Bike Renting

Pesala Ravi Kumar

14/08/2019

Contents

1. Introduction 2

1.1 Problem Statement 2

1.2 Data 2

2. Methodology 3

2.1 Pre Processing 3

*2.1.1 Outlier Analysis* 3

*2.1.2 Missing Value Analysis* 4

*2.1.3 Feature Selection* 4

2.2 Modeling 5

*2.2.1 Model Selection*  5

*2.2.2 Principal Component Analysis (PCA)* 7

*2.2.3 Linear Regression* 7

*2.2.3.1 Simple Linear Regression* 7

*2.2.3.2 Multiple Linear Regression* 7

*2.2.4 Polynomial Regression* 7

*2.2.5 K-Means Clustering* 7

3. Conclusion & Future Work 10

3.1 Model Evaluation 10

*3.1.1 MAE & MAPE* 10

*3.1.2 MSE & RMSE* 10

*3.1.3 R Squared & Adjusted R Squared* 10

Appendix 11

R code 11

Python code 17

Visualizations 22

**Chapter 1**

**Introduction**

**1.1 Problem Statement**

The objective of this Case is to Predication of bike rental count on daily based on the environmental and seasonal settings. Now a days there are many organizations who are running these bike rental work. In order to reduce the effort of renting the bike in different time periods and conditions by applying some machine learning concepts. We would like to predict the count of bike rental based on the conditions when the bike is renting by customers which are already known and easy to calculate further predictions.

**1.2 Data**

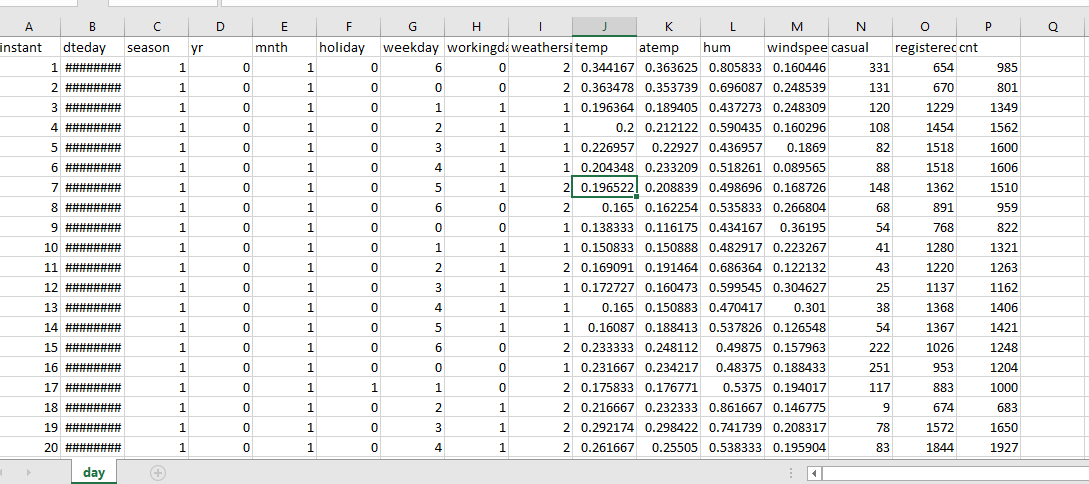


Fig 1.2.1 Data

As you can see in the fig.1.2.1 above we have the following 15 variables using which we have to predict the one of the variables.

1. Date
2. Season
3. Year
4. Month
5. Holiday
6. Weekday
7. Working Day
8. Weather
9. temp
10. atemp
11. Humidity
12. Casual
13. Registered
14. Count

**Chapter 2**

**Methodology**

**2.1 Pre Processing**

Any predictive modeling requires that we look at the data before we start modeling. 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 as Exploratory Data Analysis.

**2.1.1 Outlier Analysis**

Outlier, it is an observation which inconsistence to rest of data.

Causes of the outlier are,

* 1. Poor data quality or contamination.
  2. Low quality measurements.
  3. Manual error.
  4. Malfunctioning equipment.
  5. Correct but exceptional data

Effect of outliers is that it will gives data which is not present in data because of the outlier.

Assume data as 1, 3, 5, 7, and 14 now try to impute using mean value. Mean will be 6 which is not present under data so it might reflect the modelling.

Steps to detect an outlier are,

1. Detect variable with outlier using graphical tools
2. Replace all with NA
3. Apply the missing value analysis on NA records to impute new Values.

**2.1.2 Missing Value Analysis**

In statistics, missing data, or missing values, occur when no data value is stored for the variable in an observation. Missing data are a common occurrence and can have a significant effect on the conclusions that can be drawn from the data.

Why missing the value?

Human Error, refuse to answer, optional

In order to handle Missing Values, there are 2 cases Ignore or Impute Missing value.

Before Imputing Understand why value is missing and by plotting graphs.

Delete the observations where you not to impute.

Techniques to impute missing values:

1. Fill with central statistics like
   * 1. Mean
     2. Mode
     3. Median
2. Distance base/ Data Mining Method – KNN Imputation

While imputing using KNN method there may be a chance of getting an error: Not sufficient complete cases for computing neighbors.

In that case we can go for Central Statistics to impute missing values.

1. Prediction method – Machine Learning models.

Selecting the technique for missing values

1. Create a small subset of total data.
2. Delete some values manually.
3. Use multiple methods to fill.
4. See which technique fills correctly.
5. Select that technique for finding missing value analysis.

**2.1.3 Feature Selection**

It is also called as variable selection or attribute selection.

Selecting a subset of relevant features like variables, predictors for model construction.

Advantages of Feature selection is Dimensionality Reduction (Variable reduction).

