FIT5149 S2 2020

Assessment 1: Predict Bike-Sharing need in Metropolitan Area

Student information

- Family Name: Aniruddha
- Given Name: Gayatri
- Student ID: 30945305
- Student email: gani0001@student.monash.edu (mailto:gani0001@student.monash.edu)

Programming Language: R 3.5.1 in Jupyter Notebook

R Libraries used:

- ggplot2
- corrplot
- car
- reshape2
- e1071
- stats
- scales
- grid
- gridExtra
- glmnet
- lattice
- repr
- lubridate

Table of Contents

1. Introduction

- This assignment revolves around predicting the hourly rental bike demands.
- The demands for bikes is based on data such as - weather, season, holiday etc.
- We have been provided with two datasets - the **training set and the testing set.**
- This dataset contains :
    - Information about the weather

  - Date Information
  - Number of bikes rented per hour
- There are around **8000 instances and 14 attributes.**
- Our **target is the "Rented Bike Count"** - the one that we are required to make our prediction on.
- We are doing this so that the correct number of bikes are available at the right place and at the right time!
- This is to address the following issues :
  - Allows more people to use a bike from one location and drop it at a different location.
  - Bike sharing is generally free of cost or at reasonable rates.
  - Managing the never-ending traffic problem in metros.
  - Making the city more eco-friendly and sustainable!
- We have performed the **following tasks** in this assignment in order to perform the prediction and inference task :
  - Basic Data Exploration
  - In-depth Exploratory Data Analysis : Identifying common patterns and trends.
  - Overview of the methodology followed to train the dataset.
  - Development of the models.
  - Comparison and justification of the choice of models.
  - Analysis, interpretation of the results and conclusion.

Following are the libraries used in the notebook:

In [ ]:

```
# To create complex plots
library(ggplot2)
# To plot the correlation matrix
library(corrplot)
# To create linear regression models
library(car)
# To use the Melt function
library(reshape2)
# For the SVM model
library(e1071)
# For the statistical functions
library(stats)
# For providing interal scaling infrastructure to ggplot2
library(scales)
# In order to produce a graphical output directly
library(grid)
# In order to perform a number of user-level functions
library(gridExtra)
#library(RColorBrewer)
# To create linear regression
library(glmnet)
# To produce multiple small plots
library(lattice)
# For string and byte representations
library(repr)
# For dealing with dates
library(lubridate)
```

2. Data Exploration

- This section contains a descriptive and comprehensive data exploration.

- I have provided in depth analysis with related plots and statistical methods.
- The plots and methods have been justified with proper explanations as well.
- The following are the steps performed in order to explore the data :
  - We load and read the training and testing dataset.
  - We then get a general overview of the training dataset.
  - This involves:
    - Number of records and attributes.
    - Inspecting the first few and the last few elements.
    - Understanding the general structure of the dataset.

In [ ]:

```
# Loading the training dataset
bike_data <- read.csv("train.csv")

# Loading the testing dataset
bike_data_test <- read.csv("test.csv")
```

Exploring the training dataset : bike_data

In [ ]:

```
# Displaying the dimensions, records and attributes (rows and columns)
cat(" \n Number of records of the dataset:", dim(bike_data)[1])
cat(" \n Number of attributes of the dataset:", dim(bike_data)[2])
```

In [ ]:

```
# Displaying the structure
cat(" \n The structure of the bike dataset is as follows :\n\n")
str(bike_data)
```

summary():

- This displays the distribution of every variable.
- It returns the following for every variable :
  - Minimum Value
  - Maximum Value
  - Mean
  - Median
  - 1st and 3rd Quartile

In [ ]:

```
# Displaying the descriptive statistics of the variables
summary(bike_data)
```

IMPORTANT FINDINGS:

- Looking at our target variable - **Rented.Bike.Count** :
  - We can see that on an average there are around 700 bikes rented every hour.
  - The maximum number of bikes that have been rented are around 3500 bikes every hour.
- Rainfall and Snowfall have the following values as 0:

  - Minimum value
  - Median
  - 1st Quartile
- The following variables have their minimum values as 0:
  - Rented.Bike.Count
  - Hour
  - Humidity
  - Wind.speed

