A REPORT
ON

‘Supervised Dimensionality Reduction Techniques’

BY

Name of the Student	ID Number
Soorej S Nair	***2017A7PS0113P***
Viniti Jain	***2018A7PS0202P***
Parinistha Dev Das	***2018A1PS0200P***

Prepared on completion of the
Assignment1 Course No. BITS F464

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Supervised learning contains data and labels. A supervised model tries to learn what patterns in the data lead to the labels. The supervised part of machine learning happens during training.

1. Fischer’s Linear Discriminant Analysis

FLD selects a projection that maximizes the class separation. To do that, it maximizes the ratio between the between-class variance to the within-class variance.
In short, to project the data to a smaller dimension and to avoid class overlapping, FLD maintains 2 properties.

- A large variance among the dataset classes.
- A small variance within each of the dataset classes.

#INFORMATION ABOUT LETTER RECOGNITION DATASET

Fischer’s Linear Discriminant Analysis has been performed on the Letter Recognition dataset.
It is a multiclass problem with 26 classes.There are 20,000 instances and 17 attributes.
Attribute Information:

1.	letter	capital letter	(26 values from A to Z)
2.	x-box	horizontal position of box	(integer)
3.	y-box	vertical position of box	(integer)
4.	width	width of box	(integer)
5.	high	height of box	(integer)
6.	onpix	total # on pixels	(integer)
7.	x-bar	mean x of on pixels in box	(integer)
8.	y-bar	mean y of on pixels in box	(integer)
9.	x2bar	mean x variance	(integer)
10.	y2bar	mean y variance	(integer)
11.	xybar	mean x y correlation	(integer)
12.	x2ybr	mean of x * x * y	(integer)
13.	xy2br	mean of x * y * y	(integer)
14.	x-ege	mean edge count left to right	(integer)
15.	xegvy	correlation of x-ege with y	(integer)
16.	y-ege	mean edge count bottom to top	(integer)
17.	yegvx	correlation of y-ege with x	(integer)

The objective is to identify each of a large number of black-and-white rectangular pixel displays as one of the 26 capital letters in the English alphabet. The character images were based on 20 different fonts and each letter within these 20 fonts was randomly distorted to produce a file of 20,000 unique stimuli. Each stimulus was converted into 16 primitive numerical attributes (statistical moments and edge counts) which were then scaled to fit into a range of integer values from 0 through 15. We

typically train on the first 15000 items and then use the resulting model
to predict the letter category for the remaining 5000.

#INFORMATION ABOUT BANK NOTE AUTHENTICATION DATASET

FLD ,PCA and Metric Learning was also performed on bank note authentication dataset for the purpose of comparison:
Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images.
Attribute Information:
1. variance of Wavelet Transformed image (continuous)
2. skewness of Wavelet Transformed image (continuous)
3. curtosis of Wavelet Transformed image (continuous)
4. entropy of image (continuous)
5. class (integer)

i)LFDA

Local Fisher discriminant analysis is a localized variant of Fisher discriminant analysis and it is popular for supervised dimensionality reduction method. lfda is an R package for performing local Fisher discriminant analysis.It does local-preserving projection in a way that between-class separability is maximized while within-class separability is minimized and its local structure is preserved.
We combine the concepts for both FDA (fishers discriminant analysis)and LPP(locality preserving projection) to define local Fisher discriminant analysis.

$$S^{(w)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{i,j}^{(w)}(x_i - x_j)(x_i - x_j)^T,$$

