Question 1

Centering and Scaling : Removed 2 words *of* and *the* as there standard deviations are 0 and cause problem while centering and scaling.

a). The dataset contains 102 subjects and 4432 features. In this context,

- "subject" represents a story from the *New York Times*, from 1987 to 2007. There are 57 randomly selected stories about art and 45 stories about music.
- feature $\overrightarrow{x}$ represents words; plus an indicator wheather the story is one about or one about music.

b) Please refer to Quesion 1 part (b) in the `R` code section for the analysis described in the excerpt. **Figure** 1 represent the projection of the times stories on to the first two principal components (analog of Figure 15.6). After removing 2 variables and standardization, figure is quite different than the figure for original data. The total number of PC's are 102 and consistent with what I expect.

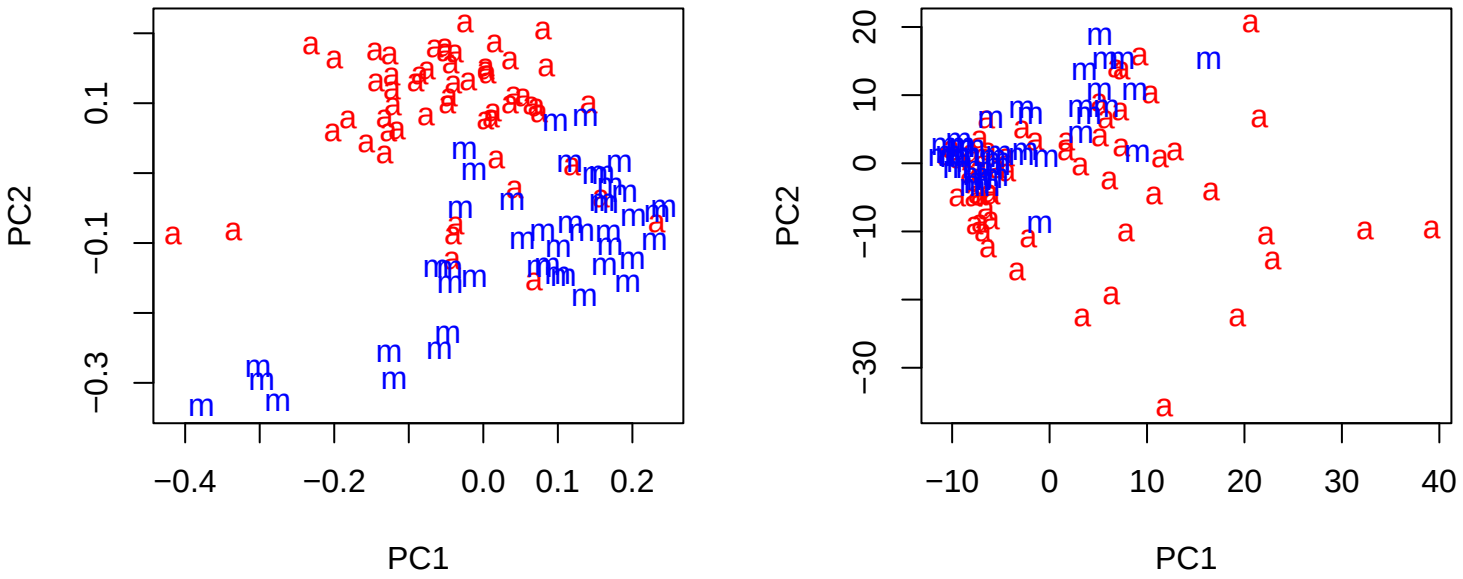

Figure 1: *Projection of the Times stories on to the first two principal components. Music stories are marked with a blue "m", art stories with a red "a". Left: Before standardizing. Right: After removing two variables and standardizing*

c) In **Figure** 1 there is a clear seperation between art and music classes. Therefore 2PCs is doing good job of predict whether the story is about art or music. However, the proportion of variance explained by PC1 and PC2 are 0.02094 and 0.01961 respectively.These two principle components accounted only for 4% of the total variance. Number of components to be used to explain 90 % of variability should be at least 87.