In [ ]:

```
cat("\n First few records of the dataset are:")
# Inspect the first few records
head(bike_data)
```

In [ ]:

```
# And the last few
cat("\n Last few records of the dataset are:")
tail(bike_data)
```

Analysis of Important Findings:

- We have the following variables :
  - Rented.Bike.Count
  - Hour
  - Temperature
  - Humidity
  - Wind.speed
  - Visibility
  - Dew.point.temperature
  - Solar.Radiation
  - Rainfall
  - Snowfall
  - Seasons
  - Holiday
  - Functioning.Day
  - Date
- Clearly, **QUANTITATIVE VARIABLES** are :
  - Hour
  - Temperature
  - Humidity
  - Wind.speed
  - Visibility
  - Dew.point.temperature
  - Solar.Radiation
  - Rainfall
  - Snowfall
- Clearly, **QUANLITATIVE VARIABLES** are :
  - Seasons
  - Holiday
  - Functioning.Day

In [ ]:

```
# Checking for duplicate dates
duplicates <- aggregate(bike_data$Date, list(bike_data$Date), NROW)
duplicates <- bike_data[bike_data$Date %in% duplicates[duplicates$x > 1,"Group.1"],]
head(duplicates[order(duplicates$Date, duplicates$Hour),],20)
```

Analysis:

- Clearly, there is no duplicate data in our dataset.

3. Exploratory Data Analysis

Overview of the Training Dataset

Investigate Distribution of Each Variable : Single Variable Distribution

Plot 1 : Variable distributions using boxplots

- Boxplots are graphical representations of single, numeric data.
- Here, the numerical data is represented through it's quartiles.

In [ ]:

```
attach(bike_data)
```

In [ ]:

```
# Generating the box plots of each variable
m1 <- melt(as.data.frame(bike_data))

# Plotting the boxplot
ggplot(m1, aes(x = variable, y = value)) +
facet_wrap(~variable, scales = "free") +
geom_boxplot() +
theme_minimal() +
scale_y_continuous(labels = function (n) {format(n, scientific = FALSE)})
```

Analysis of IMPORTANT FINDINGS:

- We can clearly see that there are some outliers present in the following variables:
  - Rented.Bike.Count
  - Wind.Speed
  - Solar Radiation
  - Rainfall
  - Snowfall

Plot 2 : Variable distributions using Histograms

- Histograms are made by binning the data and then counting the number of observations in each bin.
- They help in visualising the shape of the distribution.

In [ ]:

```
# Plotting histograms for every numeric variable
# Dividing our area into 3 rows and 3 columns
par(mfrow = c(3,3), bg = "white")

hist(Hour, col = "green", breaks = 10)
grid(col = "gray")
hist(Temperature, col = "orange", breaks = 10)
grid(col = "gray")
hist(Humidity, col = "red", breaks = 10)
grid(col = "gray")

hist(Wind.speed, col = "green", breaks = 10 )
grid(col = "gray")
hist(Visibility, col = "orange", breaks = 10 )
grid(col = "gray")
hist(Dew.point.temperature, col = "red", breaks = 10 )
grid(col = "gray")

hist(Solar.Radiation, col = "green", breaks = 10 )
grid(col = "gray")
hist(Rainfall, col = "orange", breaks = 10 )
grid(col = "gray")
hist(Snowfall, col = "red", breaks = 10 )
grid(col = "gray")
```

Analysis of IMPORTANT FINDINGS:

- **Wind.speed :**
  - The histogram is right skewed.
  - This means that at really high wind speed, there are lesser chances of people travelling outside and using bikes!
- **Visibility :**
  - This histogram is keft skewed.
  - Thus, when the visibility is low, there will be lesser people outside and there are lesser chances of using bikes.
- **Hour :**
  - This histogram has **Uniform Modality**
  - The numbers are approximately the same through all hours of the day.
  - The numbers in the starting hour are slightly more and the numbers towards the end of the day drop down!
- **Rainfall and Snowfall:**
  - Here, we do not observe that points for high values of rainfall and snowfall.
  - Clearly, when it's raining or snowing a bit too much outside, people would prefer staying indoors and hence there would be lesser usage of the rented bike counts!
- **Temperature and Humidity:**
  - These histograms are almost symmetric.
  - This plot follows a Normal Distribution.
  - Thus, only in extreme weather conditions, people are less likely to step out and there would be lesser usage of the bikes.

