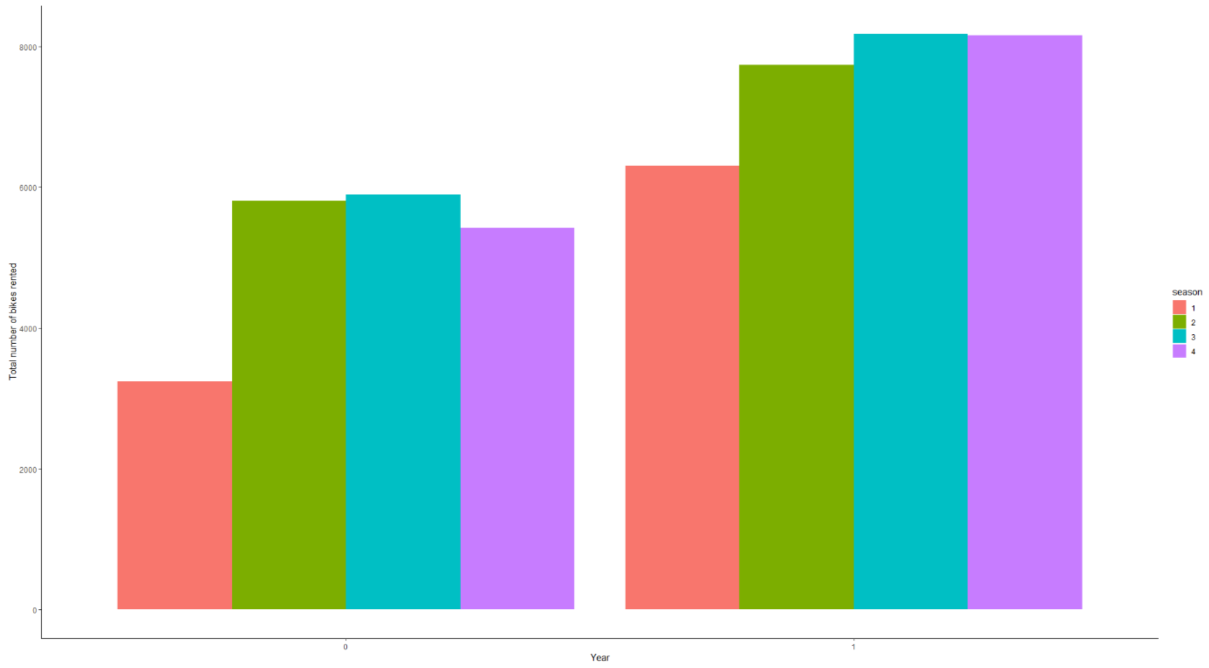
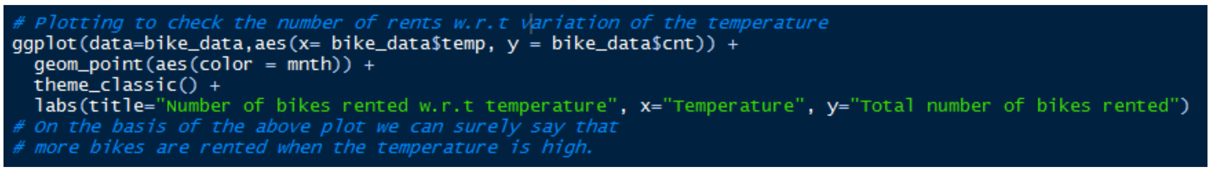


Fig 2.20: Variation of total rents w.r.t year

With the help of these plots we can surely say three things:

1. There is an increase in bike renting from year 2011 to year 2012
2. Bikes are least rented in Spring season and most rented in Fall season
3. There is a sudden increase in bike renting during summer



```
# Plotting to check the number of rents w.r.t variation of the temperature
ggplot(data=bike_data,aes(x= bike_data$temp, y = bike_data$cnt)) +
  geom_point(aes(color = mnth)) +
  theme_classic() +
  labs(title="Number of bikes rented w.r.t temperature", x="Temperature", y="Total number of bikes rented")
# On the basis of the above plot we can surely say that
# more bikes are rented when the temperature is high.
```

Fig 2.21: Code snippet for Scatterplot of count w.r.t temperature

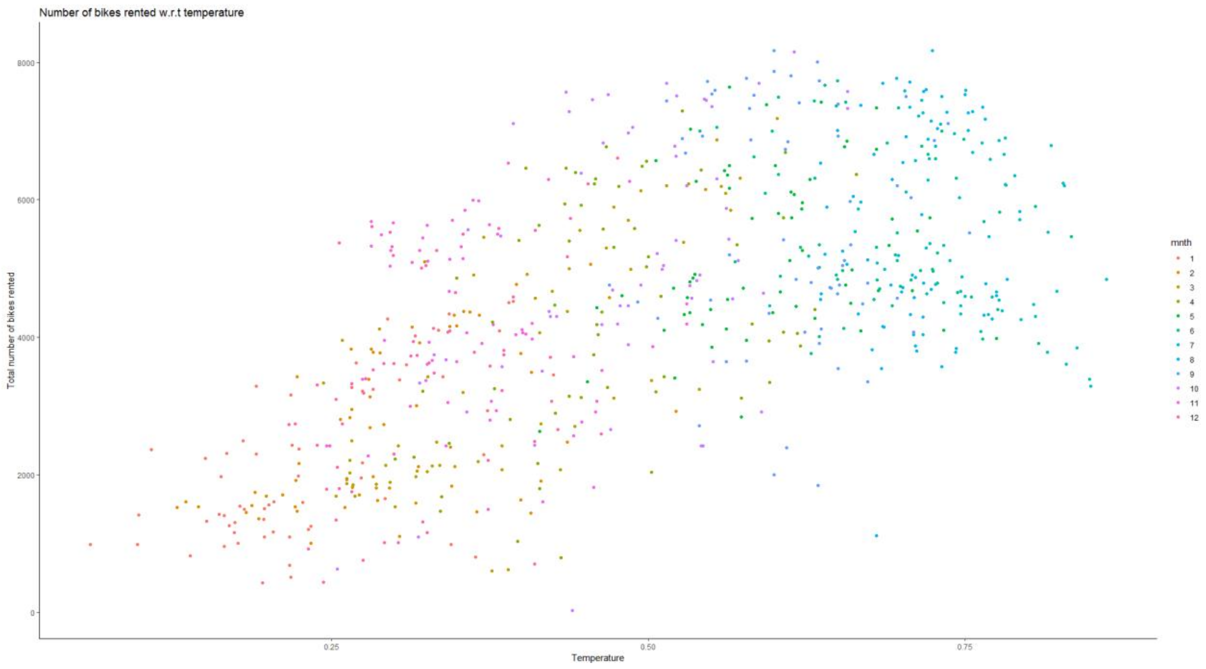


Fig 2.22: Scatterplot of count w.r.t temperature

This scatterplot shows that more bikes are rented when the temperature is high. So, bike renting increases with the increase in temperature.

Similarly, all the remaining inferences can be found in the Appendix A.

Modelling

Model Selection

The dependant variable can fall in either of the two categories:

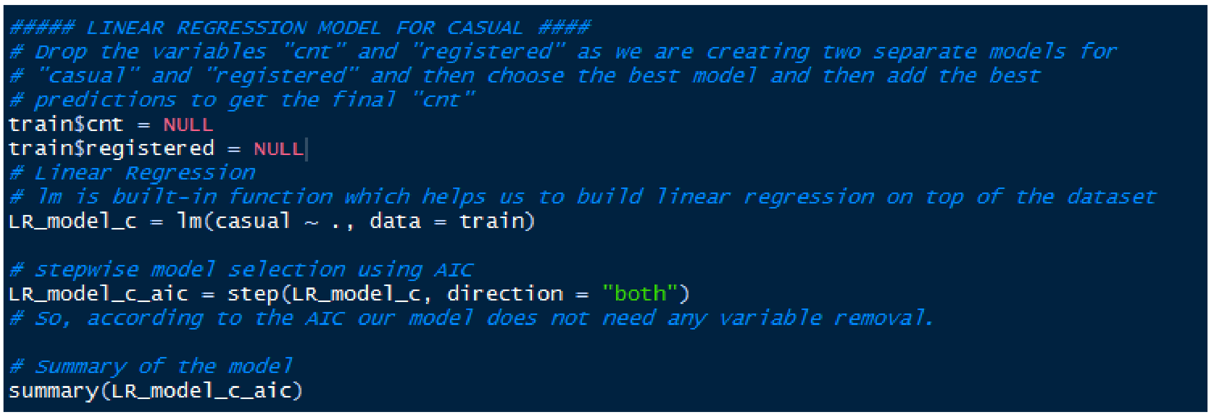
1. Categorical
2. Numerical

If the dependant variables, in our case *"casual" "registered" and "cnt"*, would have been *Categorical* the only predictive analysis that we could have performed is **Classification**. But the dependant variables we are dealing with are *Numerical*, for which we'll be doing **Regression**.

Here we'll start our model building from the simplest to more complex model. So, for our regression problem the first model will be linear regression.