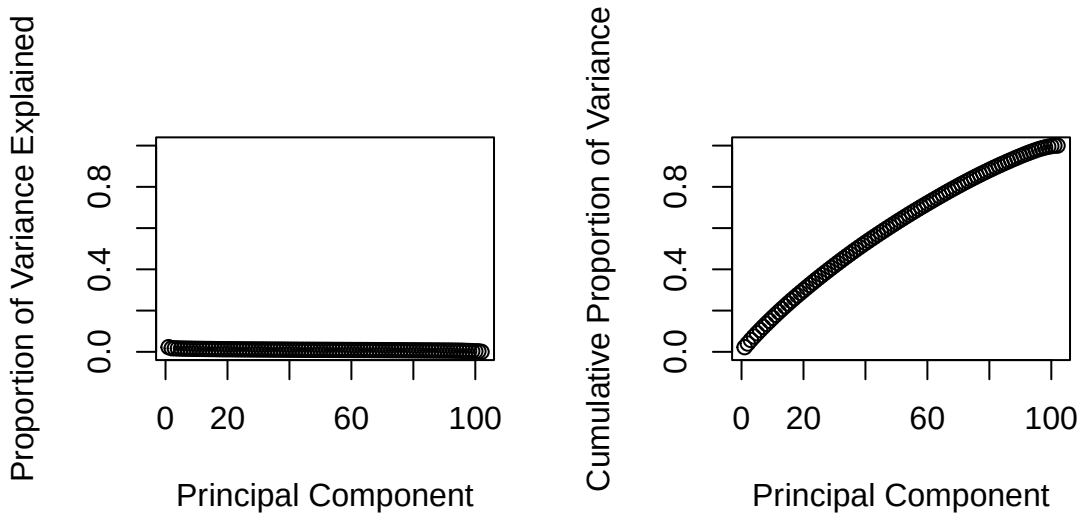

Figure 2: *Left: Variance explained by each PC. Right: Cumulative Proportion of Variance explained by PC's.*

d) Please refer Quesion 1 part (d) in the `R` code section for the fitted linear autoencoder with one hidden layer.

e) If we have an autoencoder with one hidden layer and two linearly activated nodes, the hidden layer activations will represent a linear combination of the original input variables. If we perform PCA on the same variables and retain the first two principal components, the scores of these components will also represent a linear combination of the original input variables.

Since both the hidden layer activations and the scores of the first two principal components represent linear combinations of the same original input variables, they should be identical,But **up to a scaling factor**. So they are not similar.