Techniques to dimensionality reduction,

1. Correlation analysis.
2. Chi-square test of independence.

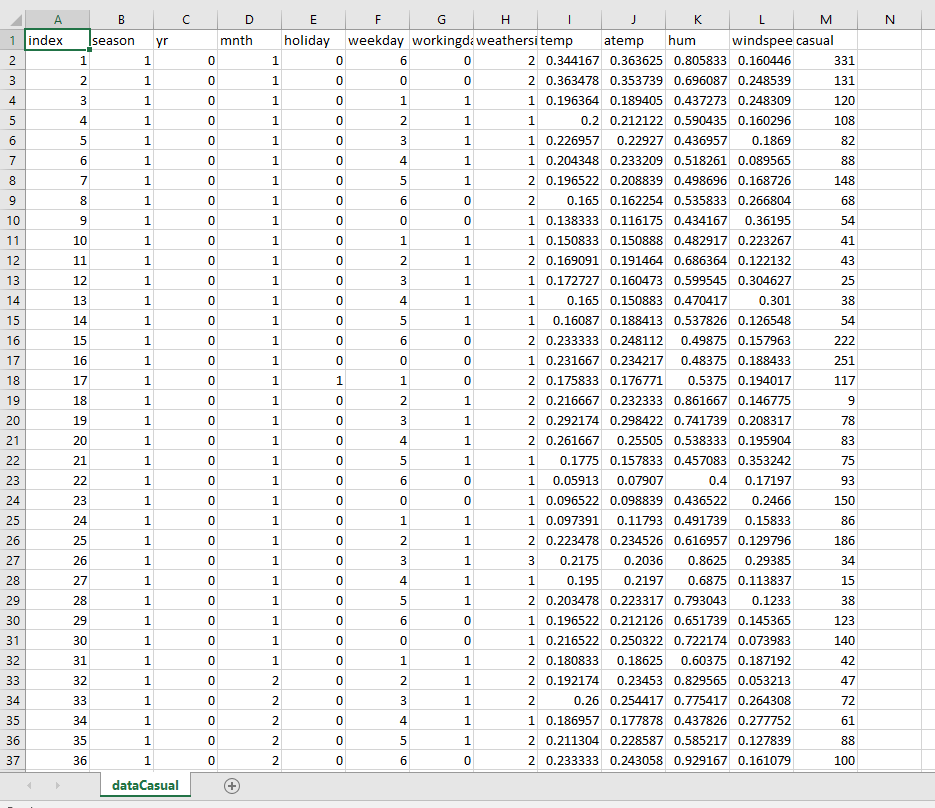
**2.2 Modeling**

**2.2.1 Model Selection**

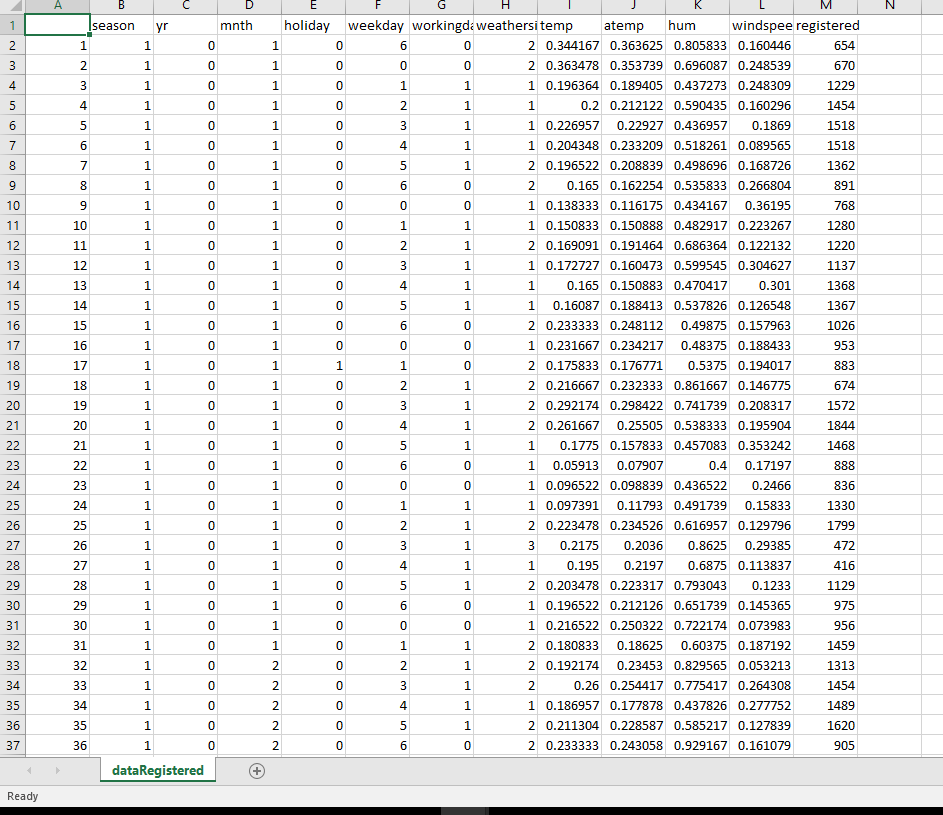
In our early stages of analysis during preprocessing we have come to understand that in order to predict the count of bike rentals we need to predict the casual and registered bikes so that we can add those two and calculate the total count of bike rentals.

In order to predict total count we’ll divide the data into 2 data sets with casual and registered variables separately.

Casual Dataset



Registered Dataset



You always start your model building from the simplest to more complex so after applying all the models select the best model according to accuracy and MAPE and etc.

**2.2.2 Principal Component Analysis**

**Principal Component Analysis (PCA)**is a statistical procedure that uses an orthogonal transformation which converts a set of correlated variables to a set of uncorrelated variables. PCA is a most widely used tool in exploratory data analysis and in machine learning for predictive models. Moreover, PCA is an unsupervised statistical technique used to examine the interrelations among a set of variables. It is also known as a general factor analysis where regression determines a line of best fit.