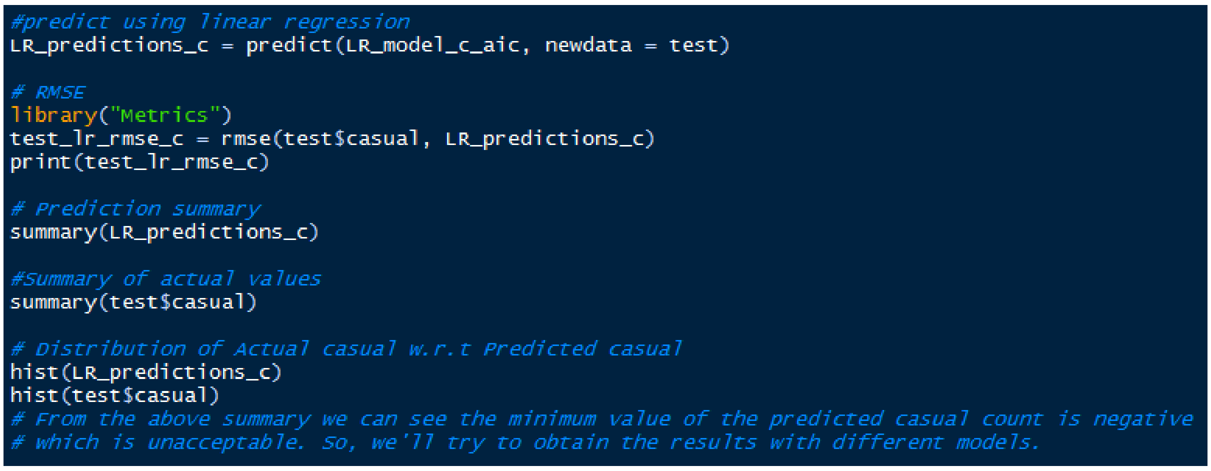
Linear Regression

Casual Count



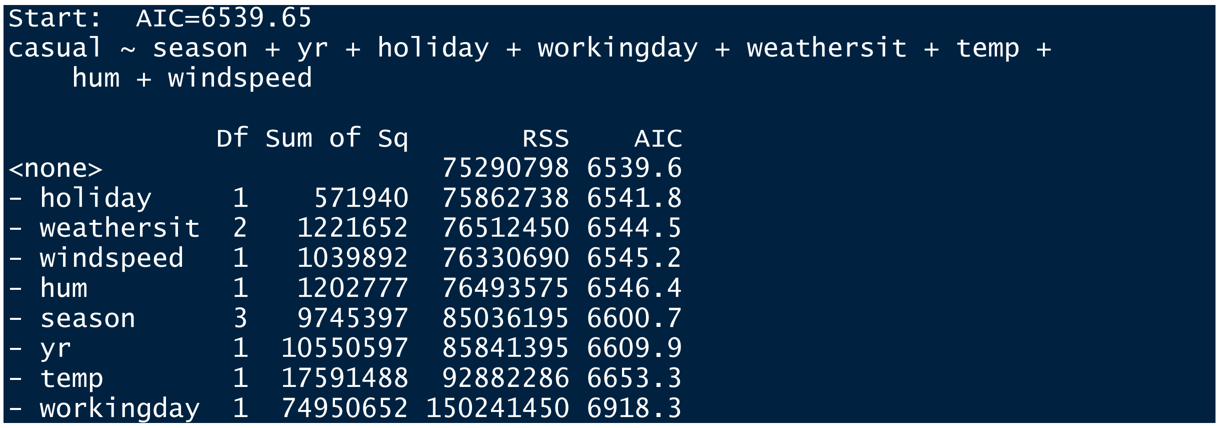
```
##### LINEAR REGRESSION MODEL FOR CASUAL ####
# Drop the variables "cnt" and "registered" as we are creating two separate models for
# "casual" and "registered" and then choose the best model and then add the best
# predictions to get the final "cnt"
train$cnt = NULL
train$registered = NULL
# Linear Regression
# lm is built-in function which helps us to build linear regression on top of the dataset
LR_model_c = lm(casual ~ ., data = train)

# stepwise model selection using AIC
LR_model_c_aic = step(LR_model_c, direction = "both")
# So, according to the AIC our model does not need any variable removal.

# Summary of the model
summary(LR_model_c_aic)
```

Fig 2.23: Code Snippet for Linear Regression model for "casual" variable



```
#predict using linear regression
LR_predictions_c = predict(LR_model_c_aic, newdata = test)

# RMSE
library("Metrics")
test_lr_rmse_c = rmse(test$casual, LR_predictions_c)
print(test_lr_rmse_c)

# Prediction summary
summary(LR_predictions_c)

#Summary of actual values
summary(test$casual)

# Distribution of Actual casual w.r.t Predicted casual
hist(LR_predictions_c)
hist(test$casual)
# From the above summary we can see the minimum value of the predicted casual count is negative
# which is unacceptable. So, we'll try to obtain the results with different models.
```

Fig 2.24: Code Snippet for prediction using Linear Regression for "casual" variable



```
Start:  AIC=6539.65
casual ~ season + yr + holiday + workingday + weathersit + temp + 
    hum + windspeed

             Df Sum of Sq       RSS    AIC
<none>                     75290798 6539.6
- holiday     1    571940  75862738 6541.8
- weathersit  2   1221652  76512450 6544.5
- windspeed   1   1039892  76330690 6545.2
- hum         1   1202777  76493575 6546.4
- season      3   9745397  85036195 6600.7
- yr          1  10550597  85841395 6609.9
- temp        1  17591488  92882286 6653.3
- workingday  1  74950652 150241450 6918.3
```

Fig 2.25: Output of the AIC on the Linear Regression Model for "casual" variable

```
Call:
lm(formula = casual ~ season + yr + holiday + workingday + weathersit +
    temp + hum + windspeed, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-1034.00  -211.91   -33.99   185.43  1608.88

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   644.05     121.41   5.305 1.65e-07 ***
season2       408.74      59.72   6.845 2.09e-11 ***
season3       175.55      79.01   2.222  0.02670 *
season4       238.49      51.22   4.656 4.06e-06 ***
yr1           283.93      32.67   8.691  < 2e-16 ***
holiday1     -204.75     101.19  -2.023  0.04352 *
workingday1  -820.69      35.43 -23.164  < 2e-16 ***
weathersit2  -108.18      43.12  -2.509  0.01241 *
weathersit3  -268.31     111.73  -2.401  0.01667 *
temp         1792.13     159.70  11.222  < 2e-16 ***
hum          -469.92     160.14  -2.934  0.00348 **
windspeed    -651.22     238.68  -2.728  0.00657 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 373.7 on 539 degrees of freedom
Multiple R-squared:  0.7196,	Adjusted R-squared:  0.7139
F-statistic: 125.7 on 11 and 539 DF,  p-value: < 2.2e-16
```

Fig 2.26: Output of Summary of Linear Regression Model for “casual” variable

In fig 2.26, the linear regression test's result for *“casual”* variable suggests us that the most important variables are *"yr-1"*, *"workingday-1"* and *"temp"* followed by *"season-2"*.

Here, in fig 2.25 the AIC suggests us that no variables from the model are dropped.

```
> # Prediction summary
> summary(LR_predictions_c)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 -398.8   474.5   911.2   849.3  1174.7  2069.4
> #Summary of actual values
> summary(test$casual)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    9.0   317.8   742.5   837.7  1061.0  3155.0
```

Fig 2.27: Summary of the actual count data vs predicted data for “casual” variable

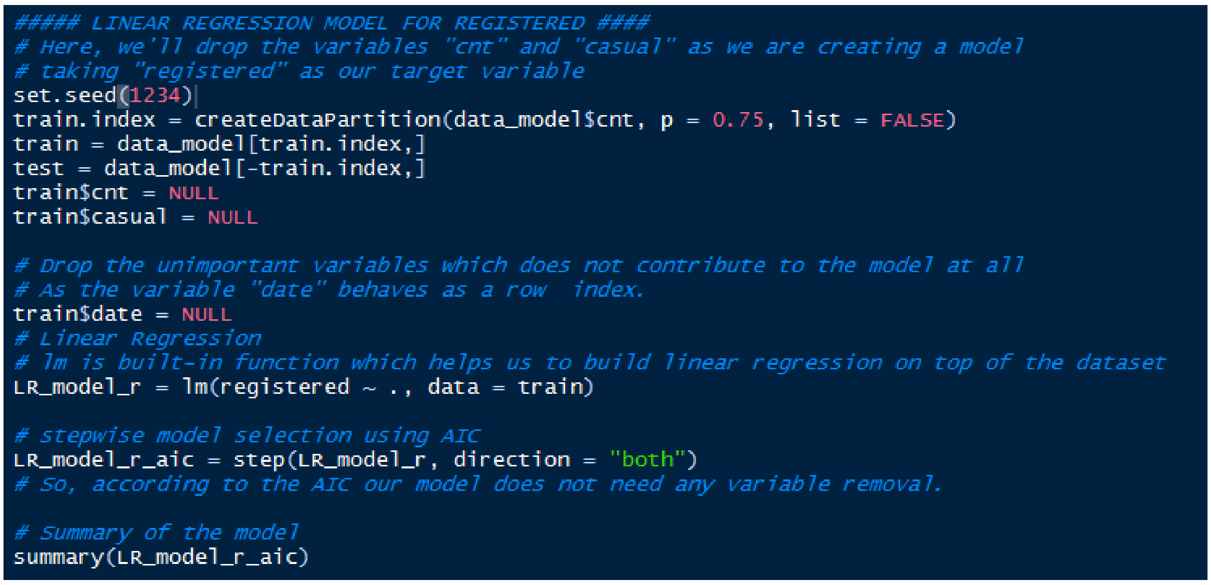
```
> test_lr_rmse_c = rmse(test$casual, LR_predictions_c)
> print(test_lr_rmse_c)
[1] 380.9494
```

Fig 2.28: RMSE of linear regression model for “casual” variable

Analysing the fig 2.27 we can say that the minimum value of the predicted *“casual”* count is negative which is unacceptable. So, we’ll try to obtain the results with different models.