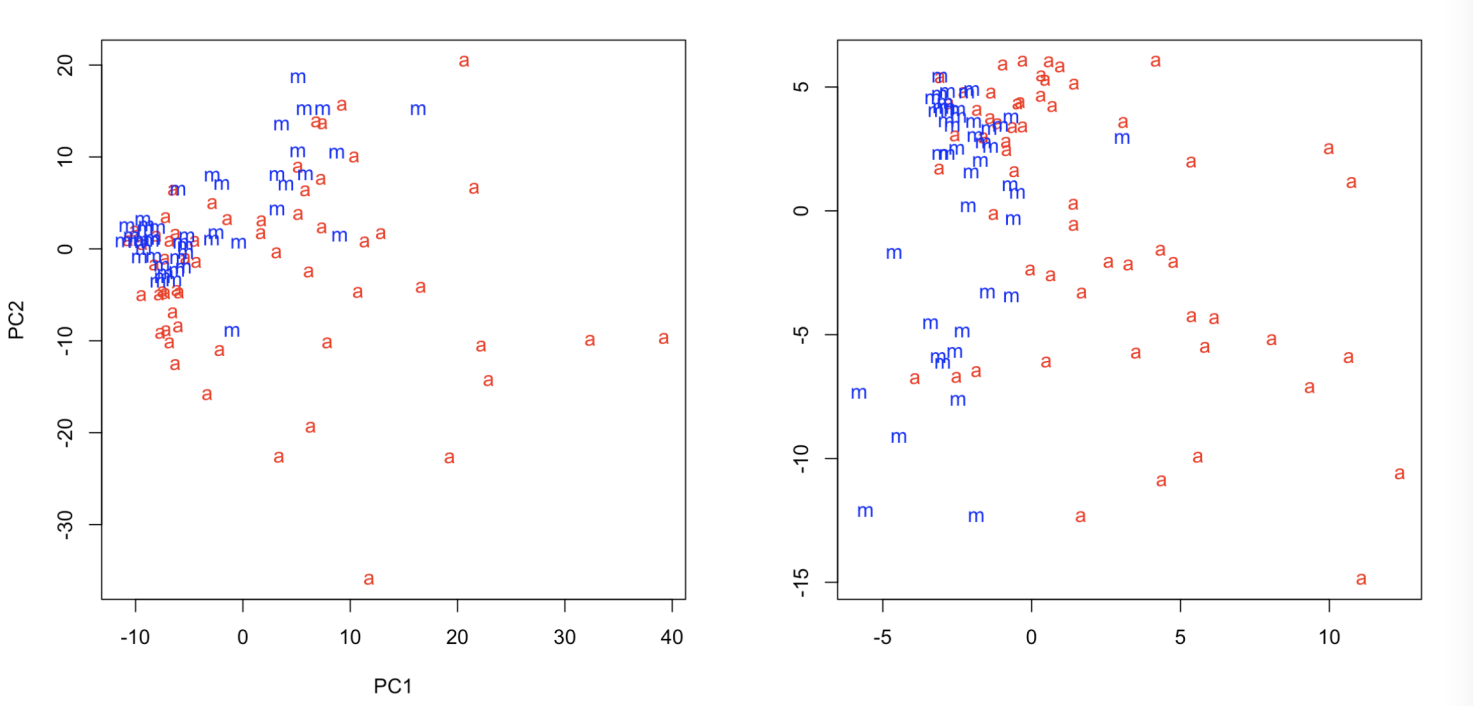

Figure 3: *Analog of Figure 15.6, Left: PC;s Middle: Autoencoder with linear activation Right: Autoencoder with ReLU*

f) Please refer Quesion 1 part (f) in the `R` code section for the fitted autoencoder with one hidden layer and ReLU activation. **Figure** shows the analog of Figure 15.6 using hidden layer activations on the two axes. ReLU activation is non linear activation and resultant plot is different than the one obtained using principal component analysis. It does not do a great job distinguishing art and music stories. This could be because ReLU may be prone to "dying" neurons, where a neuron becomes inactive and stops contributing to the model's output, which can negatively impact the model's performance.

