Godavari College of Engineering, Jalgaon

Subject Name: Machine Learning

Practical No : 01 **Date: 8/09/2020**

Title: Regression Analysis and Plot interpretation.

Aim: Study and implementation of Regression Analysis and plot interpretation.

Theory:

Regression analysis is a very widely used statistical tool to establish a relationship model between two variables. One of these variable is called predictor variable whose value is gathered through experiments. The other variable is called response variable whose value is derived from the predictor variable.

In Linear Regression these two variables are related through an equation, where exponent (power) of both these variables is 1. Mathematically a linear relationship represents a straight line when plotted as a graph. A non-linear relationship where the exponent of any variable is not equal to 1 creates a curve.
The general mathematical equation for a linear regression is-

$$y=ax+b$$

Following is the description of the parameters used −

- **y** is the response variable.
- **x**is the predictor variable.
- **A** and **b** are constants which are called the coefficients.

Steps to Establish a Regression:

A simple example of regression is predicting weight of a person when his height is known. To do this we need to have the relationship between height and weight of a person.

The steps to create the relationship is-

- Carry out the experiment of gathering a sample of observed values of height and corresponding weight.
- Create a relationship model using the **lm()**functions in R.
- Find the coefficients from the model created and create the mathematical equation using these

- Get a summary of the relationship model to know the average error in prediction. Also called **residuals**.
- To predict the weight of new persons, use the **predict()**function in R.

lm() Function

This function creates the relationship model between the predictor and the response variable.

Syntax:

The basic syntax for **lm()**function in linear regression is-

lm(formula,data)

Following is the description of the parameters used −

- **formula** is a symbol presenting the relation between x and y.
- **Data** is the vector on which the formula will be applied.

Predict() Function

The basic syntax for predict() in linear regression is-

predict(object,newdata)

Following is the description of the parameters used-

- **object** is the formula which is already created using the lm()function.
- **New data** is the vector containing the new value for predictor variable.

Source Code:

```
# Values of height
151, 174, 138, 186, 128, 136, 179, 163, 152, 131

# Values of weight.

63, 81, 56, 91, 47, 57, 76, 72, 62, 48


#Apply the lm() function

relation ← lm(y~x)
print(relation)

#get the summary of relatonship
print(summary(relation))
```

#Find weight of person with height 170
a ← data.frame(x=170)
result ← predict(relation, a)
print(result)

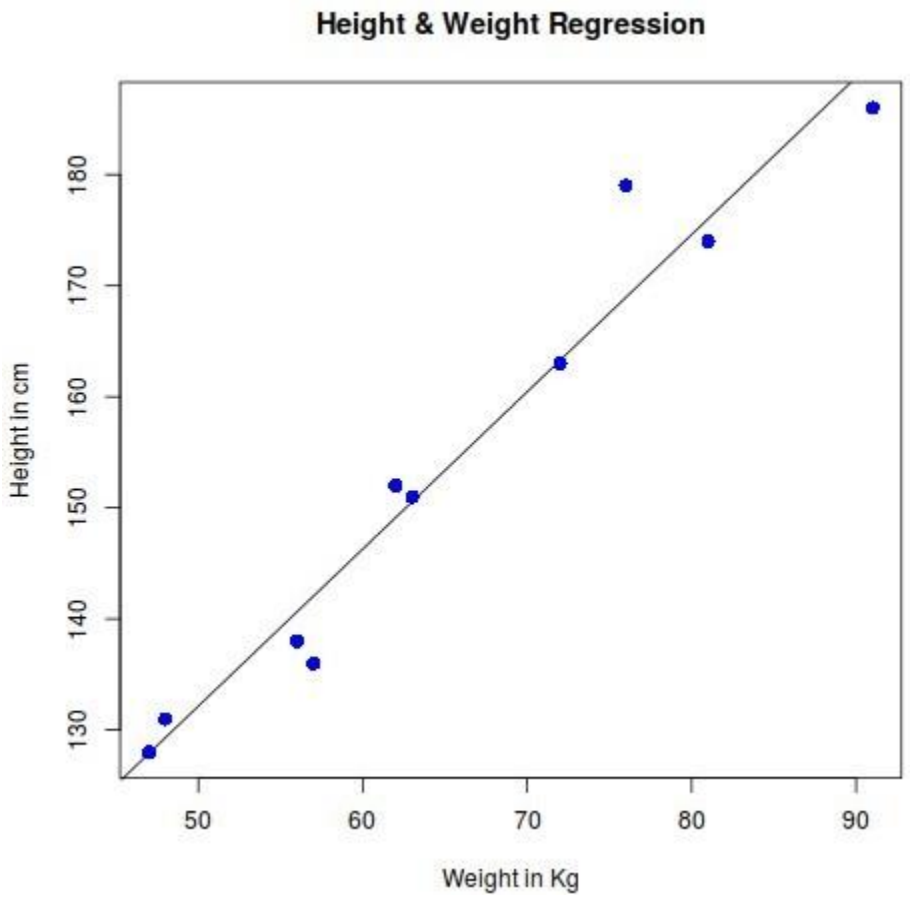
#Give the chart file a name
png(file = “linearregression.png”)