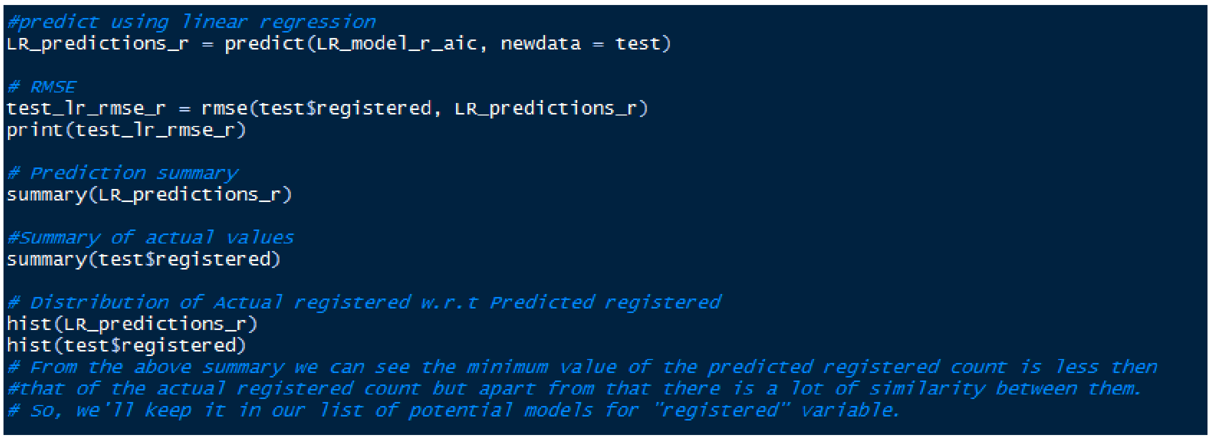
Now, we’ll build the Linear Regression model for the *“registered”* variable.

Registered Count



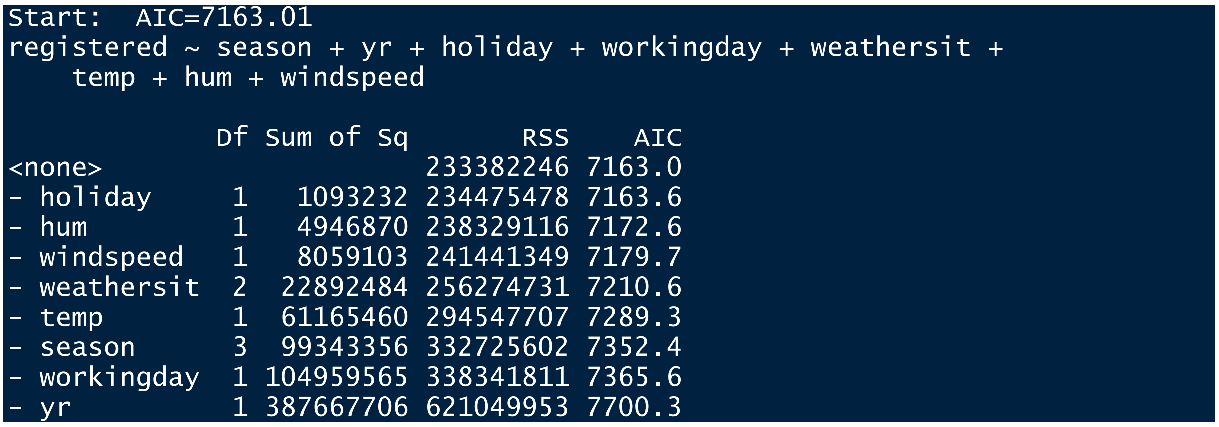
```
##### LINEAR REGRESSION MODEL FOR REGISTERED ####
# Here, we'll drop the variables "cnt" and "casual" as we are creating a model
# taking "registered" as our target variable
set.seed(1234)
train.index = createDataPartition(data_model$cnt, p = 0.75, list = FALSE)
train = data_model[train.index,]
test = data_model[-train.index,]
train$cnt = NULL
train$casual = NULL

# Drop the unimportant variables which does not contribute to the model at all
# As the variable "date" behaves as a row  index.
train$date = NULL
# Linear Regression
# lm is built-in function which helps us to build linear regression on top of the dataset
LR_model_r = lm(registered ~ ., data = train)

# stepwise model selection using AIC
LR_model_r_aic = step(LR_model_r, direction = "both")
# So, according to the AIC our model does not need any variable removal.

# Summary of the model
summary(LR_model_r_aic)
```

Fig 2.29: Code Snippet for Linear Regression model for "registered" variable



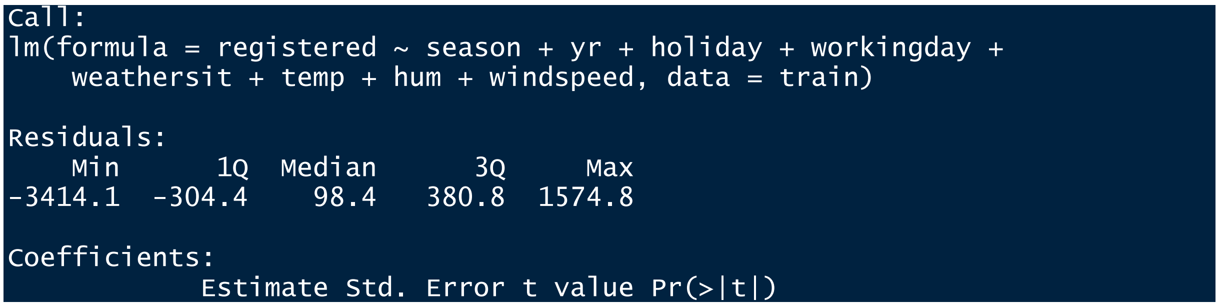
```
#predict using linear regression
LR_predictions_r = predict(LR_model_r_aic, newdata = test)

# RMSE
test_lr_rmse_r = rmse(test$registered, LR_predictions_r)
print(test_lr_rmse_r)

# Prediction summary
summary(LR_predictions_r)

#Summary of actual values
summary(test$registered)

# Distribution of Actual registered w.r.t Predicted registered
hist(LR_predictions_r)
hist(test$registered)
# From the above summary we can see the minimum value of the predicted registered count is less then
#that of the actual registered count but apart from that there is a lot of similarity between them.
# So, we'll keep it in our list of potential models for "registered" variable.
```

Fig 2.30: Code Snippet for prediction using Linear Regression for "registered" variable

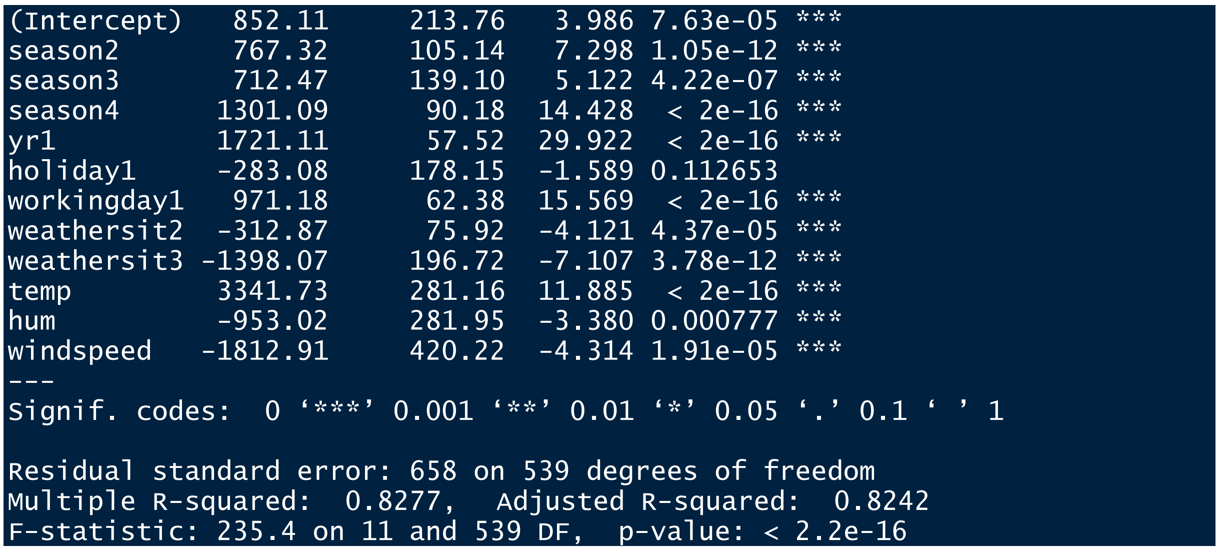


```
Start:  AIC=7163.01
registered ~ season + yr + holiday + workingday + weathersit +
    temp + hum + windspeed

             Df Sum of Sq       RSS    AIC
<none>                    233382246 7163.0
- holiday     1   1093232 234475478 7163.6
- hum         1   4946870 238329116 7172.6
- windspeed   1   8059103 241441349 7179.7
- weathersit  2  22892484 256274731 7210.6
- temp        1  61165460 294547707 7289.3
- season      3  99343356 332725602 7352.4
- workingday  1 104959565 338341811 7365.6
- yr          1 387667706 621049953 7700.3
```