Investigate Pairs of Variables

Plot 3 : Scatterplot Matrix

- Here, we have plotted the variables using a scatterplot matrix to visualise the correlations between variables.
- Here, this plot represents relationships betweeen two variables at a time.
- They also show the presence of outliers.

In [ ]:

```
pairs(bike_data[sample.int(nrow(bike_data),1000),], panel = panel.smooth)
```

Analysis of IMPORTANT FINDINGS:

- Linear Relationships :
  - Temperature and Humidity.
  - Temperature and Dew.point.temperature

In [ ]:

```
scatterplotMatrix(~Rented.Bike.Count+Hour+Seasons, data = bike_data)
```

Correlation Coefficients

- Here, we display the correlation coefficients for all pairs of variables.
- I have separated out the numeric data for easier analysis.

In [ ]:

```
numeric_data = subset(bike_data, select = c("Rented.Bike.Count","Hour","Temperature","Humic

round(cor(numeric_data), digits = 2)
```

Correlation Matrix

- Here, we visualise the matrix.
- This matrix is useful for summarising our data.
- Here, each cell in our matrix shows correlation between two variables.

In [ ]:

```
# Reference : FIT5149 Week 3 Tutorial

# Defining our panel
myPanel <- function(x, y, z, ...) {
  panel.levelplot(x,y,z,...)
  panel.text(x, y, round(z, 2))
}

# Defining the color scheme
cols = colorRampPalette(c("blue","pink"))

# Plot the correlation matrix.
levelplot(cor(numeric_data), col.regions = cols(100), main = "correlation", xlab = NULL, yl
          scales = list(x = list(rot = 90)), panel = myPanel)
```

Analysis of various values of coefficients:

- **Positive Correlation** :
  - 0 to 0.5 : Weak
  - 0.5 to 0.8 : Moderate
  - 0.8 to 1.0 : Strong
- **Negative Correlation** :
  - -0.5 to 0 : Weak
  - -0.8 to -0.5 : Moderate
  - -1.0 to -0.8 : Strong

Analysis of Important Findings:

- Top **Positive Correlations**:
  - Rented.Bike.Count and Temperature
  - Rented.Bike.Count and Hour
  - Temperature and Dew.point.temperature.
  - Dew.point.temperature and Humidity
- Top **Negative Correlations**:
  - Humidity and Visibility
  - Humidity and Hour
  - Humidity and Solar.Radiation
- Since **Temperature** and **Dew.point.temperature** are correlated,
  - One of them can be removed.
- Variables which are **more correlated** with Rented.Bike.Count :
  - Hour : Weak Positive Correlation
  - Temperature : Moderate Positive Correlation
  - Dew.point.tempearature : Weak Positive Correlation
  - Solar.Radiation : Weak Positive Correlation
  - Visibility : Weak Positive Correlation

- **INFERENCE TASK**:
  - Based on the above analysis, we can say that the attributes that contribute the most to our model's performance:
    - Hour
    - Temperature
    - Solar.Radiation

    - Visibility
  - Since correlations between the predictor values is not desirable, we can remove one of the variables :
    - Example : We can remove Dew.point.temperature as it is highly correlated with Temperature.