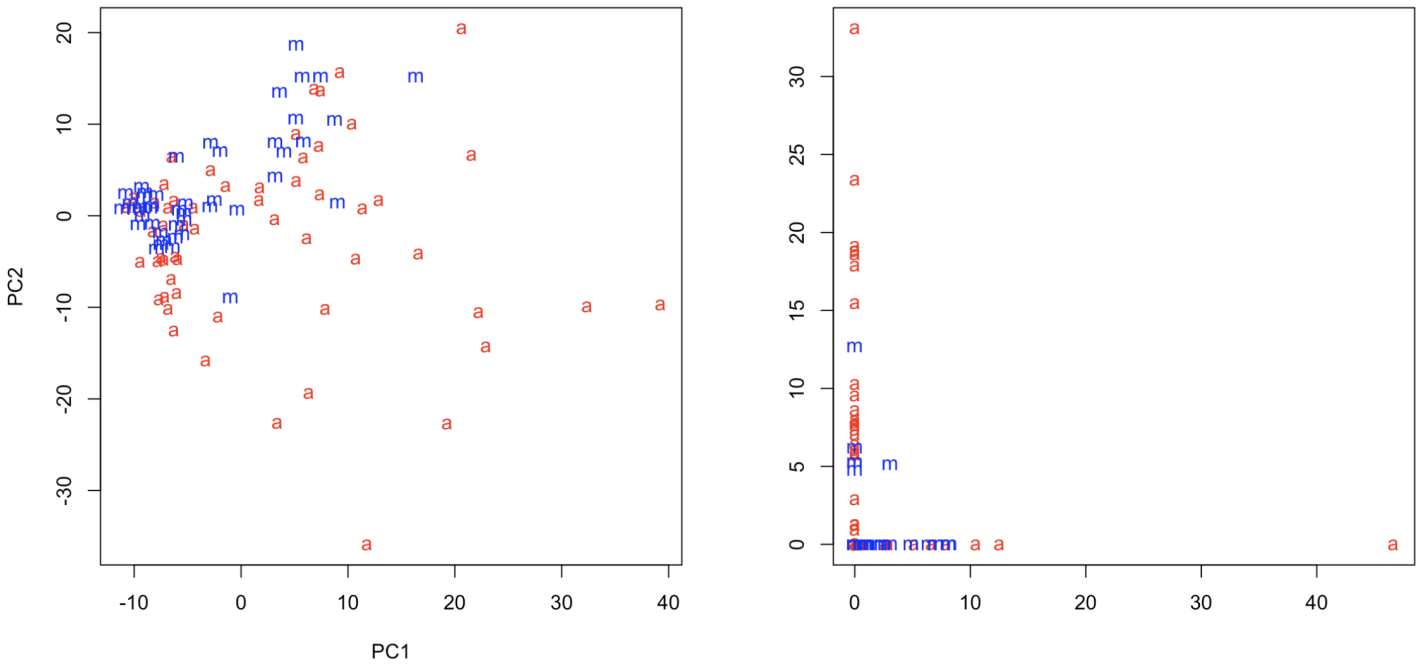

Figure 4: *Analog of Figure 15.6, Left: PC;s Middle: Autoencoder with linear activation Right: Autoencoder with ReLU*

g) According to the above figures principal components method separates two classes well compared to auto encoder with one hidden layer and ReLU activation. Therefore this particular data set it seems linear model do a better job when distinguish documents between art and music than non linear model.

Question 2

a) Dataset contains a set of ratings given to movies by a set of users, and is a workhorse of recommender system research. There are 943 rows(users) and 1664 columns(movies). This is in the format of dgCMatrix class. It is a formal class in the R package "Matrix", which represents a sparse matrix in compressed

- i: A numeric vector that contains the row indices of the non-zero elements of the matrix, in column-major order.
- j: A numeric vector that contains the column indices of the non-zero elements of the matrix, in column-major order.
- Dim: A numeric vector of length 2 that specifies the dimensions of the matrix.
- Dimnames: A list of length 2 that contains the names of the rows and columns of the matrix, respectively.
- x: A numeric vector that contains the ratings of the matrix, in column-major order.

Most of the entries are missing and treated as zero.

b) There are 565 users and 601 movies remain in the data.

c) Please refer Quesion 2 part (c) in the `R` code section.

d) See the code.Top 5 movie reccomendations for user1 are "McHale's Navy (1997)","Island of Dr. Moreau, The (1996)","Striptease (1996)","Excess Baggage (1997)","Beautician and the Beast, The (1997)".

e) We can find the optimal k by

- Trial and error method: Try different values for the number of latent factors and evaluate the performance.
- Cross Validation : Use cross-validation to evaluate the performance of the recommender system for different values of the number of latent factors.
- Grid Search
- Scree plot: plots the eigenvalues of the singular value decomposition (SVD) of the user-item matrix against the corresponding latent factor number

Question 3

a) Top US Arrest states by murder is Georgia, assault is North Carolina and rape is Nevada. There is no relationship between population with assault and murder. But it seems have a correlation between population and rape.

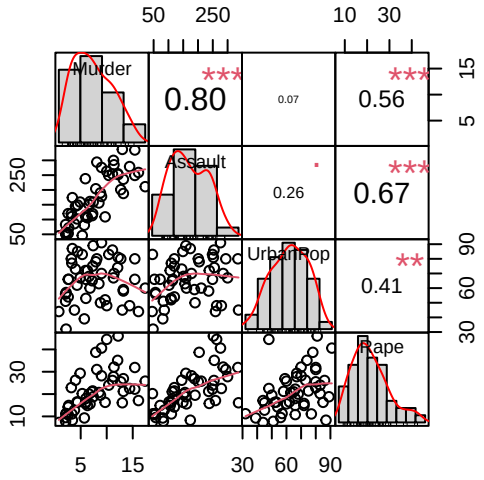

Figure 5: *Scatterplot matrix for USArrest data*

b) Below represent eigen values and eigen vectors obtain from eigen value decomposition and loading matrix obtained using PCA. PC loading are the eigen vectors obtained by eigen value decomposition of covariance matrix. This is consistent with what we learnt in the class.

```
eigen() decomposition
$values
[1] 2.4802416 0.9897652 0.3565632 0.1734301

$vectors
           [,1]       [,2]       [,3]        [,4]
[1,] -0.5358995  0.4181809 -0.3412327  0.64922780
[2,] -0.5831836  0.1879856 -0.2681484 -0.74340748
[3,] -0.2781909 -0.8728062 -0.3780158  0.13387773
[4,] -0.5434321 -0.1673186  0.8177779  0.08902432

[1] "Principal Component Analysis"

Standard deviations (1, .., p=4):
[1] 1.5748783 0.9948694 0.5971291 0.4164494

Rotation (n x k) = (4 x 4):
                PC1        PC2        PC3         PC4
Murder   -0.5358995  0.4181809 -0.3412327  0.64922780
Assault  -0.5831836  0.1879856 -0.2681484 -0.74340748
UrbanPop -0.2781909 -0.8728062 -0.3780158  0.13387773
Rape     -0.5434321 -0.1673186  0.8177779  0.08902432
```

