#####################################################################

**GOF Catalog and 8 Problems (Clsutering algorithms)**

###############################################################################################

**Create dataset for results of Kmeans clustering**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

Km1 <- kmeans(Temp\_Dataset[,3:Len], 3)

# a4 as Vector of Predicted clusters

a1 <- as.character(Km1[1])

a2 <- strsplit(a1, "") ;

a3 <- unlist(a2) ;

a4 <- as.vector(as.numeric(a3))

a4 <- a4[!is.na(a4)]

Predicted\_Class <- a4[length(a4)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("Kmeans\_GoF\_Probles8.csv"))

########################################################################

###############################################################################################

**Create dataset for results of Hirarchical clustering (Euclidean )**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

distance <- dist(Temp\_Dataset [,-1], method="euclidean")

cluster <- hclust(distance, method="average")

group.3 <- cutree(cluster, k = 3)

Predicted\_Class <- group.3 [length(group.3)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("Hirarchical\_GoF\_Probles8.csv"))

########################################################################

###############################################################################################

**Create dataset for results of Hirarchical clustering (Manhattan )**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

distance <- dist(Temp\_Dataset [,-c(1,2)], method="manhattan")

cluster <- hclust(distance, method="average")

group.3 <- cutree(cluster, k = 3)

Predicted\_Class <- group.3 [length(group.3)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("Hirarchical\_Manhatan\_GoF\_Probles8.csv"))

########################################################################

###############################################################################################

**Create dataset for results of Fuzzy Cmeans clustering**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

Km1 <- cmeans(Temp\_Dataset [,-c(1,2)], centers=3, iter.max=100, m=2, method="cmeans")

# a4 as Vector of Predicted clusters

a1 <- as.character(Km1[3])

a2 <- strsplit(a1, "") ;

a3 <- unlist(a2) ;

a4 <- as.vector(as.numeric(a3))

a4 <- a4[!is.na(a4)]

Predicted\_Class <- a4[length(a4)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("Cmeans\_GoF\_Probles8.csv"))

########################################################################

###############################################################################################

**Create dataset for results of PAM clustering(Euclidian)**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

Km1 <- pam(Temp\_Dataset [,-c(1,2)], 3, FALSE, "euclidean")

# a4 as Vector of Predicted clusters

a1 <- as.character(Km1[3])

a2 <- strsplit(a1, "") ;

a3 <- unlist(a2) ;

a4 <- as.vector(as.numeric(a3))

a4 <- a4[!is.na(a4)]

Predicted\_Class <- a4[length(a4)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("PAM\_Euclidien\_GoF\_Probles8.csv"))

########################################################################

###############################################################################################

**Create dataset for results of PAM clustering(**manhattan**)**

#################################################################################################

Dataset.file<- "D:/Experimentjournal/clusteringwithproblems/GoF8Problems.csv"

Dataset <- read.csv(Dataset.file,header=TRUE)

setwd("D:/Experimentjournal/clusteringwithproblems/GoFDatasetsWithProblemAllocation8Problems")

temp = list.files(pattern="\*.csv")

for (i in 1:length(temp)) assign(temp[i], read.csv(temp[i]))

totalfiles<- length(temp)

Out <- matrix(NA, nrow= totalfiles, ncol=4)

for(k in 1:totalfiles)

{

st<-substr(temp[k],1, nchar(temp[k])-4)

fp <- regexpr('\_', temp[k])

st<-substr(temp[k],fp+1, nchar(temp[k])-4)

sp<-regexpr('\_',st)

st1 <-substr(st,1, sp-1)

record <- data.frame(Dataset[grep(st1, Dataset$Problem),])

Categor <- toString(record[1,2])

Actual\_Class <- record[1,3]

Temp\_Dataset<- read.csv(temp[k],header=TRUE)

Len <-length(Temp\_Dataset)

DS <- as.matrix(Temp\_Dataset)

y <- which(is.na(Temp\_Dataset)==TRUE)

DS[y] <- Categor

Temp\_Dataset <-data.frame(DS)

Km1 <- pam(Temp\_Dataset [,-c(1,2)], 3, FALSE, "manhattan")

# a4 as Vector of Predicted clusters

a1 <- as.character(Km1[3])

a2 <- strsplit(a1, "") ;

a3 <- unlist(a2) ;

a4 <- as.vector(as.numeric(a3))

a4 <- a4[!is.na(a4)]

Predicted\_Class <- a4[length(a4)]

Filename <- paste(substr(temp[k],1, nchar(temp[k])-4))

Out[k,1:1] <- Filename

Out[k,2:2] <- st1

Out[k,3:3] <- Actual\_Class

Out[k,4:4] <- Predicted\_Class

}

write.csv(Out, ("PAM\_manhattan\_GoF\_Probles8.csv"))

########################################################################

#####################################################################