4. Methodology

- Here, I have developed two models to accurately predict the number of bikes required.
  - Model 1 : Linear Regression Model
  - Model 2 : SVM : Support Vector Machine Model
- **Methodology Followed for the Development of the Linear Model** :
  - First I started with the most basic approach required to tackle the prediction. Hence, I started off with a linear model.
  - I took into consideration all the variables while predicting the target variable.
  - Then, I compared the accuracy of the models by checking the Adjusted R-squared values and p values.
  - I then sorted out the variables which had greater effects on our target variable.
  - I even plotted and analysed certain diagnostic plots for each model.
  - Thus, this linear model was further improved by removing certain predictor variables and showing various interactions between the variables.
  - The different interactions used were derived using the results of the correlation matrix and hit and trial methods!
    - Example :
      - Hour:Temperature:Visibility:Seasons:Functioning.Day
      - Relation between Hour:Temperature
        - How the usage of the rented bikes changes at different hoirs of the day according to temperature.
      - Relation between Temperature and Visibility
        - How to Visibility changes at various temperatures, and how this in turn affects the use of the rented bikes.
      - Relation between Visibility, Seasons and Functioning.Day
        - How the various Seasons decide whether a given day is functional or nor, based on the visibility - and how this in turn affects tge usage of the rental bikes.
  - Some variables did not have an uniform distribution. In order to accomodate these predictions.
    - I have used sqrt() function on the target variable.
  - This model was then used to train the dataset and predict the values and the R Squared Value comes to around 0.73
  - Furthermore, I have even plotted the actual and predicted values using the Linear Regression Model to show the accuracy and variance!
- **Methodology Followed for the Development of the SVM Model** :
  - The SVM model has been used in order to improve the accuracy of the findings.
  - Here, I have considered all the variables used for predicting the target variable.
  - After training the dataset and testing the dataset, the R Squared Value comes to around 0.77

5. Model Development

- This section revolves around how the two models were developed.
- A step by step procedure has also been provided.

Model 1 : Linear Regression Model:

Model 2 : SVM : Support Vector Machine Model:

Model 1 : Building the Linear Regression Model

- Here, this is the basic model.
- We have considered all variables while determining the target variable.

Preparing the Data Frame

- Here, we re-load the data frame and factorise the categorical variables.
- Categorical variables can take values which we cannot organise in a logical sequence. Our categorical variables are :
  - Seasons
  - Holiday
  - Functioning.Day

In [ ]:

```
bike_data <- read.csv("train.csv")
#head(bike_data)

# 4 Seasons : Summer, Autumn, Spring, Winter
bike_data$Seasons <- as.factor(bike_data$Seasons)

# 2 : Holiday, No Holiday
bike_data$Holiday <- as.factor(bike_data$Holiday)

# 2 : Yes, No
bike_data$Functioning.Day <- as.factor(bike_data$Functioning.Day)

# Viewing the structure of our training dataset
str(bike_data)
```

Functions for Model Accuracy

- We will use these functions while building our model.
- We will use them in order to evaluate the accuracy of the model.

Function to Calculate RMSE:

- **Name** : Calculate_RMSE
- **Input parameters** :
  - true : vector of true values
  - predicted : vector of predicted values
- **Return values** :
  - A data frame containing the RMSE Value
- **Description** :
  - Calculate the TSS and RSS as:
  - RSS: $\sum_{i=1}^{n} (\hat{y}_i - y_i)^2$
  - Residual standard error - $\sqrt{\frac{1}{df} RSS}$

In [ ]:

```
# Function to calculate the RMSE
calculate_RMSE <- function(true, predicted) {
    RSS <- sum((predicted - true)^2)

    # Computing the RMSE value
    RMSE = sqrt(RSS/length(predicted))
    RMSE <- round(RMSE, digits = 2)

    # Returning the two values
    data.frame(
        RMSE = RMSE
    )
}
```

Function to Calculate R Squared Value:

- **Name** : Calculate_R_Squared
- **Input parameters** :
  - true : vector of true values
  - predicted : vector of predicted values
- **Return values** :
  - A data frame containing the R Squared Values
- **Description** :
  - Calculate the TSS and RSS as:
  - RSS: $\sum_{i=1}^{n}(\hat{y}_i - y_i)^2$
  - TSS: $\sum_{i=1}^{n}(y_i - \bar{y})^2$
  - R-Squared value: $R^2 = 1 - \frac{RSS}{TSS}$

In [ ]:

```
# Function to calculate the R-Squared value
calculate_R_Squared <- function(true, predicted) {
    RSS <- sum((predicted - true)^2)
    TSS <- sum((true - mean(true))^2)

    # Computing the R Squared value
    R_Squared <- 1 - (RSS / TSS)
    R_Squared <- round(R_Squared, digits = 2)

    # Returning the two values
    data.frame(
        Rsquare = R_Squared
    )
}
```

Model 1 : Fitting in all variables

- Here, we try fitting all variables to see what appears to be important.
- We see how the different variables are affecting the Rented.Bike.Count
- Here, we build the regression model with the lm() function.

