title: "HW2" author: "Anindita Deb" date: "3/23/2022" output: pdf_document ######################Task1## ############ ##########################Visualizing the data ## data(MovieLense) head(MovieLense)

look at the first few ratings of the first user##

head(as(MovieLense[1,], "list")[[1]])

visualize part of the matrix######################

image(MovieLense[1:100,1:100])

number of ratings per user########################

hist(rowCounts(MovieLense))

number of ratings per movie#######################

hist(colCounts(MovieLense))

mean rating (averaged over users)#################

mean(rowMeans(MovieLense))

available user meta information###################

head(MovieLenseUser)

dim(getRatingMatrix(MovieLense)) getRatingMatrix(MovieLense)[1:10, 1:10]

######################Normalizing the matrix to get a better rating representation########################## MovieLense_Normalize <- normalize(MovieLense) getRatingMatrix(MovieLense_Normalize)[1:10, 1:10] ######Visualize raw ratings and normalized ratings###

x11() image(MovieLense_Normalize[1:100,1:100], main = "Normalized ratings")

x11() image(MovieLense_Normalize[1:100, 1:100], main = "Raw Ratings") MovieLense_denormalize <- denormalize(MovieLense_Normalize) ###################Creating a binary matrix ##

MovieLense_binarize <- binarize(MovieLense_denormalize, minRating = 4) getRatingMatrix(MovieLense_binarize)[1:10, 1:10] image(MovieLense_binarize[1:100,1:100], main = "Binarized ratings") ######################Histogram plot of normalized ratings##

number of normalized ratings per user#########################

x11() hist(rowCounts(MovieLense_Normalize),main="Count of normalized ratings per user")

number of normalized ratings per movie#########################

x11() hist(colCounts(MovieLense_Normalize),main="Count of normalized ratings per movie") ################Splitting data into train and test## dim(R) set.seed(1) MovieLense_sample <- sample(MovieLense,4000,replace=TRUE) dim(MovieLense_sample) eval<- evaluationScheme(MovieLense_sample,method = "cross-validation", k=10,given = -1,goodRating=4) eval

##############################Building a recommender model using user based collaborative filtering## recommender_user_based <- Recommender(getData(eval,"train"), "UBCF") names(getModel(recommender_user_based)) ############Predict the missing ratings###

recommender_user_based.predict<- predict(recommender_user_based, getData(eval, "known"), type="ratings") ERROR<- rbind(UBCF = calcPredictionAccuracy(recommender_user_based.predict, getData(eval,"unknown"))) ################## evaluating the top 1, 3, 5, 10,15,20 recommendation lists using user based collaborative filtering########## results<- evaluate(eval, method = "UBCF", type="topNList", n=c(1,3,5,10,15,20)) ########################Plotting the ROC curve## x11() plot(results,annotate=TRUE,main="ROC curve") graphics.off() x11() plot(results, "prec/rec", annotate=TRUE,main="Prec/Rec curve") graphics.off() ######################Create top 10 recommendations for 3 users## recom <- predict(recommender_user_based, MovieLense[800:802], n=10)

##################End of Task1
##

################### Task3
##
################Visualization through graphical and statistical summaries## sum(is.na(iris)) is.null(iris) library(corrplot) M<-cor(iris[,1:4]) corrplot(M) cor(iris[, 1:4]) g <- ggplot(iris, aes(x = Sepal.Length, y = Sepal.Width)) g <- g + geom_point(aes(shape = factor(iris$Species), colour = factor(iris$Species))) g <- g + ggtitle (" SepalLength Vs SepalWidth wrt Species") g <- g + stat_smooth(method= lm)

g <- ggplot(iris, aes(x = Petal.Length, y = Petal.Width)) g <- g + geom_point(aes(shape = factor(iris$Species), colour = factor(iris$Species))) g <- g + ggtitle (" PetalLength Vs PetalWidth wrt Species") g <- g + stat_smooth(method= lm)

g <- ggplot(iris, aes(x = Sepal.Width, y = Petal.Width)) g <- g + geom_point(aes(shape = factor(iris$Species), colour = factor(iris$Species))) g <- g + ggtitle (" SepalLength Vs PetalLength wrt Species") g <- g + stat_smooth(method= lm)
######################PCA## ###################### library(stats) library("multtest") library("fpc") library("cluster") library("bootcluster") library("fossil") library(ggpubr) library(factoextra) iris_modified<-iris[,1:4] iris.fit <- prcomp(iris_modified, center=TRUE, scale=TRUE) summary(iris.fit) summary(iris.fit)$importance[2, 1:2] ############Creating a plot using the first two principal components and coloring the iris species by class ############## PCAdata <- as.data.frame(iris.fit$x[, 1:2]) PCAdata <- cbind(PCAdata, iris$Species) colnames(PCAdata) <- c("PC1", "PC2", "Species") percentVar <- round(100 * summary(iris.fit)$importance[2, 1:2], 0) ggplot(PCAdata) + aes(PC1, PC2, color = Species, shape = Species) + geom_point(size = 2)+ xlab(paste0("PC1: ", percentVar[1], "% variance")) + ylab(paste0("PC2: ", percentVar[2],