Each of the principal components is chosen in such a way so that it would describe most of the still available variance and all these principal components are orthogonal to each other. In all principal components first principal component has maximum variance.

**Uses of PCA:**

* It is used to find inter-relation between variables in the data.
* It is used to interpret and visualize data.
* As number of variables are decreasing it makes further analysis simpler.
* It’s often used to visualize genetic distance and relatedness between populations.

**Objectives of PCA:**

* It is basically a non-dependent procedure in which it reduces attribute space from a large number of variables to a smaller number of factors.
* PCA is basically a dimension reduction process but there is no guarantee that the dimension is interpretable.
* Main task in this PCA is to select a subset of variables from a larger set, based on which original variables have the highest correlation with the principal amount.

**Principal Axis Method:** PCA basically search a linear combination of variables so that we can extract maximum variance from the variables. Once this process completes it removes it and search for another linear combination which gives an explanation about the maximum proportion of remaining variance which basically leads to orthogonal factors. In this method, we analyze total variance.

**Eigenvector:** It is a non-zero vector that stays parallel after matrix multiplication. Let’s suppose x is eigen vector of dimension r of matrix M with dimension r\*r if Mx and x are parallel. Then we need to solve Mx=Ax where both x and A are unknown to get eigen vector and eigen values.  
Under Eigen-Vectors we can say that Principal components show both common and unique variance of the variable. Basically, it is variance focused approach seeking to reproduce total variance and correlation with all components. The principal components are basically the linear combinations of the original variables weighted by their contribution to explain the variance in a particular orthogonal dimension.

**Eigen Values:** It is basically known as characteristic roots. It basically measures the variance in all variables which is accounted for by that factor. The ratio of eigenvalues is the ratio of explanatory importance of the factors with respect to the variables. If the factor is low then it is contributing less in explanation of variables. In simple words, it measures the amount of variance in the total given database accounted by the factor. We can calculate the factor’s eigen value as the sum of its squared factor loading for all the variables.

**2.2.3 Linear Regression**

**Linear Regression** is a machine learning algorithm based on **supervised learning**. It performs a **regression task**. Regression models a target prediction value based on independent variables. It is mostly used for finding out the relationship between variables and forecasting. Different regression models differ based on – the kind of relationship between dependent and independent variables, they are considering and the number of independent variables being used.  


Linear regression performs the task to predict a dependent variable value (y) based on a given independent variable (x). So, this regression technique finds out a linear relationship between x (input) and y(output). Hence, the name is Linear Regression.  
In the figure above, X (input) is the work experience and Y (output) is the salary of a person. The regression line is the best fit line for our model.

**Hypothesis function for Linear Regression :**  

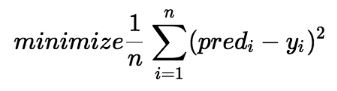

While training the model we are given :  
**x:** input training data (univariate – one input variable(parameter))  
**y:** labels to data (supervised learning)

When training the model – it fits the best line to predict the value of y for a given value of x. The model gets the best regression fit line by finding the best θ1 and θ2 values.  
**θ1:** intercept  
**θ2:** coefficient of x

Once we find the best θ1 and θ2 values, we get the best fit line. So when we are finally using our model for prediction, it will predict the value of y for the input value of x.

**How to update θ1 and θ2 values to get the best fit line ?**

**Cost Function (J):**  
By achieving the best-fit regression line, the model aims to predict y value such that the error difference between predicted value and true value is minimum. So, it is very important to update the θ1 and θ2 values, to reach the best value that minimize the error between predicted y value (pred) and true y value (y).





Cost function(J) of Linear Regression is the **Root Mean Squared Error (RMSE)** between predicted y value (pred) and true y value (y).

**Gradient Descent:**   
To update θ1 and θ2 values in order to reduce Cost function (minimizing RMSE value) and achieving the best fit line the model uses Gradient Descent. The idea is to start with random θ1 and θ2 values and then iteratively updating the values, reaching minimum cost.

**2.2.3.1 Simple Linear Regression:**

Simple linear regression is an approach for predicting a **response** using a **single feature**.

It is assumed that the two variables are linearly related. Hence, we try to find a linear function that predicts the response value(y) as accurately as possible as a function of the feature or independent variable(x).

For generality, we define:

x as **feature vector**, i.e x = [x\_1, x\_2, …., x\_n],

y as **response vector**, i.e y = [y\_1, y\_2, …., y\_n]

Now, the task is to find a **line which fits best** in above scatter plot so that we can predict the response for any new feature values. (i.e a value of x not present in dataset)

This line is called **regression line**.

The equation of regression line is represented as:

h(x\_i) = b\_0 + (b\_1)(x\_i)

Here,

* h(x\_i) represents the **predicted response value** for i'th observation.
* b\_0 and b\_1 are regression coefficients and represent **y-intercept** and **slope** of regression line respectively.

**2.2.3.2 Multiple Linear Regression:**

Multiple linear regression attempts to model the relationship between **two or more features** and a response by fitting a linear equation to observed data.

Clearly, it is nothing but an extension of Simple linear regression.

Consider a dataset with **p** features(or independent variables) and one response(or dependent variable).  
Also, the dataset contains **n** rows/observations.