In [ ]:

```
fit1 <- lm(Rented.Bike.Count ~ ., data = bike_data)
summary(fit1)
```

Analysis of Meaningful Interpretations:

- Our **output** has the following:
  - residuals
  - coefficients
  - residual standard error
  - F-statistic
- These are useful to assess and test the accuracy of our generated model.
- **Adjusted R-squared (R^2)** :
  - This value indicates that this particular model explains 66% of the variation in **Rented Bike Counts.**
- **F-statistic**:
  - This is really useful in determining whether there is a relationship between our predictor variable and response variables.
  - In general, the further the F-statistic is from 1, the better!
  - This has a p-value < 2.2e-16 - Thus, we reject the null hypothesis.
  - Null Hypothesis : the model explains nothing.
  - Hence this implies, the model is useful.
- **p-values:**
  - From the p values, we can see that :
    - The relationship between Rented.Bike.Count and Dew.point.temperature
    - It is **NOT THAT SIGNIFICANT**

In [ ]:

```
# Removing Dew.point.temperature

fit2 <- lm(Rented.Bike.Count ~ ., data = subset(bike_data, select=c(-Dew.point.temperature)
summary(fit2)
```

Interpretations:

- Here, The Adjusted R-squared for the full model is 0.66.
- The Adjusted R-squared for the second model is 0.66.
- The two values are the same.
- Thus, excluding the variable :
  - It has made the model simple.
  - Without significantly losing the modeling accuracy.

Plot Function: plot()

- The plot function produces four diagnostic plots as shown below :

In [ ]:

```
par(mfrow = c(2,2))
plot(fit1)
```

Interpretations:

The diagnostic plots show residuals in four different ways.

1. **Residual vs fitted plot**:
    - This plot is used to check if residuals have linear or non-linear patterns.
    - Linear Relationships : If there are equally spread residuals around a horizontal line without distinct patterns.
    - Non Linear Relationships: If the relationship between predictors and an response variable is non-linear, an obvious pattern could show up in this plot.
    - Here : There could be a non-linear relationship between Rented.Bike.Count and all the predictors, as the residuals are not scattered evenly.
2. **Normal Q-Q (quantile-quantile plot) plot**:
    - The normal Q-Q plot tells us if residuals are normally distributed.
    - If the residuals are properly alined on the dashed straight line, it is a good sign!
    - Here : The residuals seem to be normally distributed.
3. **Scale-location plot**:
    - This plot is used to check the assumption of equal variance.
    - This is done by displaying if the residuals are spread equally along the ranges of predictors.
    - Here : The residuals seem to be equally randomly spread around the horizontal line.
4. **Residual-leverage plot**:
    - This plot helps us identify influential data samples.
    - In the residual-leverage plot, we look for outlying values at the upper right corner or at the lower right corner.
    - Samples located in those places can be influential against a regression line.
    - Here : We have outliers such as 5741.
    - There are possible influential outliers.

In [ ]:

```
summary(update(fit1, . ~ . + Temperature:Dew.point.temperature))
```

Interpretations:

- Here, this has slightly increased the value of Adjusted R-Squared to 0.6682
- From the p-values,
    - The following variables have really low p value
    - This means that we can reject the Null Hypothesis
    - we can see that the following variables have significant contribution to our target :
        - Hour
        - Temperature
        - Visibility
        - Dew.point.temperature
        - Solar.Radiation

    - Snowfall
    - Rainfall
    - Functioning.Day

Update : Fitting a smaller model

- This model only uses predictors for which there is some evidence associated with the outcome.