"% variance")) + ggtitle("Principal component analysis (PCA)") + theme(aspect.ratio = 1) ###############Applying KMeans on the first two principal components and plotting the data with KMeans######################## ######################## And Plotting the clusters according to the question ## PCAdata <- as.data.frame(iris.fit$x[, 1:2]) kmeans.fit <- kmeans(PCAdata, centers = 3, nstart = 10) summary(kmeans.fit) table(kmeans.fit$cluster,iris$Species) PCAdata <- cbind(PCAdata, iris$Species,factor(kmeans.fit$cluster)) colnames(PCAdata) <- c("PC1", "PC2", "Species","KmeansCluster") ggscatter(PCAdata, x = "PC1", y = "PC2", color = "KmeansCluster", palette = "npg", ellipse = TRUE, ellipse.type = "convex", shape = "Species", size = 1.5, legend = "right", ggtheme = theme_bw(), xlab = paste0("PC1: ", percentVar[1], "% variance"), ylab = paste0("PC2: ", percentVar[2], "% variance")) + stat_mean(aes(color = KmeansCluster), size = 4)+ ggtitle("Kmeans clusters on PC1 and PC2") ##############Evaluating the performance of Kmeans with PCA on the basis of rand index and adjusted rand index################ true_label <- as.numeric(iris$Species) rand.index(true_label,kmeans.fit$cluster) adj.rand.index(true_label,kmeans.fit$cluster) ##################Calculating Gap Statistics## iris_mod<-iris[,1:2] gap_kmeans <- clusGap(iris_mod, kmeans, nstart = 20, K.max = 10, B = 100) plot(gap_kmeans, main = "Gap Statistic with Kmeans") fviz_nbclust(iris_mod, kmeans, method = "wss") silhouette_score <- function(k){ km <- kmeans(iris_mod, centers = k, nstart=20)

ss <- silhouette(km$cluster, dist(iris_mod)) mean(ss[, 3]) } k <- 2:10 avg_sil <- sapply(k, silhouette_score) plot(k, type='b', avg_sil, xlab='Number of clusters', ylab='Average Silhouette Scores', frame=FALSE) ##Task2## ######################## tem <- data.frame(state.x77) # Transform matrix into data frame st <- cbind(state.abb, tem, state.region) colnames(st)[1] <- "State" # Rename first column colnames(st)[10] <- "Region" ###################################Visualizations## ################

############Ploting State Wise Population## library(ggplot2) ggplot(st, aes(x = State, y = Population)) + geom_point(size = 3, color = "red") + geom_segment(aes(x = State, xend = State, y = 0, yend = Population)) + theme(axis.text.x = element_text(angle = 90, vjust = 0.5))+ ggtitle("State wise population")

########################Visualizing murder distribution## library(maps) library(tidyverse) st$region <- tolower(state.name) states <- map_data("state") map <- merge(states, st, by = "region", all.x = T) map <-

map[order(map$order),]
ggplot(map, aes(x = long, y = lat, group = group)) +
geom_polygon(aes(fill = Murder)) +
geom_path() + scale_fill_gradientn(colours = rev(heat.colors(10))) + coord_map() + labs(x = "Longitude", y = "Latitude") + guides(fill = guide_legend(title = "Murder Rate")) ##################################Visualizing the correlations between all the variables## sta<- st[, 2:9] library(ellipse) library(corrplot) corMatrix <- cor(as.matrix(sta)) # Calculate correlation matrix col <- colorRampPalette(c("red", "yellow", "blue")) # 3 colors to represent coefficients -1 to 1. corrplot.mixed(corMatrix, order = "AOE", lower = "number", lower.col = "black", number.cex = .8, upper = "ellipse", upper.col = col(10), diag = "u", tl.pos = "lt", tl.col = "black")