**Implementation:**

According to data we can specify the linear regression type as Multiple Linear Regression because of the multiple independent variables.

vif(saved\_dataCasual[,-12])

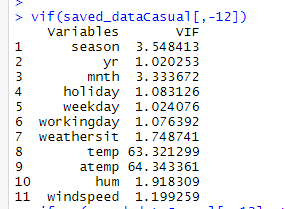


Fig 2.2.2.1 vif(Casualdata)

vifcor(saved\_dataCasual[,-12], th = 1.0)

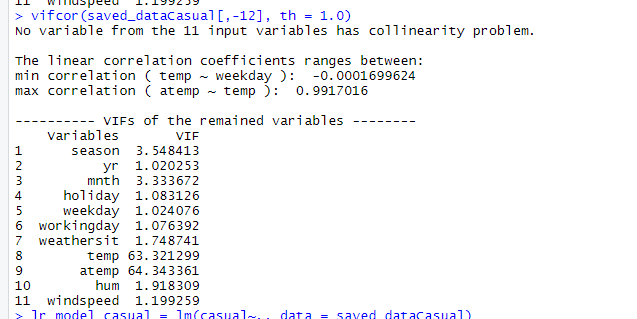


Fig 2.2.2.2 vifcor(CasualData)

lr\_model\_casual = lm(casual~., data = saved\_dataCasual)

summary(lr\_model\_casual)

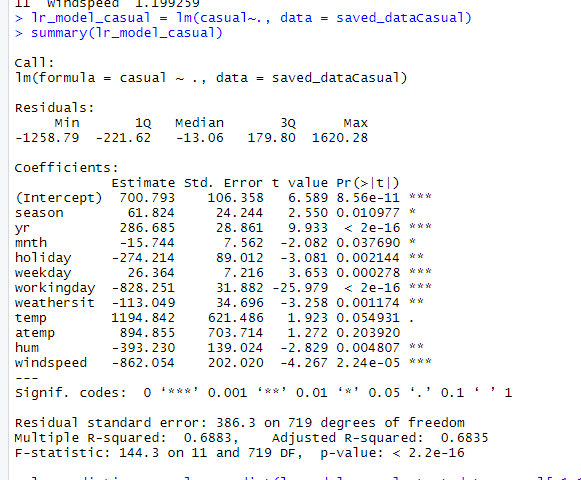


Fig 2.2.2.3 summary(linearRegressionModel)

vif(saved\_dataRegistered[,-12])

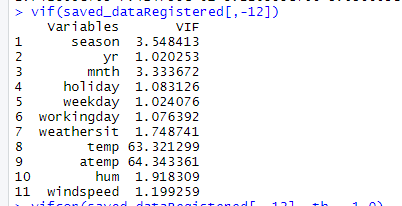


Fig 2.2.2.4 vif(RegisteredData)

vifcor(saved\_dataRegistered[,-12], th = 1.0)

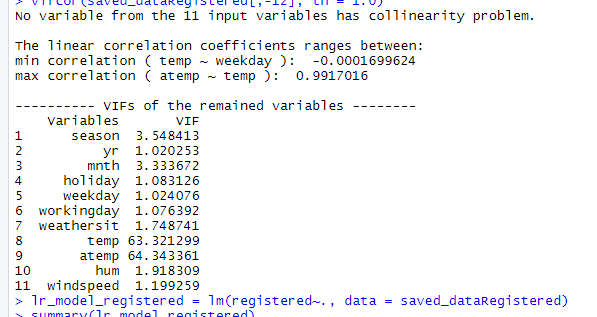


Fig 2.2.2.5 vifcor(RegisteredData)

lr\_model\_registered = lm(registered~., data = saved\_dataRegistered)

summary(lr\_model\_registered)

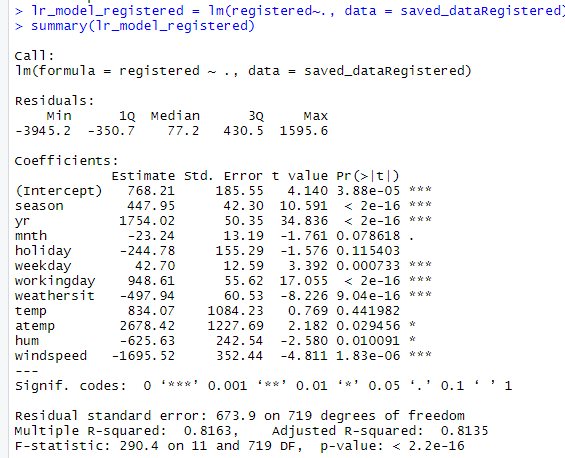


Fig 2.2.2.6 summary(linearRegressionModel)

**2.2.4 Polynomial Regression**

**Polynomial Regression**is a form of linear regression in which the relationship between the independent variable x and dependent variable y is modeled as an nth degree polynomial. Polynomial regression fits a nonlinear relationship between the value of x and the corresponding conditional mean of y, denoted E(y |x).

**Why Polynomial Regression:**

* There are some relationships that a researcher will hypothesize is curvilinear. Clearly, such type of cases will include a polynomial term.
* Inspection of residuals. If we try to fit a linear model to curved data, a scatter plot of residuals (Y axis) on the predictor (X axis) will have patches of many positive residuals in the middle. Hence in such situation it is not appropriate.
* An assumption in usual multiple linear regression analysis is that all the independent variables are independent. In polynomial regression model, this assumption is not satisfied.

**Uses of Polynomial Regression:**   
These are basically used to define or describe non-linear phenomenon such as:

* Growth rate of tissues.
* Progression of disease epidemics
* Distribution of carbon isotopes in lake sediments

The basic goal of regression analysis is to model the expected value of a dependent variable y in terms of the value of an independent variable x. In simple regression, we used following equation –

y = a + b.x + e

Here y is dependent variable, a is y intercept, b is the slope and e is the error rate.

In many cases, this linear model will not work out .For example if we analyzing the production of chemical synthesis in terms of temperature at which the synthesis take place in such cases we use quadratic model

y = a + b1.x + b2^2 + e

Here y is dependent variable on x, a is y intercept and e is the error rate.

In general, we can model it for nth value.

y = a + b1.x + b2.x^2 + …. +bn.x^n

Since regression function is linear in terms of unknown variables, hence these models are linear from the point of estimation.

**Advantages of using Polynomial Regression:**

* Broad range of function can be fit under it.
* Polynomial basically fits wide range of curvature.
* Polynomial provides the best approximation of the relationship between dependent and independent variable.