Fig 2.31: Output of the AIC on the Linear Regression Model for "registered" variable



```
Call:
lm(formula = registered ~ season + yr + holiday + workingday +
    weathersit + temp + hum + windspeed, data = train)

Residuals:
    Min      1Q  Median      3Q     Max 
-3414.1  -304.4    98.4   380.8  1574.8 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
```

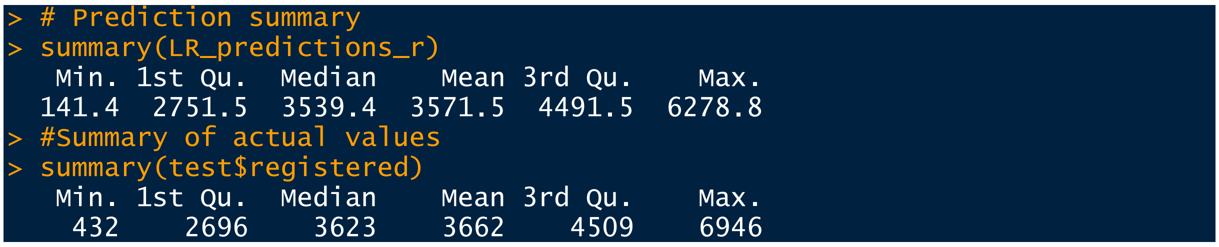


```
(Intercept)   852.11     213.76   3.986 7.63e-05 ***
season2       767.32     105.14   7.298 1.05e-12 ***
season3       712.47     139.10   5.122 4.22e-07 ***
season4      1301.09      90.18  14.428  < 2e-16 ***
yr1          1721.11      57.52  29.922  < 2e-16 ***
holiday1     -283.08     178.15  -1.589 0.112653
workingday1   971.18      62.38  15.569  < 2e-16 ***
weathersit2  -312.87      75.92  -4.121 4.37e-05 ***
weathersit3 -1398.07     196.72  -7.107 3.78e-12 ***
temp         3341.73     281.16  11.885  < 2e-16 ***
hum          -953.02     281.95  -3.380 0.000777 ***
windspeed   -1812.91     420.22  -4.314 1.91e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 658 on 539 degrees of freedom
Multiple R-squared:  0.8277,   Adjusted R-squared:  0.8242
F-statistic: 235.4 on 11 and 539 DF,  p-value: < 2.2e-16
```

Fig 2.32: Output of Summary of Linear Regression Model for "registered" variable

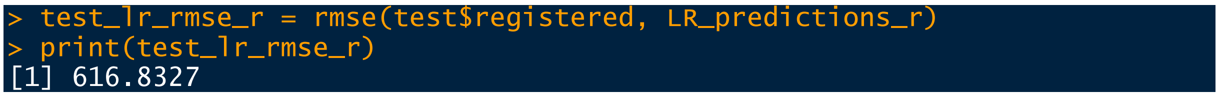
In fig 2.32, the linear regression test's result for *"registered"* variable suggests us that the most important variables are *"season-4"*, *"yr-1"*, *"workingday-1"* and *"temp"* followed by *"season-2"*.

Here, in fig 2.31 the AIC suggests us that no variables from the model are dropped.



```
> # Prediction summary
> summary(LR_predictions_r)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  141.4  2751.5  3539.4  3571.5  4491.5  6278.8 
> #Summary of actual values
> summary(test$registered)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    432    2696    3623    3662    4509    6946 
```

Fig 2.33: Summary of the actual count data vs predicted data for "registered" variable



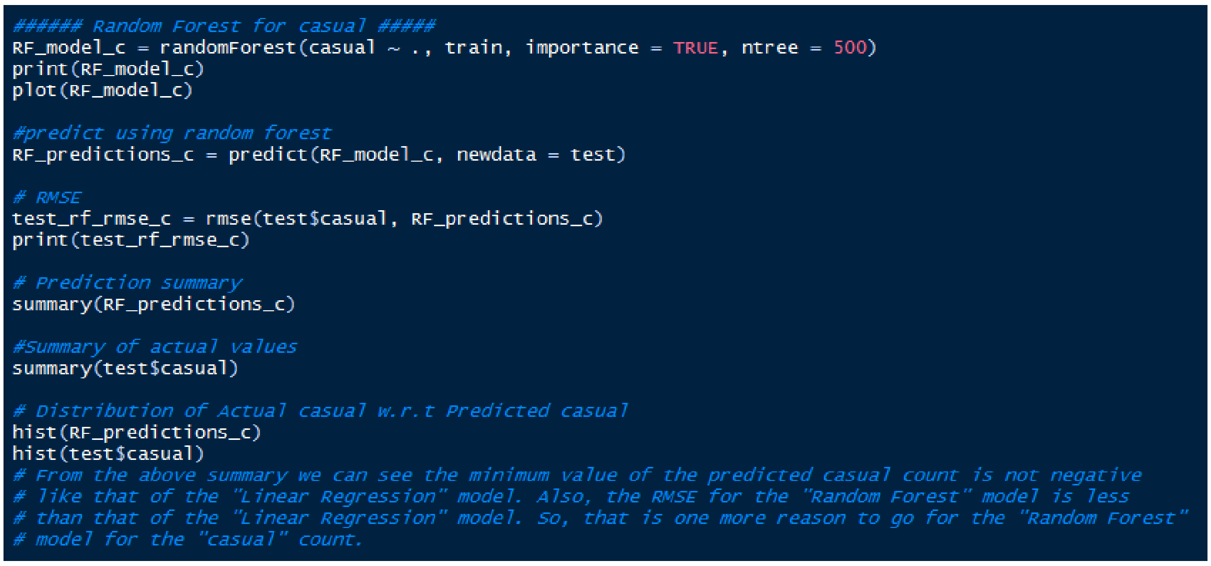
```
> test_lr_rmse_r = rmse(test$registered, LR_predictions_r)
> print(test_lr_rmse_r)
[1] 616.8327
```

Fig 2.34: RMSE of linear regression model for "registered" variable

Analysing the fig 2.33 we can say that the minimum value of the predicted *"registered"* count is less then that of the actual registered count but apart from that there is a lot of similarity between them. So, we'll keep it in our list of potential models for "registered" variable.

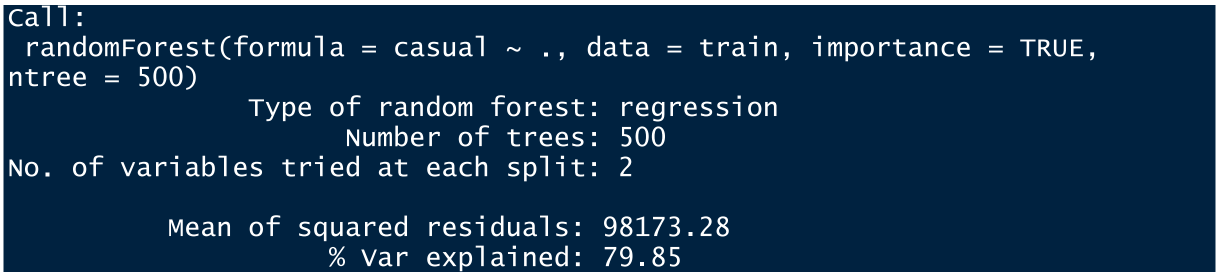
Random Forest

Casual Count



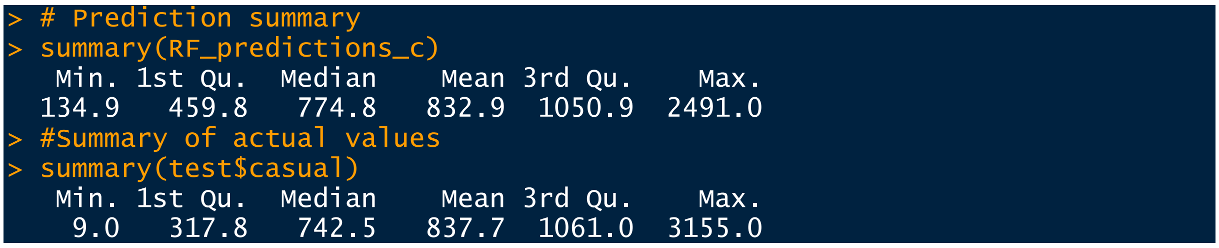
```
###### Random Forest for casual #####
RF_model_c = randomForest(casual ~ ., train, importance = TRUE, ntree = 500)
print(RF_model_c)
plot(RF_model_c)

#predict using random forest
RF_predictions_c = predict(RF_model_c, newdata = test)

# RMSE
test_rf_rmse_c = rmse(test$casual, RF_predictions_c)
print(test_rf_rmse_c)

# Prediction summary
summary(RF_predictions_c)

#Summary of actual values
summary(test$casual)

# Distribution of Actual casual w.r.t Predicted casual
hist(RF_predictions_c)
hist(test$casual)
# From the above summary we can see the minimum value of the predicted casual count is not negative
# like that of the "Linear Regression" model. Also, the RMSE for the "Random Forest" model is less
# than that of the "Linear Regression" model. So, that is one more reason to go for the "Random Forest"
# model for the "casual" count.
```