#Plot the Chart
plot(y,x,col = “blue”, main=”Height & Weight Regression”,
abline(lm(x~y)), cex = 1.3, pch = 16, xlab = “Weight in kg”, ylab = “Height in cm”)

#Save
thefiledev.off()

Output:

```
> x <- c(151,174,138,186,128,136,179,163,152,131)
> y <- c(63,81,56,91,47,57,76,72,62,48)
> relation <- lm(y~x)
> print(relation)

Call:
lm(formula = y ~ x)

Coefficients:
(Intercept)            x
   -38.4551       0.6746

> print(summary(relation))

Call:
lm(formula = y ~ x)

Residuals:
    Min      1Q  Median      3Q     Max
-6.3002 -1.6629  0.0412  1.8944  3.9775

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -38.45509    8.04901  -4.778  0.00139 **
x             0.67461    0.05191  12.997 1.16e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.253 on 8 degrees of freedom
Multiple R-squared:  0.9548,    Adjusted R-squared:  0.9491
F-statistic: 168.9 on 1 and 8 DF,  p-value: 1.164e-06

> #find weight of a person with height 170.
> result <- predict(relation, a)
> print(result)
       1
76.22869
> # Give the chart file a name.
> png(file = "linearregression.png")
> # Plot the chart.
> plot(y,x,col = "blue",main = "Height & Weight Regression", abline(lm(x~y)),cex=1.3,pch=16,xlab="Weight in Kg",ylab="Height in cm")
> # Save the file.
> dev.off()
png
  2
```



Conclusion: In this Practical I learn and implementation of Regression Analysis and plot interpretation.

Godavari College of Engineering, Jalgaon

Subject Name: Machine Learning

Practical No : 02 **Date : 30-09-2020**

Title: Logistic Regression Analysis in R.

Aim: Study and implementation of logistic regression analysis in R.

Theory:

The Logistic Regression is a regression model in which the response variable (dependent variable) has categorical values such as True/False or 0/1. It actually measures the probability of a binary response as the value of response variable based on the mathematical equation relating it with the predictor variables.

The general mathematical equation for logistic regression is-

$$y = 1/(1+e^{-(a+b1x1+b2x2+b3x3+...)})$$

Following is the description of the parameters used −

- **y** is the response variable.
- **X** is the predictor variable.
- **a** and **b** are the coefficients which are numeric constants.

The function used to create the regression model is the **glm()**function.

Syntax:

The basic syntax for **glm()**function in logistic regression is-

glm(formula, data, family)

Following is the description of the parameters used −

- **formula** is the symbol presenting the relationship between the variables.
- **Data** is the data set giving the values of these variables.
- **Family** is R object to specify the details of the model. It's value is binomial for logistic regression.