**Disadvantages of using Polynomial Regression**

* These are too sensitive to the outliers.
* The presence of one or two outliers in the data can seriously affect the results of a nonlinear analysis.
* In addition there are unfortunately fewer model validation tools for the detection of outliers in nonlinear regression than there are for linear regression.

**2.2.5 K-Means Clustering**

We are given a data set of items, with certain features, and values for these features (like a vector). The task is to categorize those items into groups. To achieve this, we will use the kMeans algorithm; an unsupervised learning algorithm.

**Overview**

(It will help if you think of items as points in an n-dimensional space).  The algorithm will categorize the items into k groups of similarity. To calculate that similarity, we will use the Euclidean distance as measurement.

The algorithm works as follows:

1. First we initialize k points, called means, randomly.
2. We categorize each item to its closest mean and we update the mean’s coordinates, which are the averages of the items categorized in that mean so far.
3. We repeat the process for a given number of iterations and at the end, we have our clusters.

The “points” mentioned above are called means, because they hold the mean values of the items categorized in it. To initialize these means, we have a lot of options. An intuitive method is to initialize the means at random items in the data set. Another method is to initialize the means at random values between the boundaries of the data set (if for a feature *x* the items have values in [0,3], we will initialize the means with values for *x* at [0,3]).

The above algorithm in pseudocode:

Initialize k means with random values

For a given number of iterations:

Iterate through items:

Find the mean closest to the item

Assign item to mean

Update mean

**Chapter 3**

**Conclusion**

**3.1 Model Evaluation**

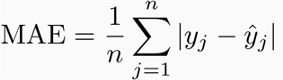
Now that we have a few models for predicting the target variable, we need to decide which one to choose. There are several criteria that exist for evaluating and comparing models. We can compare the models using any of the following criteria:

1. Predictive performance
2. MAPE
3. MSE
4. RMSE
5. R Squared & Adjusted R Squared

Predictive performance can be measured by comparing Predictions of the models with real values of the target variables, and calculating some average error measure.

**3.1.1 MAE & MAPE (Mean Absolute Error)**

MAE is the average of the absolute difference between the predicted values and observed value. The MAE is a linear score which means that all the individual differences are weighted equally in the average. For example, the difference between 10 and 0 will be twice the difference between 5 and 0. However, same is not true for RMSE which we will discuss more in details further. Mathematically, it is calculated using this formula:

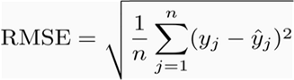


Measures accuracy as the percentage of error.

It is calculated as the average of the unsigned percentage error, as shown in the example below: Many organizations focus primarily on the MAPE when assessing forecast accuracy.

**3.1.2 MSE & RMSE (Mean Square Error & Root Mean Square Error)**

It represents the sample standard deviation of the differences between predicted values and observed values (called residuals). Mathematically, it is calculated using this formula:



Steps to calculate MSE,

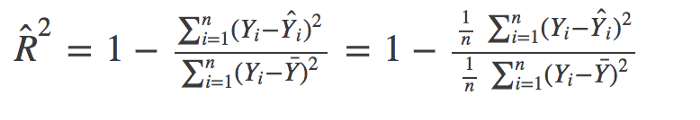
1. Find the regression line.
2. Insert your X values into the linear regression equation to find the new Y values
3. Subtract the new Y value from the original to get the error.
4. Square the errors.
5. Add up the errors.
6. Find the mean.

Calculate RMSE as root of MSE.

## 3.1.3 R Squared (R²) and Adjusted R Squared

R Squared & Adjusted R Squared are often used for explanatory purposes and explains how well your selected independent variable(s) explain the variability in your dependent variable(s). Both these metrics are quite misunderstood and therefore I would like to clarify them first before going through their pros and cons.

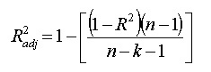
Mathematically, R\_Squared is given by:



*The numerator is MSE ( average of the squares of the residuals) and the denominator is the variance in Y values. Higher the MSE, smaller the R\_squared and poorer is the model.*

**Adjusted R²**

Just like R², adjusted R² also shows how well terms fit a curve or line but adjusts for the number of terms in a model. It is given by below formula:



Where n is the total number of observations and k is the number of predictors. Adjusted R² will always be less than or equal to R²

**Why should you choose Adjusted R² over R²?**There are some problems with normal R² which are solved by Adjusted R². An adjusted R² will consider the marginal improvement added by an additional term in your model. So it will increase if you add the useful terms and it will decrease if you add less useful predictors. However, R² increases with increasing terms even though the model is not actually improving.

**Appendix**

**R code:**

rm(list=ls())

install.packages(c("dmm","dplyr","plyr","reshape","ggplot2","data.table","psych","usdm","caret","DMwR"))

data=read.csv("day.csv",header=T)

newData=subset(data, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual","registered","cnt"))

savedData=newData

dataCasual = subset(newData, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"))

saved\_dataCasual = dataCasual;

dataRegistered = subset(newData, select = c("season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"))

saved\_dataRegistered = dataRegistered

write.csv(dataCasual, "dataCasual.csv",row.names = T)

write.csv(dataRegistered, "dataRegistered.csv",row.names = T)

# Retrive numeric data

numeric\_index\_casual = sapply(dataCasual,is.numeric)

numeric\_data\_casual = dataCasual[,numeric\_index\_casual]

numeric\_data\_cols\_casual = colnames(numeric\_data\_casual)

numeric\_index\_registered = sapply(dataRegistered,is.numeric)

numeric\_data\_registered = dataRegistered[,numeric\_index\_registered]

numeric\_data\_cols\_registered = colnames(numeric\_data\_registered)

#Calculate outliers in dataCasual

dataCasual1 = dataCasual

for(i in numeric\_data\_cols\_casual)