Fig 2.33: Code Snippet for Random Forest model implementation for “casual” variable



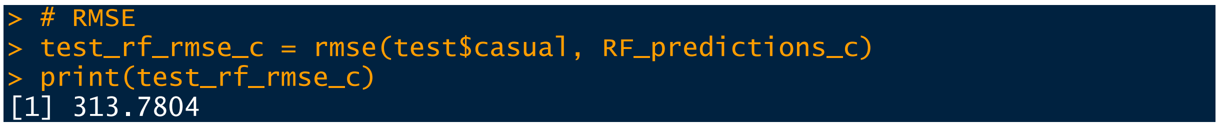
```
Call:
 randomForest(formula = casual ~ ., data = train, importance = TRUE,
ntree = 500)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 98173.28
                    % Var explained: 79.85
```

Fig 2.34: Random Forest model’s Summary for “casual” variable



```
> # Prediction summary
> summary(RF_predictions_c)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  134.9   459.8   774.8   832.9  1050.9  2491.0
> #Summary of actual values
> summary(test$casual)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    9.0   317.8   742.5   837.7  1061.0  3155.0
```

Fig 2.35: Summary of the actual count data vs predicted data for “casual” variable

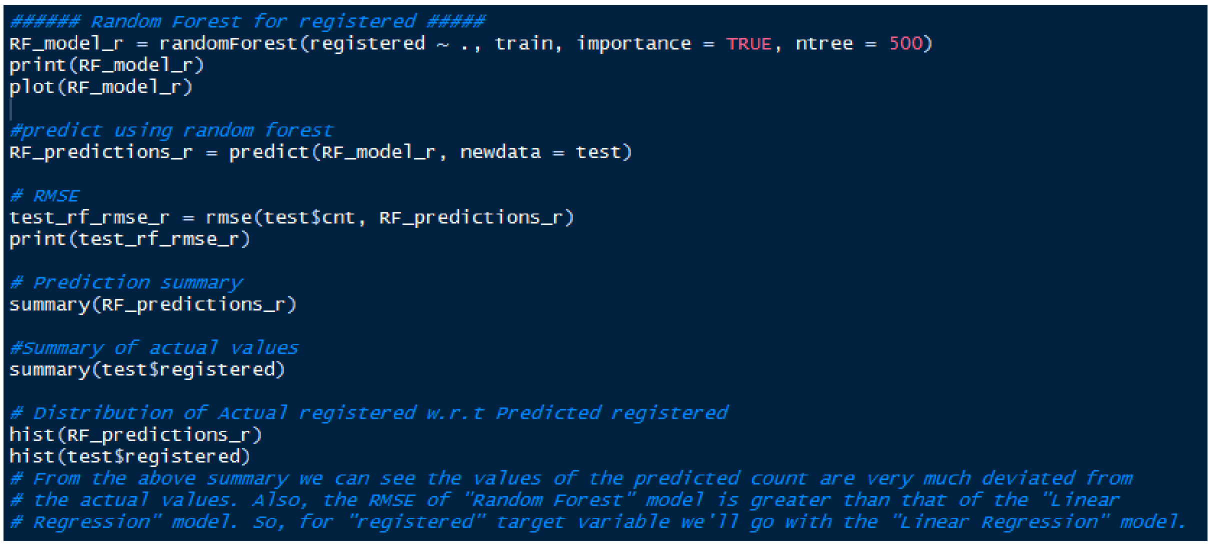


```
> # RMSE
> test_rf_rmse_c = rmse(test$casual, RF_predictions_c)
> print(test_rf_rmse_c)
[1] 313.7804
```

Fig 2.36: RMSE of random forest model for “casual” variable

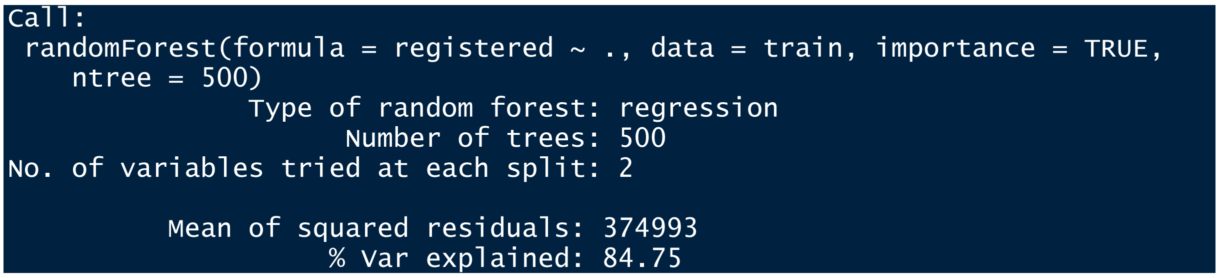
From the summary in fig 2.35 we can see the minimum value of the predicted *“casual”* count is not negative like that of the "Linear Regression" model. Also, the RMSE for the "Random Forest" model is less than that of the "Linear Regression" model. So, that is one more reason to go for the "Random Forest" model for the *"casual"* count.

Registered Count



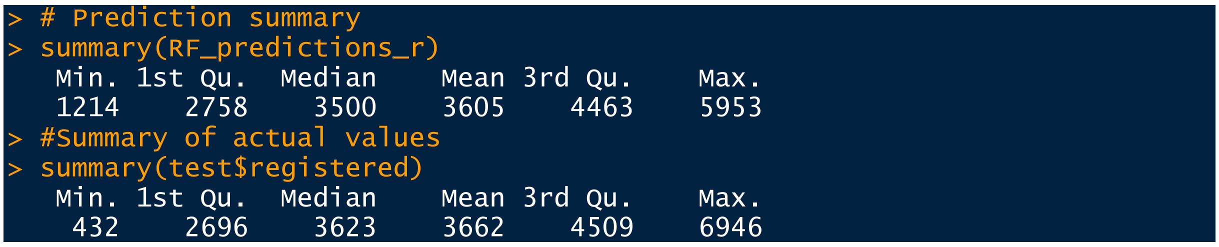
```
###### Random Forest for registered #####
RF_model_r = randomForest(registered ~ ., train, importance = TRUE, ntree = 500)
print(RF_model_r)
plot(RF_model_r)

#predict using random forest
RF_predictions_r = predict(RF_model_r, newdata = test)

# RMSE
test_rf_rmse_r = rmse(test$cnt, RF_predictions_r)
print(test_rf_rmse_r)

# Prediction summary
summary(RF_predictions_r)

#Summary of actual values
summary(test$registered)

# Distribution of Actual registered w.r.t Predicted registered
hist(RF_predictions_r)
hist(test$registered)
# From the above summary we can see the values of the predicted count are very much deviated from
# the actual values. Also, the RMSE of "Random Forest" model is greater than that of the "Linear
# Regression" model. So, for "registered" target variable we'll go with the "Linear Regression" model.
```

Fig 2.37: Code Snippet for Random Forest model implementation for “registered” variable



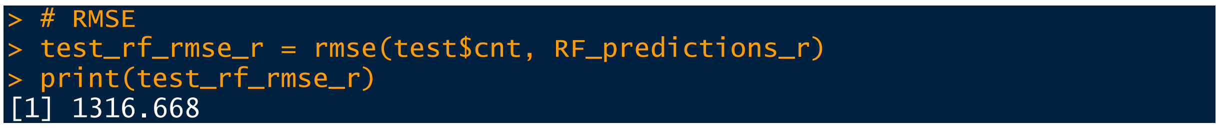
```
Call:
 randomForest(formula = registered ~ ., data = train, importance = TRUE,
    ntree = 500)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 2

          Mean of squared residuals: 374993
                    % Var explained: 84.75
```

Fig 2.38: Random Forest model’s Summary for “registered” variable



```
> # Prediction summary
> summary(RF_predictions_r)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1214    2758    3500    3605    4463    5953
> #Summary of actual values
> summary(test$registered)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    432    2696    3623    3662    4509    6946
```

Fig 2.39: Summary of the actual count data vs predicted data for “registered” variable



```
> # RMSE
> test_rf_rmse_r = rmse(test$cnt, RF_predictions_r)
> print(test_rf_rmse_r)
[1] 1316.668
```

Fig 2.40: RMSE of random forest model for “registered” variable

From the summary in fig 2.39 we can see the values of the predicted *“registered”* count are very much deviated from the actual values. Also, the RMSE of "Random Forest" model is greater than that of the "Linear Regression" model. So, for *"registered"* target variable we'll go with the "Linear Regression" model.

Chapter 3

Conclusion

Model Evaluation

Now that we have four models to predict the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive Performance
2. Interpretability
3. Computational Efficiency

In Our case of Bike Rentals Prediction, the last one, Computational Efficiency, do not hold much significance as the most important aspect in our business problem is to predict the bike rental counts and make business decisions keeping in mind the important factors which aid in bike rentals count to increase, not to build a model which can do faster predictions.

Therefore, we will use Predictive Performance and Interpretability to compare the two models.