Sourse Code:

The in-built data set "mtcars" describes different models of a car with their various engine specifications. In "mtcars" data set, the transmission mode (automatic or manual) is

described by the column am which is a binary value (0 or 1).Wecan create a logistic regression model between the columns "am" and 3 other columns - hp, wt andcyl.

#Select some columns from mtcars.

Input ←
mtcars[,c(“am”,”cyl”,”hp”,”wt”)]
print(head(input))

#We use the **glm()**function to create the regression model and get its summary for analysis.

am.data = glm(formula = am ~ cyl + hp + wt, data = input, family =
binomial) print(summary(am.data))

Output:

```
> input <- mtcars[,c("am","cyl","hp","wt")]
>
> print(head(input))
                  am cyl  hp    wt
Mazda RX4          1   6 110 2.620
Mazda RX4 Wag      1   6 110 2.875
Datsun 710         1   4  93 2.320
Hornet 4 Drive     0   6 110 3.215
Hornet Sportabout  0   8 175 3.440
Valiant            0   6 105 3.460
> input <- mtcars[,c("am","cyl","hp","wt")]
>
> am.data = glm(formula = am ~ cyl + hp + wt, data = input, family = binomial)
>
> print(summary(am.data))

Call:
glm(formula = am ~ cyl + hp + wt, family = binomial, data = input)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-2.17272  -0.14907  -0.01464   0.14116   1.27641

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) 19.70288    8.11637   2.428   0.0152 *
cyl          0.48760    1.07162   0.455   0.6491
hp           0.03259    0.01886   1.728   0.0840 .
wt          -9.14947    4.15332  -2.203   0.0276 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 43.2297  on 31  degrees of freedom
Residual deviance:  9.8415  on 28  degrees of freedom
AIC: 17.841

Number of Fisher Scoring iterations: 8

>
```

Conclusion: In the summary as the p-value in the last column is more than 0.05 for the variables "cyl" and "hp", we consider them to be insignificant in contributing to the value of the variable "am". Only weight (wt) impacts the "am" value in this regression model.

Godavari college of Engineering, Jalgaon

Subject Name : Machine Learning

Practical No :03 **Date: 19-10-2020**

Title: Random Forest and Parameter Tuning in R

Aim: Study and implementation of Random Forest and Parameter Tuning in R

Theory:

In the random Forest approach , a large m=number of decision trees are created. Every observation is fed into every decision tree. The most Common outcome for each observation is used as final output. A new observation is fed into all the trees and taking a majority vote for each classification model.

An error estimation is made for the cases which were not used while building the tree. That is called an OOB(Out-of -bag) error estimation which is mentioned as a percentage.

The R package "randomForest" is used to create random forest.

Install R Package:

Use the below command in R console to install the package. Your also have install the dependent packages if any.

```
install.packages('randomForest')
```

The Package "randomForest has the function **randomForest()** which is used to create and analy" as a percentage.The package'randomForest"has the function **randomForest() which** is used to create and analyze random forests.

Syntax:

The basic syntax for creating a random forest in R is−

```
randomForest(formula,data)
```

Following is the description of the parameters used−

formula is a formula describing the predictor and response variables.
Data is the name of the data set used.

Input Data:

We will use the R inbuilt dataset named reading Skills to create tree. It descibes the score of someone's reading sills if we know the variable "age","shoesize","score" and whethe person is a native speaker.
Here is the sample data.

```
#Load the party package. It will automatically
#load other required packages.
library(party)

#Print some records from dataset reading
#Skills.
print(head(readingSkills))
```

When we execute the above code, it produces the following result and chart—

Output:-

```
Native Speaker age shoe Size
Score
1       yes  524.8318932.29385
2       yes 625.9523836.63105
3        no 130.4217049.60593
4       yes 728.6645040.28456
5       yes 131.8820755.46085
6       yes 1030.0784352.83124
Loading required package:
methods Loading required package
:  grid
.....
.....
```

