# 21. Mean:

height<-c(150,174,138,186,128,136,171,163,152,131)

result.mean <-mean(height) print(result.mean)

# Barplot:

temperatures<-c(20,22,25,29,23,27,28) result <- barplot(temperatures,

main="MaximumTemperaturesinaWeek", xlab = "Degree Celsius",

ylab= "Day",

names.arg=c("Sun","Mon","Tue","Wed","Thu","Fri","Sat"), col = "blue",

)

print(result)

# Boxplot:

b<- c(10,12,13,14,17,19,20,30,50,70,90,100)

print(boxplot(b,col="green"))

# Decisiontree:

library(rpart) library(rpart.plot)

data=read.csv("C:\\Users\\arunk\\OneDrive\\Desktop\\DWDM\\Gender.csv") tree <- rpart(Height ~ Gender+Weight,data)

a<-data.frame(Gender=c("Male"),Weight=c(85)) result <- predict(tree,a)

print(result) rpart.plot(tree)

tree1<-rpart(Gender~Height+Weight,data) a<-data.frame(Height=c(170),Weight=c(85)) result <- predict(tree,a)

print(result) rpart.plot(tree1)

# Division:

num1=as.integer(readline(prompt="Enterthenumber1:")) num2 =as.integer(readline(prompt = "Enter a number2:")) div=num1/num2

print((paste("Division:",div)))

# Histogram:

temperatures<-c(20,22,25,29,23,27,28)

result<-hist(temperatures,

main="MaximumTemperaturesinaWeek", xlab = "Degree Celsius",

ylab= "Day",

names.arg=c("Sun","Mon","Tue","Wed","Thu","Fri","Sat"), col="green"

)

print(result)

# Linearregression:

x<-c(150,174,138,186,128,136,171,163,152, 131)

y<-c(63,81,56, 91,47,57,76,72,62,48)

relation <-lm(y~x) print(summary(relation)) a <-data.frame(x=170)

result<-predict(relation,a) print(result)

png(file="linear\_regression.png")

plot(y,x,col="red",main="HeightandWeightRegression",abline(lm(x~y)),cex=

1.3,pch=16,xlab="WeightinKg",ylab="Heightincm") dev.off()

# Median:

height<-c(150,174,138,186,128,136,171,163,152,131)

result.median <-median(height) print(result.median)

# Minmax normalization:

original\_vector<-c(10,20,30,40,50)

normalized\_vector<-(original\_vector-min(original\_vector))/(max(original\_vector)

* 1. min(original\_vector)) print(normalized\_vector)

original\_vector<-c(100,200,309,40,50,60,70,80,90,10)

normalized\_vector<-(original\_vector-min(original\_vector))/(max(original\_vector)

* 1. min(original\_vector)) print(normalized\_vector)

# Mode:

getmode<- function(v)

{

uniqv <- unique(v) uniqv[which.max(tabulate(match(v, uniqv)))]

}

v<-c(150,174,138,186,128, 136,171,163,152, 131,171,131,171)

result<-getmode(v) print(result)

1. **Multiple Regression:** d=read.csv("C:\\Users\\arunk\\OneDrive\\Desktop\\DWDM\\set1.csv") View(d)

summary(d) plot(d$Glucose,d$DiabetesPedigreeFunction) p1=runif(nrow(d))

p2=order(p1) training\_ds=d[p2[1:25],] test\_ds=d[p2[26:39],]

Multiple\_resgression=lm(DiabetesPedigreeFunction~Glucose+Age, data=training\_ds)

abline(Multiple\_resgression,col="red") summary(Multiple\_resgression) plot(Multiple\_resgression)

pred\_values=predict(Multiple\_resgression,newdata = test\_ds) test\_ds$pred\_DiabetesPedigreeFunction=pred\_values View(test\_ds)

# Multiplication:

num1=as.integer(readline(prompt = "Enter the num 1:")) num2=as.integer(readline(prompt="Enteranumber2:")) mul=num1\*num2

print((paste("Multiplication:",mul)))

# odd or Even:

num=as.integer(readline(prompt="Enteranumber:")) if (num %% 2 ==0){

print(paste(num,"isEvennumber!!"))

}else{

print(paste(num,"isOddnumber!!"))