Predictive Performance

The only thing we check here is the accuracy of the model and the considered error metric to train and make the predictions for the test data.

For *"registered"* variable the *Linear Regression* model is more accurate than *Random Forest* as the RMSE value for random forest is higher than that of the linear regression. So, we'll go with the random forest model for *"registered"* variable.

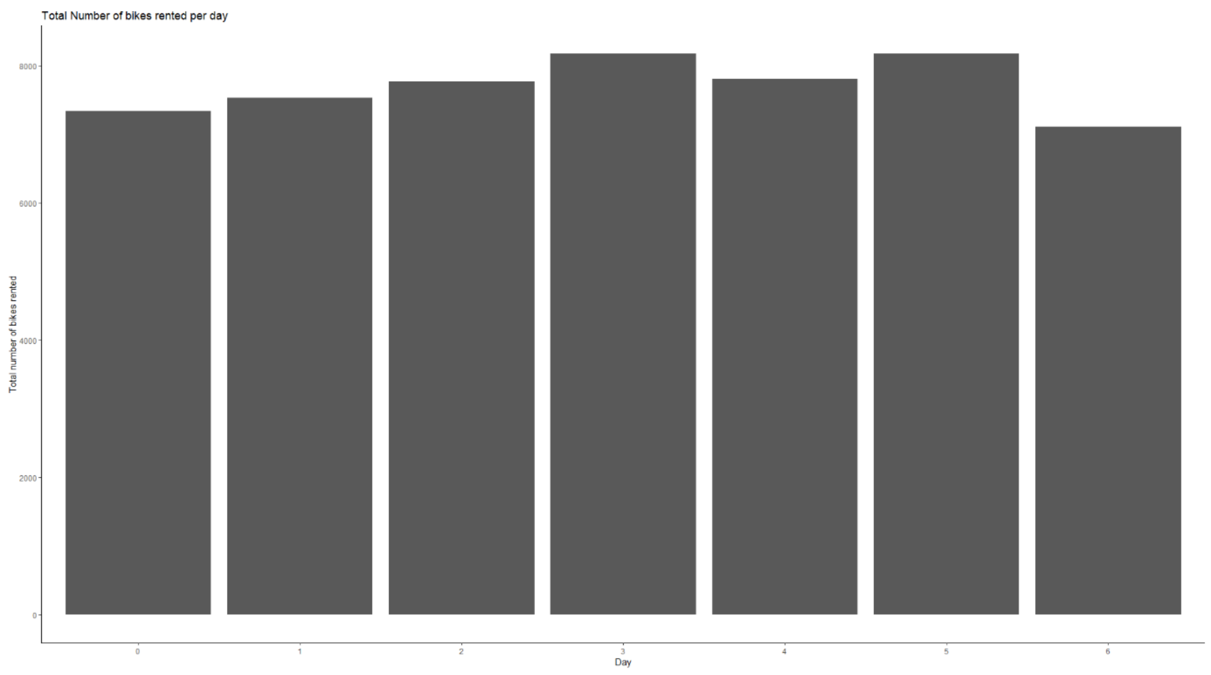
For *"casual"* variable the *Random Forest* model is more accurate than *Linear Regression* as the RMSE value for linear regression is higher than that of the random forest. So, we'll go with the linear regression model for *"casual"* variable.

Interpretability

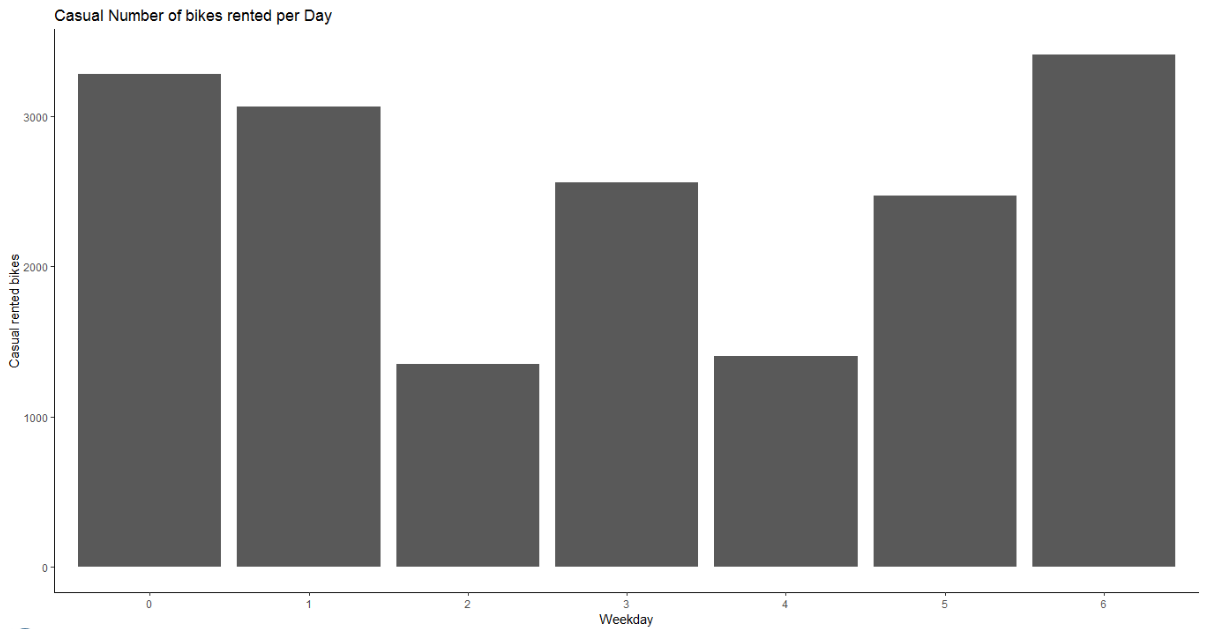
The only thing we check here is how close the predcited values are to the actual values. We, clearly saw in the summary of both the models that for *"casual"* variable the linear regression model's predicted values are negative which are unacceptable for *"cnt"* variable. For the "registered" variable, there is huge deviation in the predicted values of the random forest model whereas the predicted values for the linear regression model are very much similar to that of the actual values.

So, considering both *Predictive Performance* and *Interpretability* we will choose *Linear Regression* for *"registered"* variable and *Random Forest* for *"casual"* variable.