Below represent the covariance matrix of scores. The covariance matrix of scores is a diagonal matrix with eigenvalues on the diagonal

```
         PC1       PC2       PC3       PC4
PC1 2.480242 0.0000000 0.0000000 0.0000000
PC2 0.000000 0.9897652 0.0000000 0.0000000
PC3 0.000000 0.0000000 0.3565632 0.0000000
PC4 0.000000 0.0000000 0.0000000 0.1734301
```

c) Singular values obtained by svd are 11.024148, 6.964086, 4.179904, 2.915146. These singular values are equal to the $\sqrt{(n-1) * \text{eigen values}}$. V matrix contains eigen vectors obtained above. This is what we expected in the class.

```
[1] "Singular values using svd"

[1] 11.024148  6.964086  4.179904  2.915146

[1] "Singular values using eigen values"

[1] 11.024148  6.964086  4.179904  2.915146
```

d) PCA- and SVD-based factorizations of D are identical for any k = 1, 2, 3, 4. Below output shows 1st row of approximated D matrix using both svd and pca for all k values.

```
[1] "When k=1"

[1] "PCA"

        Murder   Assault  UrbanPop      Rape
[1,] 0.5228559 0.5689892 0.2714198 0.5302052
[2,] 1.0345742 1.1258581 0.5370580 1.0491162

[1] "SVD"

          [,1]      [,2]      [,3]      [,4]
[1,] 0.5228559 0.5689892 0.2714198 0.5302052
[2,] 1.0345742 1.1258581 0.5370580 1.0491162

[1] "When k=2"

[1] "PCA"

        Murder   Assault   UrbanPop      Rape
[1,] 0.9920554 0.7799093 -0.7078698 0.3424735
[2,] 1.4788608 1.3255791 -0.3902348 0.8713524

[1] "SVD"

          [,1]      [,2]       [,3]      [,4]
[1,] 0.9920554 0.7799093 -0.7078698 0.3424735
[2,] 1.4788608 1.3255791 -0.3902348 0.8713524

[1] "When k=3"

[1] "PCA"

        Murder   Assault  UrbanPop        Rape
[1,] 1.1421308 0.8978419 -0.541617 -0.01718823
[2,] 0.7897413 0.7840532 -1.153638  2.52285512

[1] "SVD"

          [,1]      [,2]      [,3]        [,4]
[1,] 1.1421308 0.8978419 -0.541617 -0.01718823
[2,] 0.7897413 0.7840532 -1.153638  2.52285512

[1] "When k=4"

[1] "PCA"

        Murder   Assault   UrbanPop         Rape
[1,] 1.2425641 0.7828393 -0.5209066 -0.003416473
[2,] 0.5078625 1.1068225 -1.2117642  2.484202941

[1] "SVD"

          [,1]      [,2]       [,3]         [,4]
[1,] 1.2425641 0.7828393 -0.5209066 -0.003416473
[2,] 0.5078625 1.1068225 -1.2117642  2.484202941
```