Example:

We will use the **randomForest()**function to create the decision tree andsee it's graph.

```
#Load the party package. It will automatically
#load other required packages.
library(party)
library(randomForest)

# Create the forest.
output.forest< randomForest(nativeSpeaker~age+shoeSize+score,
      readingSkills)
#View the forest
#results
print(output.forest)
```

When we execute the above code, it produces the following result—

Output:-

Call:

```
randomForest(formula=nativeSpeaker~age+shoeSize+score,data=
readingSkills)
Type of random forest: classification
Number of trees :500
No. of variables tried data each split: 1
OOB estimate of error rate: 1%
Confusion matrix :
 noyesclass.errorn
 o   991     0.01
 yes 199     0.01
```

Conclusion:- Study and implementation of Random Forest and Parameter Tuning in R. From the random forest shown above we can conclude that the shoe size and score are the important factors deciding if some one is an active speak error_not. Also the model has only 1% error which means we can predict with 99% accuracy.

Godavari college of Engineering , Jalgaon

Subject Name: Machine Learning

Practical No: 04 **Date: 7-11-2020**

Title: Clustering Algorithm and Evaluation in R

Aim: Study and implementation of Clustering Algorithm and Evaluation in R

Theory:

K-means clustering aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster. This results in a partitioning of the data space into Voronoi cells.

Given a set of observations $(x_1, x_2, \ldots, x_n)$, where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k groups $G = \{G_1, G_2, \ldots, G_k\}$ so as to minimize the within-cluster sum of squares (WCSS) defined as follows −

$$\text{argmin}\sum_{i=1}^{k}\sum_{x\in S_i}\|x-\mu_i\|_2$$

The later formula shows the objective function that is minimized in order to find the optimal prototypes in k-means clustering. The intuition of the formula is that we would like to find groups that are different with each other and each member of each group should be similar with the other members of each cluster.

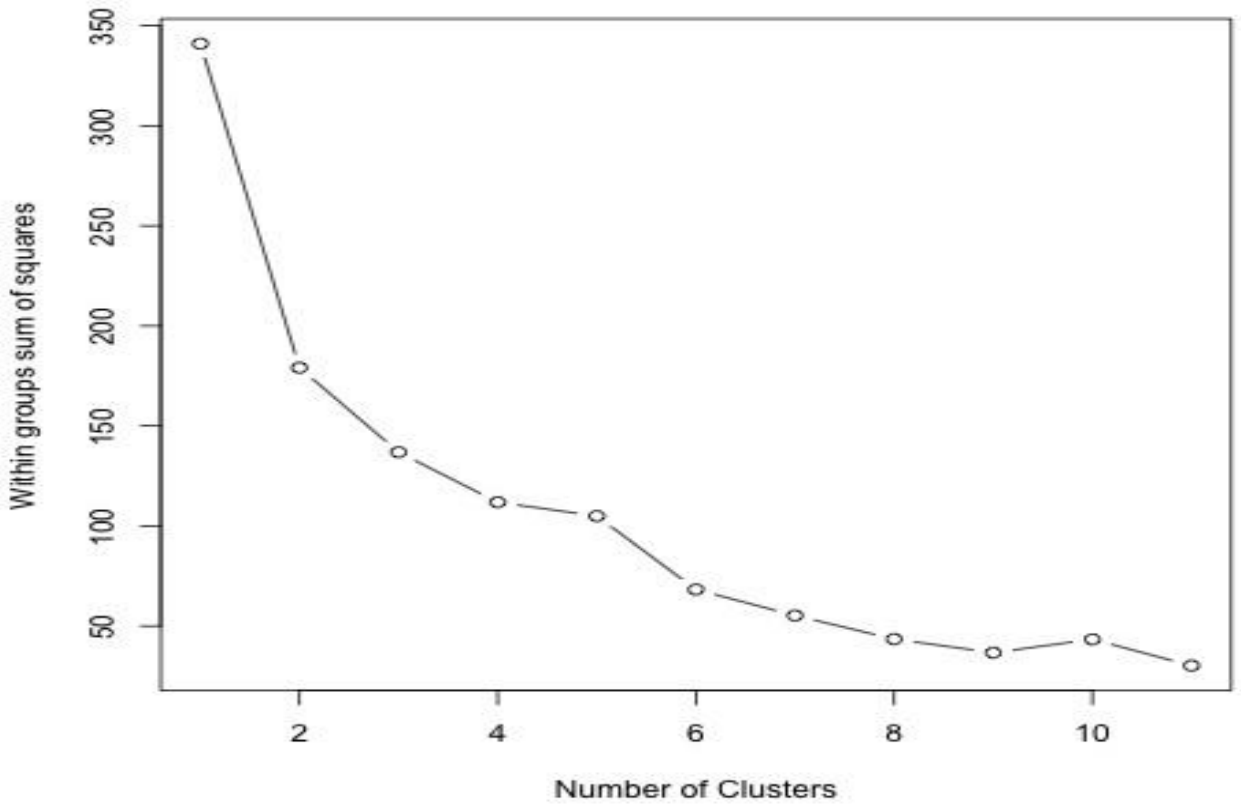
The following example demonstrates how to run the k-means clustering algorithm in R.