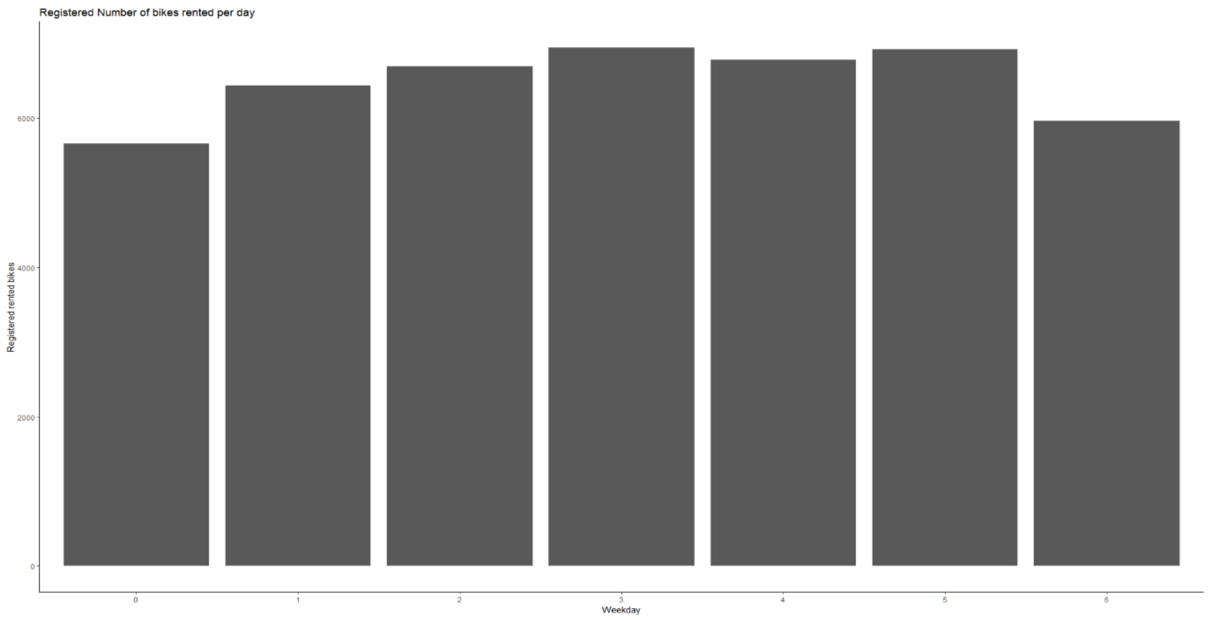
Appendix A – Extra Figures



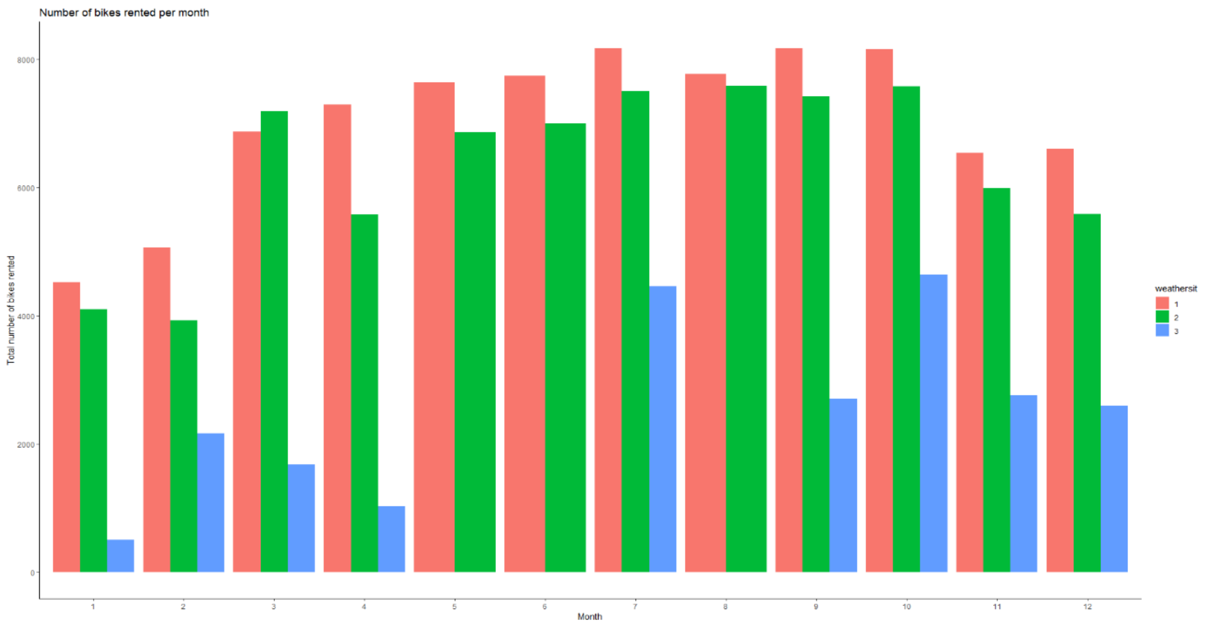
Variation of total count w.r.t. weekday



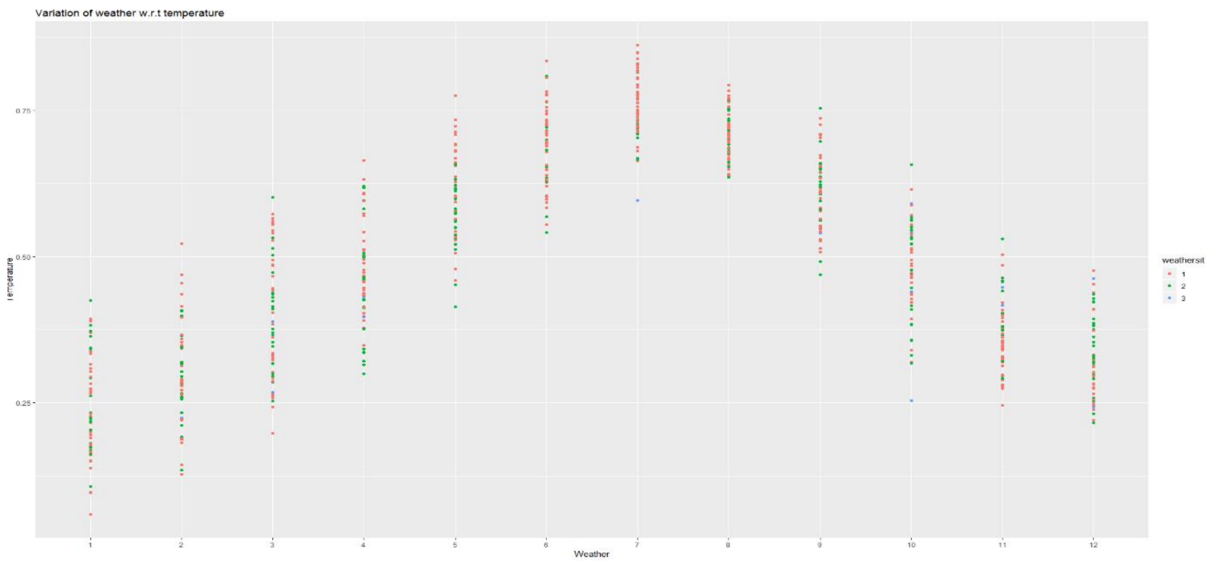
Variation of casual count w.r.t. weekday