In [ ]:

```
fit3 = lm(Rented.Bike.Count ~ Hour + Temperature + Visibility + Solar.Radiation + Snowfall
summary(fit3)
```

Comparing fit 1 and fit 2 and fit 3:

- Here, the Adjusted R-Squared value has decreased.
- There is not much difference between the two.
- Hence, I have gone ahead with the basic simple model.
- Also, a model with lesser number of predictors is preferable.
- Thus, we **can remove the predictor : Dew.point.temperature!**

In [ ]:

```
# Comparing the diagnostic plots
par(mfcol=c(2,2))

plot(fit1, which = 1)
plot(fit2, which = 1)

plot(fit1, which = 2)
plot(fit2, which = 2)
```

- Thus, we can see that removing the Dew.point.temperature does not make that much of a difference!

Update : Step() function

- Here, we select the best variables by using the step function!
- Here, I am performing the step functions on the fit1 model - where I had considered all my variables.

In [ ]:

```
step1 <- step(fit1)
summary(step1)
```

Interpretation:

- Thus, the best model has the following predictors:
    - Functioning.Day
    - Wind.speed
    - Snowfall

- Rainfall
- Visibility
- Humidity
- Solar.Radiation
- Temperature
- Hour
- Date

Analysing the interaction between the various models

Using anova() function in order to compare the two models:

- Here, all terms of the smaller model appear in the larger model.
- In the following steps, I have analysed the interactions between the different variables in order to improve the accuracy of our model by improving the R Squared Value.

Interaction 1 : Temperature and Dew.point.temperature

- Here, I have considered this because of the high level of correlation between the two from the correlation matrix.
- We can see that the R value has slightly increased.

In [ ]:

```
fit4 = update(step1, . ~ . + Temperature:Dew.point.temperature)
summary(fit4)
#0.6675
```

Interaction 2 : Hour:Temperature

- Here, we see how the temperature changes over different hours affecs the Rented Bike Count.
- This has also slightly increased our R value.

In [ ]:

```
fit6 = update(step1, . ~ . + Hour:Temperature)
summary(fit6)
#0.6962
```

Interaction 3 : Hour - Temperature and Seasons

- We further see how this change in temperature occurs in various seasons.
- This also increases the Adjusted R Square Value.

In [ ]:

```
fit9 = update(step1, . ~ . + Hour:Temperature:Seasons)
summary(fit9)
#0.7021
```

In [ ]:

```
fit10 = update(step1, . ~ . + Hour:Temperature:Seasons:Functioning.Day)
summary(fit10)
#0.7135
```

In [ ]:

```
fit11 = update(step1, . ~ . + Hour:Temperature:Visibility:Seasons:Functioning.Day)
summary(fit11)
#0.7151
```

In [ ]:

```
fit12 = update(step1, . ~ . + Hour:Temperature:Visibility:Seasons:Functioning.Day:Holiday)
summary(fit12)
#0.7157
anova(fit1, fit12)
```

Interactions Summary:

- Thus, with every variable interaction, we can see that the Adjusted Square Value has increased!
- Thus, our model accuracy has gradually increased.

Using the Model to Predict Prices

In [ ]:

```
# Splitting features and target variables

train_target <- bike_data[,c("Rented.Bike.Count")]
train_features <- bike_data[,c("Hour","Temperature","Humidity","Wind.speed","Visibility","D

test_target <- bike_data_test[,c("Rented.Bike.Count")]
test_features <- bike_data_test[,c("Hour","Temperature","Humidity","Wind.speed","Visibility
```

Predict the Rented Bike Count for the Testing Dataset

Linear Model 1 :

- Here, all variables have been used to predict the target variable.
- The target variable is kept as it is while developing the model.

In [ ]:

```
# Developing our Linear Model
model_1<- lm(Rented.Bike.Count~. +
                   Hour:Temperature:Visibility:Seasons:Functioning.Day:Holiday, data = bike
summary(model_1)
#R-Squared : 0.716
```

In [ ]:

```
# Visualising our linear model 1
par(mfcol = c(2,2))
plot(model_1)
```

In [ ]:

```
# Predictions with model 1 for the training dataset
model_1_train_prediction <- predict(model_1, newdata = bike_data)

cat(" \n Model Accuracy for training dataset:")
# Evaluating the Model Accuracy for training dataset
calculate_RMSE(test_target, model_1_train_prediction)
calculate_R_Squared(test_target, model_1_train_prediction)
```