Input Data:

```
library(ggplot2)
# Prepare Data
data=mtcars

# We need to scale the data to have zero mean and unit variance
data<- scale(data)

# Determine number of clusters
wss<-(nrow(data)-1)*sum(apply(data,2,var))
for(i in2:dim(data)[2]){
wss[i]<- sum(kmeans(data, centers = i)$withinss)
}

# Plot the clusters
plot(1:dim(data)[2],wss, type ="b",xlab="Number of Clusters",
ylab="Within groups sum of squares")
```

In order to find a good value for K, we can plot the within groups sum of squares for different values of K. This metric normally decreases as more groups are added, we would like to find a point where the decrease in the within groups sum of squares starts decreasing slowly. In the plot, this value is best represented by K = 6.

Output:



Now that the value of K has been defined, it is needed to run the algorithm with that value.

Input data:

```
# K-Means Cluster Analysis
fit<- kmeans(data, 5) # 5 cluster solution

# get cluster means
aggregate(data,by = list(fit$cluster),FUN = mean)

# append cluster assignment
data<- data.frame(data, fit$cluster)
```

Output data:

```
Group.1        mpg         cyl       disp         hp        drat
wt
1       1  0.1082193 -0.58493208 -0.4486701 -0.6496905 -0.04967936 -
0.02346989
2       2 -0.0565170  0.08165718 -0.1977204  0.4980222  0.63297355 -
0.36767056
3       3  1.3739630 -1.22485777 -1.1370289 -0.9643131  1.03241235 -
1.21514154
4       4 -1.2395370  1.01488215  1.4981449  1.1859553 -0.70427805
1.50088285
5       5 -0.5483556  1.01488215  0.6850701  0.3400164 -1.02222599
0.48169844
        qsec         vs         am       gear       carb fit.cluster
1  1.1854841  1.1160357 -0.8141431 -0.1573201 -0.4145882    3.142857
2 -1.1483763 -0.8680278  1.1899014  1.3271364  1.1479487    4.333333
3  0.4763722  1.1160357  1.1899014  0.6171790 -0.8568156    2.000000
4 -0.5342923 -0.8680278 -0.8141431 -0.9318192  0.7352031    1.000000
5 -0.2958964 -0.8680278 -0.8141431 -0.9318192 -0.2376972    4.000000
  fit.cluster.1 fit.cluster.2 fit.cluster.3
1             4             2             5
2             5             1             3
3             1             3             4
4             3             4             2
5             2             5             1
```

Conclusion: In this practical we Study and implementation of K-means Clustering Algorithm and Evaluation in R.