Variation of registered count w.r.t. weekday



Variation of total count w.r.t. month and weather situation



Variation of temperature w.r.t. month and weather situation

Appendix B – R Code

```
# Clear the environment
rm(list = ls())

#importing the required libraries
library("ggplot2")
library("corrgram")
library("randomForest")
library("C50")

# Load the file
bike_data = read.csv("day.csv")
date_data = bike_data$dteday

# Explore the data
str(bike_data)

###### univariate analysis #####
# Convert the dteday column into date format
bike_data$dteday = as.Date(bike_data$dteday)

# Convert the variables from integer class to factor
# As, it is better to convert the variable's data
# in the form of a factor as it improves the efficiency of the code
bike_data$season = as.factor(bike_data$season)
bike_data$yr = as.factor(bike_data$yr)
bike_data$mnth = as.factor(bike_data$mnth)
bike_data$holiday = as.factor(bike_data$holiday)
bike_data$weekday = as.factor(bike_data$weekday)
bike_data$workingday = as.factor(bike_data$workingday)
bike_data$weathersit = as.factor(bike_data$weathersit)

###### Pre-Proecessing #####

# Check for missing values
sapply(bike_data, function(x) sum(is.na(x)))
# SO there are no missing values in the dataset

# Drop the unimportant variables which does not contribute to the model at all
# As the instant variable is just row index
bike_data$instant = NULL

# Here, we are separating the numerical variables
numeric_index = sapply(bike_data, is.numeric)
#This selects only those columns which are numeric

numeric_data = bike_data[,numeric_index]
#This transfers the data of the columns which are numeric

cnames = colnames(numeric_data)
cnames = cnames[-7]

# We can also, plot the probability density functions of the numerical variables
# to check their distribution and also skewness
old_par = par("mar")
par(mar = c(1,1,1,1))
library(psych)
multi.hist(bike_data[,sapply(bike_data, is.numeric)], main = NA, dcol = c("blue", "red"),
        dlty = c("solid", "solid"), bcol = "linen")
```

```
par(mar = old_par)

###### Outlier analysis #####
#Boxplots - Distribution & Outlier Check

# Checking the presence of outliers in the variables w.r.t to a suitable variable
#Box-plot
#Here continuous var is on y-axis.
library("scales")
ggplot(bike_data, aes_string(x = bike_data$season, y = bike_data$temp)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Season") + ylab('Temperature') +ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$season, y = bike_data$atemp)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Season") + ylab('Feeling Temperature') +ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$weathersit, y = bike_data$hum)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Weather Situation") + ylab('Humidity') +ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$weathersit, y = bike_data$windspeed)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Weather Situation") + ylab('Windspeed') + ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$mnth, y = bike_data$casual)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Month") + ylab('Count of Casual Riders') + ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$mnth, y = bike_data$registered)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Month") + ylab('Count of Registered Riders') + ggtitle("Outlier Analysis")


#Repalce all the outliers with NA and Impute
#Create NA on custAge
for(i in cnames){
  val = numeric_data[,i][numeric_data[,i] %in% boxplot.stats(numeric_data[,i])$out]
  numeric_data[,i][numeric_data[,i] %in% val] = NA
}

library("DMwR")
numeric_data = knnImputation(numeric_data, k = 3)
summary(numeric_data$hum)
summary(numeric_data$windspeed)

# Now replacing the imputed data back into the data
summary(bike_data$hum)
summary(bike_data$windspeed)

bike_data$hum = numeric_data$hum
bike_data$windspeed = numeric_data$windspeed

# Now plotting the probabilisty density functions after performing the Outlier Analysis
old_par = par("mar")
par(mar = c(1,1,1,1))
library(psych)
multi.hist(bike_data[,sapply(bike_data, is.numeric)], main = NA, dcol = c("blue", "red"),
           dlty = c("solid", "solid"), bcol = "linen")
par(mar = old_par)
```

```
# Also, plotting the boxplots after performing Outlier Analysis
ggplot(bike_data, aes_string(x = bike_data$weathersit, y = bike_data$hum)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Weather Situation") + ylab('Humidity') +ggtitle("Outlier Analysis")

ggplot(bike_data, aes_string(x = bike_data$weathersit, y = bike_data$windspeed)) +
  geom_boxplot(outlier.color = "red", outlier.size = 3) + theme_bw() +
  xlab("Weather Situation") + ylab('Windspeed') + ggtitle("Outlier Analysis")


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

# Creating the correlation analysis plot to analyse the dependecies of numeric
# variables
library(corrplot)
cor_plot = cor(bike_data[,sapply(bike_data, is.numeric)])
corrplot(cor_plot, method = "color", addgrid.col = "darkgray", addCoef.col = "black",
         outline = TRUE, number.digits = 2)
# The correlation plot tells us that the numerical variables "atemp" - "temp" and "registered"-"cnt"
# are correlated. So, we can drop "atemp" variable. As "cnt" is our target variable
# we want the independant variables to have high correlation with it.

# Now, we'll plot various visualizations to see the variation of a variable w.r.t other variable
# First, plotting the variation of the casual bike count w.r.t yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$casual, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented by casual users")

# Second, plotting the variation of the casual bike count w.r.t yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$registered, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented by registered users")

# Third, plotting the variation of the total bike count w.r.t the yr and season
ggplot(data=bike_data,aes(x= bike_data$yr, y = bike_data$cnt, fill = season)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(x="Year", y="Total number of bikes rented")
# On the basis of these 3 plots we can say surely say few things:
# 1. There is an increase in bike renting from year 2011 to year 2012
# 2. Bikes are least rented in Spring season and most rented in Fall season
# 3. There is a sudden increase in bike renting during summer


# Plotting to check which day has the highest number of rents
ggplot(data=bike_data,aes(x= bike_data$weekday, y = bike_data$cnt)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(title="Total Number of bikes rented per day", x="Day", y="Total number of bikes rented")

# Variation of casual renting w.r.t the weekday
ggplot(data=bike_data,aes(x= bike_data$weekday, y = bike_data$casual)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(title="Casual Number of bikes rented per Day", x="Weekday", y="Casual rented bikes")
# So, there are more casual users on the weekends

# Variation of registered renting w.r.t the weekday
ggplot(data=bike_data,aes(x= bike_data$weekday, y = bike_data$registered)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(title="Registered Number of bikes rented per day", x="Weekday", y="Registered rented bikes")
# So, there are more registered users on the weekdays
# The distribution of count is similar for every day of the week, so if there is a little
# dependency between the weekday and any other dependent variable then we can drop weekday variable

# Plotting to check which month has the highest number of rents also with the variation of the weather
ggplot(data=bike_data,aes(x= bike_data$mnth, y = bike_data$cnt, fill = weathersit)) +
  geom_bar(stat = "identity", position = position_dodge()) +
  theme_classic() +
  labs(title="Number of bikes rented per month", x="Month", y="Total number of bikes rented")
# On the basis of the abaove plot we can say that the bikes are most rented when the weather is 1(Clear,
Few Clouds)
# and least bikes are rented when the weather is 3 and no bikes are rented when the weather is
# 4(Heavy Rain + Ice Pallets + Thunderstorm)

# So lets check the dependency between the month and weather situation
ggplot(data=bike_data,aes(x= bike_data$mnth, y = bike_data$temp)) +
  geom_point(stat = "identity", aes(color = weathersit)) +
  labs(title="Variation of weather w.r.t temperature", x="Weather", y="Temperature")
# High temperatures are obtained in the month of 6,7 & 8 which is obvious. Also, there is a high
probability
# of 1 weather situation in every month but the probability of 3 weather situation is high
# in the month of 4,5,6,7 & in 10, 11, 12.

# Plotting to check the number of rents w.r.t variation of the temperature
ggplot(data=bike_data,aes(x= bike_data$temp, y = bike_data$cnt)) +
  geom_point(aes(color = mnth)) +
  theme_classic() +
  labs(title="Number of bikes rented w.r.t temperature", x="Temperature", y="Total number of bikes
rented")
# On the basis of the above plot we can surely say that
# more bikes are rented when the temperature is high.

#Chi-Squared test of independence
factor_index = sapply(bike_data, is.factor)
#In the previous we selected only the categorical variables

factor_data = bike_data[,factor_index]

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$season, factor_data[,i])))
}


for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$yr, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$mnth, factor_data[,i])))
}


for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$holiday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$weekday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$workingday, factor_data[,i])))
}

for (i in 1:7){
  print(names(factor_data)[i])
  print(chisq.test(table(factor_data$weathersit, factor_data[,i])))
}
# According, to the chi-square test the "mnth" variable has very high dependency with "season" and
"weekday" variable
# has very high dependency with "workingday". Thus they can be dropped.

#Dimension reduction
data_del = subset(bike_data, select = -c(dteday, mnth, weekday, atemp))


# Clear the environment except data_del
library(DataCombine)
rmExcept(c("data_del","bike_data", "date_data"))

##### MODEL DEVELOPMENT ####
data_model = data_del
data_model$date = date_data

# Divide the data into train and test using stratified sampling method
# createdatapartition is one more function used to create stratified samples
# Here in createdatapartiotion function 1st arg is of the reference variable on the basis
# of which splitting will take place, 2nd arg is the percentage of observation we need
# in our sample, list = false implies that there will no repetitive observations
library("caret")
set.seed(1234)
train.index = createDataPartition(data_model$cnt, p = 0.75, list = FALSE)
train = data_model[train.index,]
test = data_model[-train.index,]

# Drop the unimportant variables which does not contribute to the model at all
# As the variable "date" behaves as a row  index.
train$date = NULL

##### LINEAR REGRESSION MODEL FOR CASUAL ####
# Drop the variables "cnt" and "registered" as we are creating two separate models for
# "casual" and "registered" and then choose the best model and then add the best
# predictions to get the final "cnt"
train$cnt = NULL
train$registered = NULL
# Linear Regression
# lm is built-in function which helps us to build linear regression on top of the dataset
LR_model_c = lm(casual ~ ., data = train)

# stepwise model selection using AIC
LR_model_c_aic = step(LR_model_c, direction = "both")
# So, according to the AIC our model does not need any variable removal.
# Summary of the model
summary(LR_model_c_aic)

#predict using linear regression
LR_predictions_c = predict(LR_model_c_aic, newdata = test)

# RMSE
library("Metrics")
test_lr_rmse_c = rmse(test$casual, LR_predictions_c)
print(test_lr_rmse_c)

# Prediction summary
summary(LR_predictions_c)

#Summary of actual values
summary(test$casual)

# Distribution of Actual casual w.r.t Predicted casual
hist(LR_predictions_c)
hist(test$casual)
# From the above summary we can see the minimum value of the predicted casual count is negative
# which is unacceptable. So, we'll try to obtain the results with different models.


###### Random Forest for casual #####
RF_model_c = randomForest(casual ~ ., train, importance = TRUE, ntree = 500)
print(RF_model_c)
plot(RF_model_c)

#predict using random forest
RF_predictions_c = predict(RF_model_c, newdata = test)

# RMSE
test_rf_rmse_c = rmse(test$casual, RF_predictions_c)
print(test_rf_rmse_c)

# Prediction summary
summary(RF_predictions_c)

#Summary of actual values
summary(test$casual)

# Distribution of Actual casual w.r.t Predicted casual
hist(RF_predictions_c)
hist(test$casual)
# From the above summary we can see the minimum value of the predicted casual count is not negative
# like that of the "Linear Regression" model. Also, the RMSE for the "Random Forest" model is less
# than that of the "Linear Regression" model. So, that is one more reason to go for the "Random Forest"
# model for the "casual" count.


##### LINEAR REGRESSION MODEL FOR REGISTERED ####
# Here, we'll drop the variables "cnt" and "casual" as we are creating a model
# taking "registered" as our target variable
set.seed(1234)
train.index = createDataPartition(data_model$cnt, p = 0.75, list = FALSE)
train = data_model[train.index,]
test = data_model[-train.index,]
train$cnt = NULL
train$casual = NULL
```

```
# Drop the unimportant variables which does not contribute to the model at all
# As the variable "date" behaves as a row  index.
train$date = NULL
# Linear Regression
# lm is built-in function which helps us to build linear regression on top of the dataset
LR_model_r = lm(registered ~ ., data = train)

# stepwise model selection using AIC
LR_model_r_aic = step(LR_model_r, direction = "both")
# So, according to the AIC our model does not need any variable removal.

# Summary of the model
summary(LR_model_r_aic)

#predict using linear regression
LR_predictions_r = predict(LR_model_r_aic, newdata = test)

# RMSE
test_lr_rmse_r = rmse(test$registered, LR_predictions_r)
print(test_lr_rmse_r)

# Prediction summary
summary(LR_predictions_r)

#Summary of actual values
summary(test$registered)

# Distribution of Actual registered w.r.t Predicted registered
hist(LR_predictions_r)
hist(test$registered)
# From the above summary we can see the minimum value of the predicted registered count is less then
# that of the actual registered count but apart from that there is a lot of similarity between them.
# So, we'll keep it in our list of potential models for "registered" variable.


###### Random Forest for registered #####
RF_model_r = randomForest(registered ~ ., train, importance = TRUE, ntree = 500)
print(RF_model_r)
plot(RF_model_r)

#predict using random forest
RF_predictions_r = predict(RF_model_r, newdata = test)

# RMSE
test_rf_rmse_r = rmse(test$cnt, RF_predictions_r)
print(test_rf_rmse_r)

# Prediction summary
summary(RF_predictions_r)

#Summary of actual values
summary(test$registered)

# Distribution of Actual registered w.r.t Predicted registered
hist(RF_predictions_r)
hist(test$registered)
# From the above summary we can see the values of the predicted count are very much deviated from
# the actual values. Also, the RMSE of "Random Forest" model is greater than that of the "Linear
# Regression" model. So, for "registered" target variable we'll go with the "Linear Regression" model.

# Add the best predictions from both the models
```

```
Final_predictions = RF_predictions_c + LR_predictions_r

# Save the results
results <- data.frame(date = test$date, count = Final_predictions)

# Write the results to a csv file
write.csv(results, file = 'Predictions.csv', row.names = FALSE, quote=FALSE)
```