In [ ]:

```
# Predictions with model 1 for the testing dataset
model_1_test_prediction <- predict(model_1, newdata = bike_data_test)

cat(" \n Model Accuracy for testing dataset:")
# Evaluating the Model Accuracy for testing dataset
calculate_RMSE(test_target, model_1_test_prediction)
calculate_R_Squared(test_target, model_1_test_prediction)
```

Analysing the Accuracy of the Model :

- Here, the high value of RSquared indicates high accuracy of the model!
- We started with an RSquared Value of 0.66 and now the value of 0.73

Linear Model 2:

- Here, certain interactions between the predictor variables have been taken into consideration.
- Also, we have considered the sqrt() of the target variable while developing the model.
- Usage of sqrt()
  - In order to solve the problem of heteroskedasticity.

In [ ]:

```
# Updating our Linear Model
model_2 <- lm(sqrt(Rented.Bike.Count)~. +
                   Hour:Temperature:Visibility:Seasons:Functioning.Day:Holiday, data = bike
summary(model_2)
#R-Squared : 0.7704
```

In [ ]:

```
# Visualising our Linear Model 2
par(mfcol = c(2,2))
plot(model_2)
```

In [ ]:

```
# Predictions with model 2 for the training dataset
model_2_train_prediction <- predict(model_2, newdata = bike_data)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, model_2_train_prediction^2)
calculate_R_Squared(test_target, model_2_train_prediction^2)
```

In [ ]:

```
# Predictions with model 2 for the testing dataset
model_2_test_prediction <- predict(model_2, newdata = bike_data_test)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, model_2_test_prediction^2)
calculate_R_Squared(test_target, model_2_test_prediction^2)
```

Linear Model 3 : Adding some more interactions

In [ ]:

```
# Updating our Linear Model by adding some more variable interactions
model_3 <- lm(sqrt(Rented.Bike.Count)~. +
                   Hour:wday(Date, label = TRUE): Humidity +
                   Hour:Solar.Radiation:Temperature:month(Date, label = TRUE) +
                   Hour:Dew.point.temperature:Solar.Radiation +
                   sqrt(Visibility):month(Date, label = TRUE):Temperature - Visibility +
                   Humidity:Temperature +
                   Humidity:Rainfall +
                   Humidity:Solar.Radiation +
                   Holiday:wday(Date, label = TRUE):Temperature +
                   Seasons:Solar.Radiation +
                   Date:Seasons +
                   Functioning.Day:wday(Date, label = TRUE), data = bike_data)
summary(model_3)
#R-Squared : 0.7831
```

In [ ]:

```
# Predictions with model 2 for the training dataset
model_3_train_prediction <- predict(model_3, newdata = bike_data)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, model_3_train_prediction^2)
calculate_R_Squared(test_target, model_3_train_prediction^2)
```

In [ ]:

```
# Predictions with model 2 for the testing dataset
model_3_test_prediction <- predict(model_3, newdata = bike_data_test)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, model_3_test_prediction^2)
calculate_R_Squared(test_target, model_3_test_prediction^2)
```

Analysing the Accuracy of the Model :

- Here, the high value of RSquared indicates high accuracy of the model!
- We started with an RSquared Value of 0.66 and now the value of 0.75!

Model 2 : SVM Model

- These are Support-Vector Machines Models.
- They are supervised learning models.
- They can efficiently perform linear and non linear classification.
- This model clusters data into groups and then mapping of this data to tho new groups which have been formed.
- **SVM Model 1 :**
  - Here, we are first considering all our variables for predictions.

In [ ]:

```
# Developing our SVM Model 1
# Here, our type would be that of eps-regression
svm_model <- svm(Rented.Bike.Count~Hour + Temperature + Humidity + Wind.speed + Visibility
#summary(svm_model)
```

In [ ]:

```
# Predictions with SVM model 1 for the training dataset
svm_model_train_prediction <- predict(svm_model, newdata = bike_data)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, svm_model_train_prediction)
calculate_R_Squared(test_target, svm_model_train_prediction)
```

In [ ]:

```
# Predictions with SVM model 1 for the testing dataset
svm_model_test_prediction <- predict(svm_model, newdata = bike_data_test)

# Evaluating the Model Accuracy
calculate_RMSE(test_target, svm_model_test_prediction)
calculate_R_Squared(test_target, svm_model_test_prediction)
```

Analysis:

- We can clearly see that our R Squared Value has significantly increased to 0.77!