Section 2: R code

```
## ----setup, include=FALSE-----------------------------------------
knitr::opts_chunk$set(echo = TRUE)
options(xtable.comment = FALSE)
knitr::opts_chunk$set(dev = 'pdf')


## ----echo=FALSE---------------------------------------------------
library(knitr)
opts_chunk$set(comment="",warning = FALSE, message=FALSE,tidy.opts=list(keep.blank.line=TRUE, width.cutoff=120),

##############################################
#                  Question 1
##############################################

## ----include=FALSE------------------------------------------------

nyt<-read.csv("/Users/nissi_wicky/Documents/UTD/11) 2023 Spring/STAT 6390 Deep Learning/Deep Learning Mini Proje


## ----include=FALSE------------------------------------------------
############ Part b
col1<-match("of",names(nyt))
col2<-match("the",names(nyt))

nyt.frame <- nyt[,-c(1,col1,col2)]

#column wise normalization
nyt.std <- apply(nyt[,-c(1,col1,col2)], 2, function(x) (x-mean(x))/sd(x))

#Remove 2 features
#nyt.frame<-nyt[,-c(1,2665,3971)]

#Standardize
#nyt.std = apply(nyt.frame, 2, function(x) x-mean(x)/sd(x))

#standardize and perform principal component analysis
nyt.pca <- prcomp(nyt.frame,center = TRUE, scale. = TRUE)

#perform principal component analysis without standardizing and removing variables
nyt.pca.without <- prcomp(nyt[,-1])

#loading matrix
nyt.latent.sem <- nyt.pca$rotation

signif(sort(nyt.latent.sem[, 1], decreasing = TRUE)[1:30], 2)
signif(sort(nyt.latent.sem[, 1], decreasing = FALSE)[1:30], 2)
signif(sort(nyt.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
signif(sort(nyt.latent.sem[, 2], decreasing = FALSE)[1:30], 2)


## ----q1b, echo=FALSE, fig.cap="\\textit{Projection of the Times stories on to the first two principal componer

par(mfrow=c(1,2))
plot(nyt.pca.without$x[, 1:2], pch = ifelse(nyt[, "class.labels"] == "music",
"m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
"red"))
plot(nyt.pca$x[, 1:2], pch = ifelse(nyt[, "class.labels"] == "music",
"m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
```

```
"red"))


## ----include=FALSE-----------------------------------------------
############### Part c

# Compute the proportion of variance explained (PVE)
summary(nyt.pca)
pc.var <- nyt.pca$sdev^2
pve <- pc.var/sum(pc.var)


## ----q1c, echo=FALSE, fig.cap="\\textit{Left: Variance explained by each PC. Right: Cumulative Proportion of V

par(mfrow=c(1,2))
plot(pve, xlab = "Principal Component", ylab = "Proportion of Variance Explained", ylim = c(0,1), type = 'b')
plot(cumsum(pve), xlab = "Principal Component", ylab = "Cumulative Proportion of Variance", ylim = c(0,1), type


## ----eval=FALSE, include=FALSE-----------------------------------
## xtrain = nyt.std
## input_size = dim(xtrain)[2]
## latent_size = 2
##
## #encoder
## encoder_model <- keras_model_sequential()
## encoder_model %>%
##   layer_dense(units = latent_size, input_shape = input_size) %>%
##   layer_dense(units = latent_size)
##
## summary(encoder_model)
##
## #decoder
## decoder_model <- keras_model_sequential()
## decoder_model %>%
##   layer_dense(units = input_size, input_shape = latent_size)
##
## summary(decoder_model)
##
## #autoencoder
## aen_input = layer_input(shape = input_size)
## aen_output = aen_input %>%
##   encoder_model() %>%
##   decoder_model()
##
## autoencoder_model = keras_model(aen_input, aen_output)
## summary(autoencoder_model)
##
## # Compile the model
## autoencoder_model %>% compile(
##   optimizer = "adam",
##   loss = "mean_squared_error"
## )
##
## # fit
## history <- autoencoder_model %>% fit(
##   xtrain, xtrain,
##   epochs = 500,
##   batch_size = 32
```

```