{

print(i)

val\_casual = dataCasual1[,i][dataCasual1[,i]%in%boxplot.stats(dataCasual1[,i])$out]

print(length(val\_casual))

dataCasual1 = dataCasual1[which(!dataCasual1[,i]%in%val\_casual),]

}

# Replace all outliers in dataCasual with NA and impute missing using missing value analysis

dataCasual1 = dataCasual

for(i in numeric\_data\_cols\_casual)

{

val\_casual = dataCasual1[,i][dataCasual1[,i]%in%boxplot.stats(dataCasual1[,i])$out]

dataCasual1[,i][dataCasual1[,i]%in%val\_casual] = NA

}

#Calculate outliers in dataRegistered

dataRegistered1 = dataRegistered

for(i in numeric\_data\_cols\_registered)

{

print(i)

val\_registered = dataRegistered1[,i][dataRegistered1[,i]%in%boxplot.stats(dataRegistered1[,i])$out]

print(length(val\_registered))

dataRegistered1 = dataRegistered1[which(!dataRegistered1[,i]%in%val\_registered),]

}

# Replace all outliers in dataRegistered with NA and impute missing using missing value analysis

dataRegistered1 = dataRegistered

for(i in numeric\_data\_cols\_registered)

{

val\_registered=dataRegistered1[,i][dataRegistered1[,i]%in%boxplot.stats(dataRegistered1[,i])$out]

dataRegistered1[,i][dataRegistered1[,i]%in%val\_registered] = NA

}

# Apply KNN imputation for dataCasual

require(DMwR)

dataCasual1 = knnImputation(dataCasual1,k=5)

# Apply Mean imputation for columns with NA (because of error: there are not sufficient cases)

dataCasual1$holiday[is.na(dataCasual1$holiday)] = mean(dataCasual1$holiday,na.rm = T)

dataCasual1$hum[is.na(dataCasual1$hum)] = mean(dataCasual1$hum,na.rm = T)

dataCasual1$windspeed[is.na(dataCasual1$windspeed)] = mean(dataCasual1$windspeed,na.rm = T)

dataCasual1$casual[is.na(dataCasual1$casual)] = mean(dataCasual1$casual,na.rm = T)

# Apply KNN imputation for dataRegistered

dataRegistered1 = knnImputation(dataRegistered1,k=5)

# Apply Mean imputation for columns with NA (because of error: there are not sufficient cases)

dataRegistered1$holiday[is.na(dataRegistered1$holiday)] = mean(dataRegistered1$holiday,na.rm = T)

dataRegistered1$hum[is.na(dataRegistered1$hum)] = mean(dataRegistered1$hum,na.rm = T)

dataRegistered1$windspeed[is.na(dataRegistered1$windspeed)] = mean(dataRegistered1$windspeed,na.rm = T)

dataRegistered1$registered[is.na(dataRegistered1$registered)] = mean(dataRegistered1$registered,na.rm = T)

# chi-square test of Independency

# factor\_index=sapply(dataCasual1, is.factor)

# factor\_data=dataCasual1[,factor\_index]

#

# for (i in 1:ncol(dataCasual1)-1)

# {

# print(chisq.test(table(dataCasual1$dataCasual1[,length(dataCasual1)],dataCasual1[,i])))

# }

# select Dependent columns from dataCasual

nonDependentCols\_casual = names(dataCasual1)%in% c("casual")

dependentData\_casual = dataCasual1[!nonDependentCols\_casual]

dependentCols\_casual = colnames(dependentData\_casual)

# select Dependent columns from dataRegistered

nonDependentCols\_registered = names(dataRegistered1)%in% c("registered")

dependentData\_registered = dataRegistered1[!nonDependentCols\_registered]

depedentCols\_registered = colnames(dependentData\_registered)

# Feature Scaling

# require(graphics)

# for(i in dependentCols\_casual)

# {

# print(i)

# qqnorm(dependentData\_casual$i)

# hist(dependentData\_casual$i)

# }

# Sampling Techniques

d\_dataCasual = dependentData\_casual

simpleRandomSampling\_dataCasual = d\_dataCasual[sample(nrow(d\_dataCasual),100,replace = F),]

d\_dataRegistered = dependentData\_registered

simpleRandomSampling\_dataRegistered = d\_dataRegistered[sample(nrow(d\_dataRegistered),100,replace = F),]

# # divide train & test data

train\_index\_casual = sample(1:nrow(saved\_dataCasual), 0.8 \* nrow(saved\_dataCasual), prob = NULL)

train\_data\_casual = saved\_dataCasual[train\_index\_casual,]

test\_data\_casual = saved\_dataCasual[-train\_index\_casual,]

train\_index\_registered = sample(1:nrow(saved\_dataRegistered), 0.8 \* nrow(saved\_dataRegistered), prob = NULL)

train\_data\_registered = saved\_dataRegistered[train\_index\_registered,]

test\_data\_registered = saved\_dataRegistered[-train\_index\_registered,]

# # builds decision tree

# library(rpart)

# fit = rpart(registered~., data=train\_data\_registered, method="anova")

# library(MASS)

# predictions = predict(fit, test\_data\_registered[,-12])

# # Calculate MAPE, MSE, RMSE, MAE

# library(DMwR)

# regr.eval(test\_data\_registered[,12], predictions, stats = c('mae','mape','mse','rmse'))

# mae mape mse rmse

# 5.827114e+02 2.279066e+00 7.503266e+05 8.662140e+02

# KNN

# install.packages("caret")

# train\_index\_casual = sample(1:nrow(saved\_dataCasual), 0.8 \* nrow(saved\_dataCasual), prob = NULL)

# train\_data\_casual = saved\_dataCasual[train\_index\_casual,]

# test\_data\_casual = saved\_dataCasual[-train\_index\_casual,]

# train\_index\_registered = sample(1:nrow(saved\_dataRegistered), 0.8 \* nrow(saved\_dataRegistered), prob = NULL)

# train\_data\_registered = saved\_dataRegistered[train\_index\_registered,]

# test\_data\_registered = saved\_dataRegistered[-train\_index\_registered,]

# library(class)

# knn\_predictions\_casual = knn(train\_data\_casual[,1:11], test\_data\_casual[,1:11], train\_data\_casual$casual, k=5)

# knn\_predictions\_registered = knn(train\_data\_registered[,1:11], test\_data\_registered[,1:11], train\_data\_registered$registered, k=5)