6. Results and discussion

- Here, we will be **COMPARING** the two models.

Plot 1 : Plotting the actual and predicted values using the SVM Model

In [ ]:

```
set_plot_dimensions <- function(width_choice, height_choice) {
        options(repr.plot.width = width_choice, repr.plot.height=height_choice)
        }

# For the SVM Model : Plotting the actual values vs the predicted vales
par(mfcol = c(1,2))

x_axis <- c(1:length(svm_model_test_prediction))

set_plot_dimensions(10, 6)
plot(x_axis, test_target, col = "green", width = 5, height = 10)
title('Actual Test Data')

set_plot_dimensions(10, 6)
plot(x_axis, svm_model_test_prediction, col = "blue")
title('Predicted Data')
```

Plot 2 : Plotting the actual and predicted values using the Linear Regression Model

- Here, I have considered the final linear model developed.

In [ ]:

```
set_plot_dimensions <- function(width_choice, height_choice) {
        options(repr.plot.width=width_choice, repr.plot.height=height_choice)
        }

# For the SVM Model : Plotting the actual values vs the predicted vales
par(mfcol = c(1,2))

x_axis <- c(1:length(model_3_test_prediction))

set_plot_dimensions(10, 6)
plot(x_axis, test_target, col = "green", width = 5, height = 10)
title('Actual Test Data')

set_plot_dimensions(10, 6)
plot(x_axis, model_3_test_prediction, col = "blue")
title('Predicted Data')
```

Analysis from the two plots:

- Thus, we can see that the predictions made by the SVM are more accurate than the predictions made by the linear model.
- **Correlation between the predictor variables:**
  - In general, greater correlation between the target and predictor variables is a good sign and correlation between the different predictor variables is a bad sign.
  - Here, from the correlation matrix, we can see that there is a significant amount of correlation between the variables.
  - This affects the accuracy while predicting the Rented.Bike.Count.
- **Non-Linear Relationships between the target and predictor variables:**
  - In order to accomodate these relationships, we have used sqrt() transformations and have used an SVM model.
- **Advantages of the Linear Regression Model:**

- Simple approach
- Easy to read, understand and interpret
- **Dis-advantages of the Linear Regression Model:**
  - It assumes a linear relationship between the variables
  - This does not take into consideration the non linear aspects
  - Over-simplifies the problem
- **Advantages of the SVM Model:**
  - It takes into consideration the Non-Linear aspects
  - This is comparatively memory efficient

7. Conclusion

- Thus, we have successfully performed significant data exploration, model development and testing.
- **In Data Exploration and Exploratory Data Analysis:**
  - We have removed certain variables like Dew.point.temperature from the analysis, because of the high correlation found with Temperature.
  - We have even analysed that certain variables like Hour, Temperature, Solar.Radiation contribute a lot more than the other variables while predicting the Rented.Bike.Count
- **In Model Development and Testing:**
  - We have been able to understand the steps taken to develop a linear regression and a SVM Model.
  - We have handled and accomodated the Non-Linear relationships as well.
  - We have successfully trained our datasets in order to make right predictions.
  - Finally, we have been able to able to measure the accuracy of our models developed!

8. References

- https://en.wikipedia.org/wiki/Support_vector_machine (https://en.wikipedia.org/wiki/Support_vector_machine)
- https://www.geeksforgeeks.org/ml-advantages-and-disadvantages-of-linear-regression/ (https://www.geeksforgeeks.org/ml-advantages-and-disadvantages-of-linear-regression/)
- https://statistics.laerd.com/statistical-guides/understanding-histograms.php (https://statistics.laerd.com/statistical-guides/understanding-histograms.php)
- https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51 (https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51)
- https://www.displayr.com/what-is-a-correlation-matrix/ (https://www.displayr.com/what-is-a-correlation-matrix/)
- https://statisticsbyjim.com/regression/interpret-r-squared-regression/ (https://statisticsbyjim.com/regression/interpret-r-squared-regression/)