}

# pie Chart:

a<- c(80,70,50,60,70,100)

result<- (pie(a,main="piechart",labels=c("student1","student2","student3","student 4","student 5","student 6"),

col=c("red","orange","yellow","blue","green","black"))) print(result)

# Quantile:

names<-c("Ram","Shyam","Kumar") age<-c(23,24,35)

marks<-c(88,78,25)

df<-data.frame(names,age,marks) quantile(df $age) write.csv(df,"datafr.csv")

# Range:

names<-c("Ram","Shyam","Kumar") age<-c(23,24,35)

marks<-c(88,78,25)

df<-data.frame(names,age,marks) range(df $age) write.csv(df,"datafr.csv")

# Scatterplot:

input<-mtcars[,c('wt','mpg')] print(head(input))

plot(x=input$wt,y=input$mpg, xlab = "Weight",

ylab = "Milage", xlim=c(0.5,3.5), ylim = c(15, 30),

main="WeightvsMilage"

)

# Subtraction:

num1=as.integer(readline(prompt = "Enter the num 1:")) num2=as.integer(readline(prompt="Enteranumber2:")) sub=num1-num2

print((paste("subractionvalue:",sub)))

# Z-Scorenormalization:

original\_vector<-c(3,5,5,8,9,12,12,13,15,16,17,19,22,24,25,134)

x<-mean(original\_vector) print(paste("Mean:",x))

u<-sd(original\_vector) print(paste("S.D:",u))

normalized\_vector<-(original\_vector-x)/u print(normalized vector)

# K-Means:

#Loadadataset data(iris)

#Selectthevariablestobeusedforclustering

x<-iris[,c("Sepal.Length","Sepal.Width","Petal.Length","Petal.Width")] # Perform K-means clustering with K=3

kmeans\_model<-kmeans(x,centers=3) # Print the results

kmeans\_model

# Create a scatterplot of the first two variables with points colored by cluster library(ggplot2) ggplot(iris,aes(x=Sepal.Length,y=Sepal.Width,color=factor(kmeans\_model$cluste))

) +geom\_point()

# NormalDistribution:

x<-rnorm(100,mean=0,sd=1) hist(x)

dnorm(1,mean=0,sd=1) pnorm(1,mean=0,sd=1)

# Array:

vector1<-c(5,9,3)

vector2<-c(10,11,12,13,14,15)

result<-array(c(vector1,vector2),dim=c(3,3,2)) print(result)

# SquareRoot:

x <-4

sqrt(x)

# LineChart:

v<- c(17,25,38,13, 41)

plot(v,type="o")

1. **Random Forest:** install.packages("caTools") install.packages("randomForest") library(caTools) library(randomForest)

split<-sample.split(iris,SplitRatio=0.7) split

train<-subset(iris,split=="TRUE") test<-subset(iris,split=="FALSE") set.seed(120)

classifier\_RF=randomForest(x=train[-5],y=train$Species,ntree=500) classifier\_RF

y\_pred=predict(classifier\_RF,newdata=test[-5]) confusion\_mtx = table(test[, 5], y\_pred) confusion\_mtx

plot(classifier\_RF) importance(classifier\_RF) varImpPlot(classifier\_RF)

# ConfusionMatrix:

set.seed(123)

data<-data.frame(Actual=sample(c("True","False"),100,replace=TRUE), Prediction = sample(c("True","False"), 100, replace = TRUE)

)

table(data$Prediction,data$Actual)

# Chi Square:

library (MASS) print(str(survey))

stu\_data = data.frame(survey$Smoke,survey$Exer) stu\_data = table(survey$Smoke,survey$Exer) print(stu\_data)

print(chisq.test(stu\_data))

# Decimal Scaling:

library(caret)

gfg<-c(244,753,596,645,874,141,639,465,999,654)

ss <- preProcess(as.data.frame(gfg), method=c("range")) gfg <- predict(ss, as.data.frame(gfg))

gfg

# Apriori Algorithm:

library(arules)

library(arulesViz) library(RColorBrewer) data("Groceries")

rules<-apriori(Groceries,parameter=list(supp=0.01,conf=0.2)) inspect(rules[1:10])

arules::itemFrequencyPlot(Groceries,topN=20,col=brewer.pal(8,'Pastel2'),main

='RelativeItemFrequencyPlot',type="relative",ylab="ItemFrequency(Relative)")