M<-cor(sta) corrplot(M) ####################Visualizing the life expectency## ggplot(st, aes(x = Region, y = Life.Exp, fill = Region)) + geom_violin(trim = FALSE) + geom_boxplot(width = 0.1) ##################################Visulizing the illiteracy distribution# ################region wise### ggplot(st, aes(x = Illiteracy, fill = Region)) + geom_density(alpha = 0.3) ###################################

library(dplyr)

group income into IncomeType first

st.income <- st %>% mutate(IncomeType = factor(ifelse(Income < 3500, "Under3500",
ifelse(Income < 4000 & Income >= 3500, "3500-4000", ifelse(Income < 4500 & Income >= 4000, "4000-4500", ifelse(Income < 5000 & Income >= 4500, "4500-5000", "Above5000")))))) ggplot(st.income, aes(x = Murder, y = Life.Exp)) + geom_point(aes(shape = IncomeType, color = Region, size = HS.Grad)) + geom_smooth(method = 'lm', formula = y ~ x)

stars(st, key.loc = c(13, 1.5), draw.segments = T)
####################We need to scale the data because in hierarchial clustering or SOMs we calculate ###################### ############### the euclidean distance ## st=remove_rownames(st) census_data_mod<-st census_data_mod$Population<-log(census_data_mod$Population) census_data_mod$Income<-log(census_data_mod$Income) census_data_mod$Illiteracy<-log(census_data_mod$Illiteracy) census_data_mod$Life.Exp<-

log(census_data_mod$Life.Exp) census_data_mod$Murder<-log(census_data_mod$Murder) census_data_mod$HS.Grad<-log(census_data_mod$HS.Grad) census_data_mod$Frost<-log(census_data_mod$Frost) census_data_mod$Area<-log(census_data_mod$Area) census_data_mod<-census_data_mod[,-c(1,10)]

############################Hierarachial clustering## #define linkage methods

set.seed(1) idx <- sample(c(1:length(census_data_mod[,1])), 50,replace=TRUE) census_data_sample <- census_data_mod[idx,]

d <- dist(census_data_sample)

hc <- hclust(d, method = "ave") plot(hc,hang = -1, labels = st$Region[idx]) plot(hc,hang = -1, labels = st$State[idx]) ct <- cutree(hc, k = 3) si <- silhouette(ct, dist = d) plot(si)

silhouette_score <- function(k){ ct <- cutree(hc, k = k) ss <- silhouette(ct, dist=d) mean(ss[, 3]) }

k <- 2:10 avg_sil <- sapply(k, silhouette_score) plot(k, type='b', avg_sil, xlab='Number of clusters', ylab='Average Silhouette Scores for different dendogram cuts on Hierarachial', frame=FALSE) ct <- cutree(hc, k = 2) si <- silhouette(ct, dist = d)

##Modelling with SOMS set.seed(1) census_data_mod<-st census_data_mod$Population<-scale(census_data_mod$Population) census_data_mod$Income<-scale(census_data_mod$Income) census_data_mod$Illiteracy<-scale(census_data_mod$Illiteracy) census_data_mod$Life.Exp<-scale(census_data_mod$Life.Exp) census_data_mod$Murder<-scale(census_data_mod$Murder) census_data_mod$HS.Grad<-scale(census_data_mod$HS.Grad) census_data_mod$Frost<-scale(census_data_mod$Frost) census_data_mod$Area<-scale(census_data_mod$Area) census_data_mod<-census_data_mod[,-c(1,10)] census_data_mod<-as.matrix(census_data_mod) som_grid <- somgrid(xdim = 5, ydim = 5, topo = "hexagonal") census_data_mod.som <- som(census_data_mod, grid = som_grid, rlen = 10000)

codes <- census_data_mod.som$codes[[1]]

plot(census_data_mod.som,main="Self Organizing Maps on Census data") census_data_mod.som$unit.classif plot(census_data_mod.som,type="changes",main="Self Organizing Maps on Census data") plot(census_data_mod.som,type="changes",main="Self Organizing Maps on Census data") plot(census_data_mod.som, type = "count")

```
plot(census_data_mod.som, type = "mapping") coolBlueHotRed <- function(n, alpha = 1){rainbow(n, end=4/6, alpha = alpha)[n:1]} x11() plot(census_data_mod.som, type = "dist.neighbours", palette.name = coolBlueHotRed)

for (i in 1:13){ x11() plot(census_data_mod.som, type = "property", property=codes[,i], main = colnames(codes)[i]) } rownames(codes)<- NULL d_som <- dist(codes) hc_som <- hclust(d_som,method='ave') plot(hc_som,hang = -1,labels = st$State)

silhouette_score <- function(k){ ct <- cutree(hc_som, k = k) ss <- silhouette(ct, dist=d_som) mean(ss[, 3]) }

k <- 2:10 avg_sil <- sapply(k, silhouette_score) plot(k, type='b', avg_sil, xlab='Number of clusters', ylab='Average Silhouette Scores for different dendogram cuts on Hierarachial', frame=FALSE) ct <- cutree(hc_som, k = 2) si <- silhouette(ct, dist = d_som) plot(si)

my_pal <- c("red", "green") my_bhcol <- my_pal[si]

graphics.off()

x11() plot(census_data_mod.som, type = "mapping", col = "black", bgcol = my_bhcol) add.cluster.boundaries(census_data_mod.som, si)
```