# #Accuracy

# knn\_CM\_casual = table(knn\_predictions\_casual, test\_data\_casual$casual)

# sum(diag(knn\_CM\_casual))/nrow(test\_data\_casual)

# TN\_casual = knn\_CM\_casual[0,0]

# FN\_casual = knn\_CM\_casual[1,0]

# TP\_casual = knn\_CM\_casual[1,1]

# FP\_casual = knn\_CM\_casual[0,1]

#

# knn\_CM\_registered = table(knn\_predictions\_registered, test\_data\_registered$registered)

# sum(diag(knn\_CM\_registered))/nrow(test\_data\_registered)

# TN\_registered = knn\_CM\_registered[0,0]

# FN\_registered = knn\_CM\_registered[1,0]

# TP\_registered = knn\_CM\_registered[1,1]

# FP\_registered = knn\_CM\_registered[0,1]

#

# library(DMwR)

# regr.eval(test\_data\_casual[,12], knn\_predictions\_casual, stats = c('mae','mape','mse','rmse'))

# regr.eval(test\_data\_registered[,12], knn\_predictions\_registered, stats = c('mae','mape','mse','rmse'))

# Linear Regression (Registered)

install.packages("usdm")

library(usdm)

vif(saved\_dataCasual[,-12])

vifcor(saved\_dataCasual[,-12], th = 1.0)

lr\_model\_casual = lm(casual~., data = saved\_dataCasual)

summary(lr\_model\_casual)

lr\_prediction\_casual = predict(lr\_model\_casual, test\_data\_casual[,1:11])

library(DMwR)

library(MASS)

regr.eval(test\_data\_registered[,12], lr\_prediction\_casual, stats = c('mae','mape','mse','rmse'))

vif(saved\_dataRegistered[,-12])

vifcor(saved\_dataRegistered[,-12], th = 1.0)

lr\_model\_registered = lm(registered~., data = saved\_dataRegistered)

summary(lr\_model\_registered)

lr\_prediction\_registered = predict(lr\_model\_registered, test\_data\_casual[,1:11])

library(DMwR)

library(MASS)

regr.eval(test\_data\_registered[,12], lr\_prediction\_registered, stats = c('mae','mape','mse','rmse'))

# KMeans Clustering

install.packages("NbClust")

library(NbClust)

d\_casual = saved\_dataCasual

clusters\_casual = NbClust(d\_casual, min.nc = 2, max.nc = 10, method = "kmeans")

barplot(table(clusters\_casual$Best.nc[1,]), xlab="X", ylab="Y", main="")

kmeans\_model\_casual = kmeans(d\_casual,4,nstart = 25)

cluster\_accuracy\_casual = table(d\_casual$casual,kmeans\_model\_casual$cluster)

d\_registered = saved\_dataRegistered

clusters\_registered = NbClust(d\_registered, min.nc = 2, max.nc = 10, method = "kmeans")

barplot(table(clusters\_registered$Best.nc[1,]), xlab="X", ylab="Y", main="")

kmeans\_model\_registered = kmeans(d\_registered,4,nstart = 25)

cluster\_accuracy\_registered = table(d\_registered$registered,kmeans\_model\_registered$cluster)

library(ggplot2)

library(scales)

library(psych)

library(gplots)

newData\_casual = dataCasual

newData\_registered = dataRegistered

# Bar plot ( Categorical variables VS Target variable)

# Casual Data

ggplot(newData\_casual, aes\_string(x=newData\_casual$season, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("season") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Season vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$mnth, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("month") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Month vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$holiday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("holiday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Holiday vs Casual ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weekday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("weekday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weekday vs Casul ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$workingday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("workingday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Working Day vs Casual") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weathersit, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("Weather") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot weather vs casual ") + theme(text = element\_text(size=10))

# Registered Data

ggplot(newData\_registered, aes\_string(x=newData\_registered$season, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("season") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Season vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$mnth, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("month") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Month vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$holiday, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("holiday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Holiday vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_casual, aes\_string(x=newData\_casual$weekday, y=newData\_casual$casual)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("weekday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weekday vs Registered ") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$workingday, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("workingday") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Working Day vs Registered") + theme(text = element\_text(size=10))

ggplot(newData\_registered, aes\_string(x=newData\_registered$weathersit, y=newData\_registered$registered)) +

geom\_bar(stat = "identity",fill="Blue") + theme\_bw() +

xlab("Weather") + ylab("casual") +

scale\_y\_continuous(breaks = pretty\_breaks(n=10)) +

ggtitle("Bar plot Weather vs Registered ") + theme(text = element\_text(size=10))

**Python code:**

import os

os.getcwd()

os.chdir("C:/Users/gopin/Documents/R/BikeRental-Project")

import pandas as pd

import numpy as np

import matplotlib as mlt

import matplotlib.pyplot as plt

from sklearn.neighbors import KNeighborsClassifier

data = pd.read\_csv("day.csv")

savedData = data

dataCasual = savedData[["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"]]

dataRegsitered = savedData[["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"]]

dCasual = dataCasual.copy()

dRegistered = dataRegsitered.copy()

# Store continuous variable names

cnames\_C = ["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","casual"]

cnames\_R = ["season","yr","mnth","holiday","weekday","workingday","weathersit","temp","atemp","hum","windspeed","registered"]