## )
##
## #encoder_1 <- keras_model(input_layer, encoder_layer)
## encoder_score <- predict(encoder_model, xtrain)


## ----auto1, eval=FALSE, fig.cap="\\textit{Left: s.}", fig.height=3, fig.width=6, include=FALSE----
##
## par(mfrow=c(1,2))
##
## plot(nyt.pca$x[,1:2], pch = ifelse(nyt[, "class.labels"] == "music",
##                                    "m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
##                                                            "red"))
##
## plot(encoder_score, pch = ifelse(nyt[, "class.labels"] == "music",
##                                    "m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
##                                                            "red"),xlab=" ",ylab=" ")


## ----eval=FALSE, include=FALSE-------------------------------------
## xtrain = nyt.std
## input_size = dim(xtrain)[2]
## latent_size = 2
##
## #encoder
## encoder_model <- keras_model_sequential()
## encoder_model %>%
##   layer_dense(units = latent_size, input_shape = input_size) %>%
##   layer_dense(units = latent_size,activation = "relu")
##
## summary(encoder_model)
##
## #decoder
## decoder_model <- keras_model_sequential()
## decoder_model %>%
##   layer_dense(units = input_size, input_shape = latent_size)
##
## summary(decoder_model)
##
## #autoencoder
## aen_input = layer_input(shape = input_size)
## aen_output = aen_input %>%
##   encoder_model() %>%
##   decoder_model()
##
## autoencoder_model = keras_model(aen_input, aen_output)
## summary(autoencoder_model)
##
## # Compile the model
## autoencoder_model %>% compile(
##   optimizer = "adam",
##   loss = "mean_squared_error"
## )
##
## # fit
## history <- autoencoder_model %>% fit(
##   xtrain, xtrain,
##   epochs = 500,
##   batch_size = 32
```

```
## )
##
## #encoder_1 <- keras_model(input_layer, encoder_layer)
## encoder_score2 <- predict(encoder_model, xtrain)


## ----auto2, eval=FALSE, fig.cap="\\textit{Left: s.}", fig.height=3, fig.width=6, include=FALSE----
##
## par(mfrow=c(1,2))
##
## plot(nyt.pca$x[,1:2], pch = ifelse(nyt[, "class.labels"] == "music",
##                                    "m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
##                                                            "red"))
##
## plot(encoder_score2, pch = ifelse(nyt[, "class.labels"] == "music",
##                                    "m", "a"), col = ifelse(nyt[, "class.labels"] == "music", "blue",
##                                                            "red"),xlab=" ", ylab=" ")


################################################
#                  Question 2
################################################

## ----eval=FALSE, include=FALSE-------------------------------------
## library(recommenderlab)
## data("MovieLense")
## Movie<-MovieLense@data
## str(Movie)


## ----eval=FALSE, include=FALSE-------------------------------------
## data("MovieLense")
## Movie<-MovieLense@data
##
## # Forming sub data set
##
## # Selecting columns with at least 50 nonzero entries
## col_50<-c()
## for (j in 1:ncol(Movie)) {
##   a<-sum(Movie[,j]!=0)
##   if(a>=50){
##     col_50<-c(col_50,j)
##   }else{
##     col_50=col_50
##   }
## }
## # col_50 = 601
##
## # Selecting rows with at least 50 nonzero entries
## row_50<-c()
## for (i in 1:nrow(Movie)) {
##   b<-sum(Movie[i,]!=0) # only using above selected movies (or columns)
##   if(b>=50){
##     row_50<-c(row_50,i)
##   }else{
##     row_50=row_50
##   }
## }
##
```

```