# Detect outliers & delete(Casual)

for i in cnames\_C:

q75, q25 = np.percentile(dCasual.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dCasual = dCasual.drop(dCasual[dCasual.loc[:,i] < innerfence].index)

dCasual = dCasual.drop(dCasual[dCasual.loc[:,i] > outerfence].index)

# Replace with NA

dCasual = dataCasual.copy()

for i in cnames\_C:

q75, q25 = np.percentile(dCasual.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dCasual.loc[dCasual[i] < innerfence,:i] = np.nan

dCasual.loc[dCasual[i] > outerfence,:i] = np.nan

# Calculate Missing values

missing\_C = pd.DataFrame(dCasual.isnull().sum())

# Impute using Mode method

for i in cnames\_C:

dCasual[i] = dCasual[i].fillna(dCasual[i].mode()[0])

#missing\_C = pd.DataFrame(dCasual.isnull().sum())

savedDataCasual = dCasual

# Detect outliers & delete (Registered)

for i in cnames\_R:

q75, q25 = np.percentile(dRegistered.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dRegistered = dRegistered.drop(dRegistered[dRegistered.loc[:,i] < innerfence].index)

dRegistered = dRegistered.drop(dRegistered[dRegistered.loc[:,i] > outerfence].index)

# Replace with NA

dRegistered = dRegistered.copy()

for i in cnames\_R:

q75, q25 = np.percentile(dRegistered.loc[:,i],[75,25])

iqr = q75 - q25

innerfence = q25 - (iqr \* 1.5)

outerfence = q75 + (iqr \* 1.5)

dRegistered.loc[dRegistered[i] < innerfence,:i] = np.nan

dRegistered.loc[dRegistered[i] > outerfence,:i] = np.nan

# Calculate Missing values

missing\_R = pd.DataFrame(dRegistered.isnull().sum())

# Impute using Mode method

for i in cnames\_R:

dRegistered[i] = dRegistered[i].fillna(dRegistered[i].mode()[0])

savedDataRegistered = dRegistered

# Feature selection

'''import seaborn as sns

from scipy.stats import chi2\_contingency

from random import randrange,uniform

for i in dCasual.columns:

print(i)

p = chi2\_contingency(pd.crosstab(dCasual['casual'],dCasual[i]))

'''

# Gives Barplot

# %matplotlib inline

# plt.hist(dCasual['weekday'], bins='auto')

# Sampling using Systematic sampling

simpleRandomSampling\_C = dCasual.sample(100)

simpleRandomSampling\_R = dRegistered.sample(100)

from sklearn import tree

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import train\_test\_split

# Train and Test data

dCasual = savedDataCasual.copy()

xc = dCasual.values[:, 0:11]

yc = dCasual.values[:, 11]

xc\_train, xc\_test, yc\_train, yc\_test = train\_test\_split(xc,yc,test\_size=0.2)

# Linear Regression model for Data Casual

import statsmodels.api as sm

train\_c, test\_c = train\_test\_split(dCasual,test\_size=0.2)

model\_C = sm.OLS(train\_c.iloc[:,11], train\_c.iloc[:,0:11]).fit()

model\_C.summary()

dRegistered = savedDataRegistered.copy()

xr = dRegistered.values[:, 0:11]

yr = dRegistered.values[:, 11]

xr\_train, xr\_test, yr\_train, yr\_test = train\_test\_split(xr,yr,test\_size=0.2)

# Linear Regression model for Data Registered

import statsmodels.api as sm

train\_c, test\_c = train\_test\_split(dRegistered,test\_size=0.2)

model\_R = sm.OLS(train\_c.iloc[:,11], train\_c.iloc[:,0:11]).fit()

model\_R.summary()

import ggplot

from ggplot import \*

dC = savedDataCasual.copy()

ggplot(dC, aes(x='season', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Casual") + ggtitle("Barplot\_seasonVScasual") + theme.bw()

ggplot(dC, aes(x='holiday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Holiday") + ylab("Casual") + ggtitle("Barplot\_holidayVScasual") + theme.bw()

ggplot(dC, aes(x='mnth', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Month") + ylab("Casual") + ggtitle("Barplot\_monthVScasual") + theme.bw()

ggplot(dC, aes(x='weather', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weather") + ylab("Casual") + ggtitle("Barplot\_weatherVScasual") + theme.bw()

ggplot(dC, aes(x='weekday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weekday") + ylab("Casual") + ggtitle("Barplot\_weekDayVScasual") + theme.bw()

ggplot(dC, aes(x='workkingday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Working Day") + ylab("Casual") + ggtitle("Barplot\_workingDayVScasual") + theme.bw()

ggplot(dC, aes(x='season', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Casual") + ggtitle("Barplot\_seasonVScasual") + theme.bw()

dR = savedDataRegistered.copy()

ggplot(dR, aes(x='season', y='registered')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Season") + ylab("Registered") + ggtitle("Barplot\_seasonVSregistered") + theme.bw()

ggplot(dR, aes(x='holiday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Holiday") + ylab("Registered") + ggtitle("Barplot\_holidayVSregistered") + theme.bw()

ggplot(dR, aes(x='mnth', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Month") + ylab("Registered") + ggtitle("Barplot\_monthVSregistered") + theme.bw()

ggplot(dR, aes(x='weather', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weather") + ylab("Registered") + ggtitle("Barplot\_weatherVSregistered") + theme.bw()

ggplot(dR, aes(x='weekday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Weekday") + ylab("Registered") + ggtitle("Barplot\_weekDayVSregistered") + theme.bw()

ggplot(dR, aes(x='workkingday', y='casual')) +\

geom\_bar(fill="blue") +\

scale\_color\_brewer(type="diverging", palette=4) +\

xlab("Working Day") + ylab("Registered") + ggtitle("Barplot\_workingDayVSregistered") + theme.bw()

**Visualizations:**