## # row_50 = 565
##
## # Sub dataset
## sub_Movie<-Movie[row_50,col_50]
## #subsusb<-sub_Movie[c(1:50),c(1:75)]
## D<-as.matrix(sub_Movie)


## ----eval=FALSE, include=FALSE-------------------------------------
## rec_sys <- function(X,  hidden_lyr , l_rate, niter)
## {
##   err_it =0
##   input_lyr = dim(X)[1]
##   output_lyr = dim(X)[2]
##   I = diag(input_lyr)
##   Xhat = matrix(rep(0, ), ncol = output_lyr)
##   U = matrix(rep(0.1,(input_lyr*hidden_lyr)),nrow = input_lyr)
##   V = matrix(rep(0.1,(output_lyr*hidden_lyr)),nrow = output_lyr)
##
##   #epochs
##   for(k in 1:niter)
##   {
##
##     for(i in 1:input_lyr)
##     {
##       #forward propergation
##       H = I[i,]%*%U # I[1,] is 1 by 3
##       Xhat = H%*%t(V)
##
##
##       err= matrix(rep(0), nrow = 1, ncol = output_lyr) #4 by 1 vector
##       for(j in 1:output_lyr)
##       {
##         if(X[i,j]>0){
##           err[1,j] = X[i,j] - Xhat[1,j]
##         }
##       }
##       #backward propergation
##
##       # calculate gradients
##       dV = -2 * t(err)%*%U[i,]
##       V = V - l_rate*dV
##
##       dU = -2 * err%*%V
##       U[i,] = U[i,] - l_rate*dU
##     }
##     Xtemp = U%*%t(V)
##     obs_it = ifelse(X > 0, 1, 0)
##     err_it[k]= sum((X-obs_it*Xtemp)^2)
##   }
##
##   Xnw = U%*%t(V)
##   View(Xnw)
##   obs = ifelse(X > 0, 1, 0)
##   final_err= sum((X-obs*Xnw)^2)
##   final_err
##  return(list(err= final_err, plt= err_it, Xnw=Xnw))
## }
##
```

```
## X = matrix(c(1,0,0,4,0,0,2,3,0,5,0,0),nrow = 3,ncol = 4,byrow = T)
## R = rec_sys( X,  hidden_lyr = 2, l_rate=0.001, niter =200)
## #Works well with this data set


## ----eval=FALSE, include=FALSE-----------------------------------
## X<-as.matrix(sub_Movie)
## R1 = rec_sys( X,  hidden_lyr = 5, l_rate=0.0001, niter =500)
## rw1=R1$Xnw[1,]
## m1=order(rw1)[1:5]
## colnames(X)[m1]


## ----eval=FALSE, include=FALSE-----------------------------------
## X<-as.matrix(sub_Movie)
## R2 = rec_sys( X,  hidden_lyr = 10, l_rate=0.0001, niter =500)
## rw2=R2$Xnw[1,]
## m2=order(rw1)[1:5]
## colnames(X)[m2]

##############################################
#                 Question 3
##############################################

## ----q3a, echo=FALSE, fig.cap="\\textit{Scatterplot matrix for USArrest data}",fig.height=3, fig.width=3----
############ Part a
arrests <- USArrests

library(PerformanceAnalytics)
chart.Correlation(arrests)


## ----echo=FALSE--------------------------------------------------
######## Part b

D <- as.matrix(apply(arrests, 2, function(x) (x-mean(x))/sd(x)))
S = cov(D)
arrest.pca <- prcomp(D)
arrest.eigen <- eigen(S)

arrest.eigen
print("Principal Component Analysis")
arrest.pca


## ----echo=FALSE--------------------------------------------------
#covariance matrix of scores
round(cov(arrest.pca$x), 7)


## ----echo=FALSE--------------------------------------------------
arrest.svd <- svd(D)
print("Singular values using svd")
arrest.svd$d


## ----echo=FALSE--------------------------------------------------
print("Singular values using eigen values")
sqrt(49*arrest.eigen$values)
```

```
## ----echo=FALSE-----------------------------------------------------
lambda<-arrest.svd$d
SU <-arrest.svd$u
SV <-arrest.svd$v

PU = arrest.pca$x
PV = arrest.pca$rotation

#when k = 1
svd.sub1 <- lambda[1]*SU[,1]%*%t(SV[,1])
pca.sub1 <- PU[,1]%*%t(PV[,1])

print("When k=1")
print("PCA")
print(pca.sub1[1:2,])
print("SVD")
print(svd.sub1[1:2,])

#when k = 2
svd.sub2 <- lambda[1]*SU[,1]%*%t(SV[,1]) + lambda[2]*SU[,2]%*%t(SV[,2])
pca.sub2 <- PU[,1]%*%t(PV[,1]) + PU[,2]%*%t(PV[,2])

print("When k=2")
print("PCA")
print(pca.sub2[1:2,])
print("SVD")
print(svd.sub2[1:2,])

#when k=3
svd.sub3 <- lambda[1]*SU[,1]%*%t(SV[,1]) + lambda[2]*SU[,2]%*%t(SV[,2]) + lambda[3]*SU[,3]%*%t(SV[,3])
pca.sub3 <- PU[,1]%*%t(PV[,1]) + PU[,2]%*%t(PV[,2]) + PU[,3]%*%t(PV[,3])

print("When k=3")
print("PCA")
print(pca.sub3[1:2,])
print("SVD")
print(svd.sub3[1:2,])

#when k = 4
svd.sub4 <- lambda[1]*SU[,1]%*%t(SV[,1]) + lambda[2]*SU[,2]%*%t(SV[,2]) + lambda[3]*SU[,3]%*%t(SV[,3]) + lambda[
pca.sub4 <- PU[,1]%*%t(PV[,1]) + PU[,2]%*%t(PV[,2]) + PU[,3]%*%t(PV[,3]) + PU[,4]%*%t(PV[,4])

print("When k=4")
print("PCA")
print(pca.sub4[1:2,])
print("SVD")
print(svd.sub4[1:2,])
```