$$S^{(b)} = \frac{1}{2}\sum_{i,j=1}^{n} W_{i,j}^{(b)}(x_i - x_j)(x_i - x_j)^T,$$

where

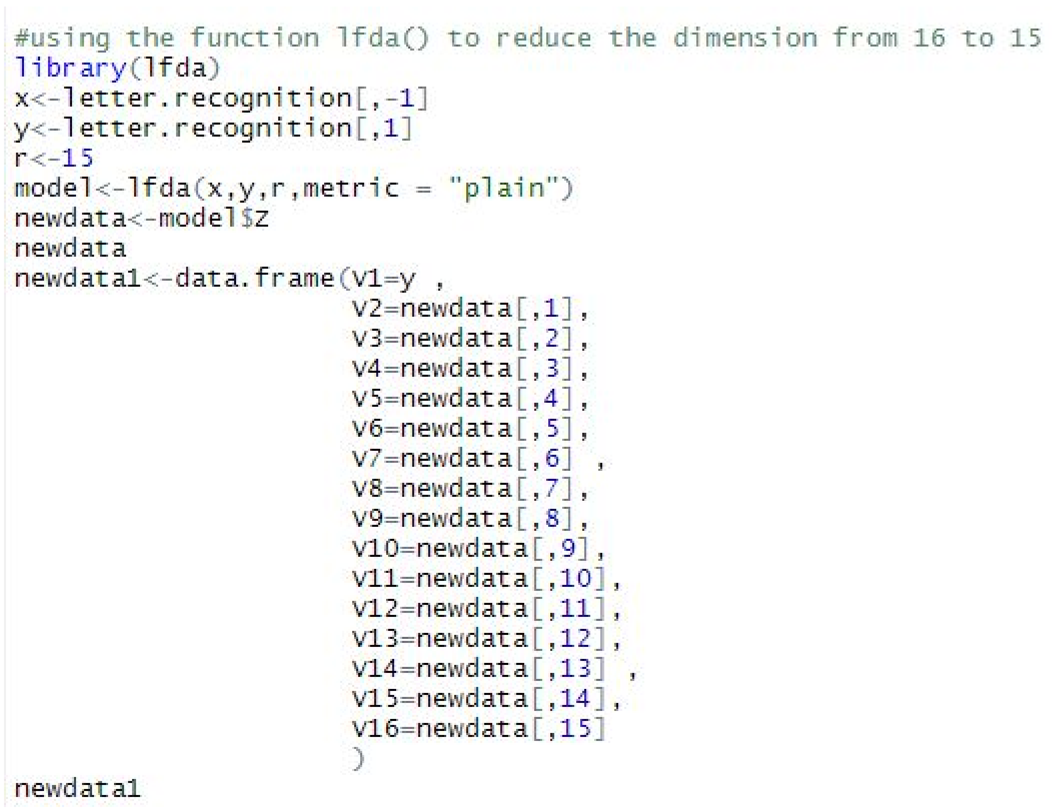
$$W_{i,j}^{(w)} = \begin{cases} A_{i,j}/n_l & \text{if } y_i = y_j = l, \\ 0 & \text{if } y_i \neq y_j, \end{cases}$$

$$W_{i,j}^{(b)} = \begin{cases} A_{i,j}(1/n - 1/n_l) & \text{if } y_i = y_j = l, \\ 1/n & \text{if } y_i \neq y_j. \end{cases}$$

The LFDA transformation matrix TLFDA is then defined as the following:

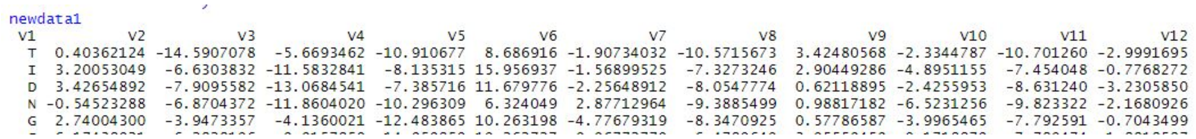
$$T_{LFDA} = argmax_{T \in \mathbb{R}^{d \times r}} \left[tr((T^T S^{(w)} t)^{-1} T^T S^{(b)} T) \right]$$

The transformation matrix TLFDA is found so that data pairs that are near each other in the same class become closer and data pairs in different classes become distant from each other. Note that data pairs in the same class that are far away from each other do not come closer.
In the data set letter_recognition.data lfda is used.
Lfda is a function in the library lfda.

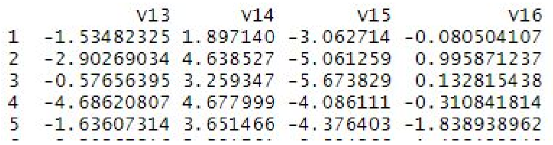


```
#using the function lfda() to reduce the dimension from 16 to 15
library(lfda)
x<-letter.recognition[,-1]
y<-letter.recognition[,1]
r<-15
model<-lfda(x,y,r,metric = "plain")
newdata<-model$z
newdata
newdata1<-data.frame(V1=y ,
                     V2=newdata[,1],
                     V3=newdata[,2],
                     V4=newdata[,3],
                     V5=newdata[,4],
                     V6=newdata[,5],
                     V7=newdata[,6] ,
                     V8=newdata[,7],
                     V9=newdata[,8],
                     V10=newdata[,9],
                     V11=newdata[,10],
                     V12=newdata[,11],
                     V13=newdata[,12],
                     V14=newdata[,13] ,
                     V15=newdata[,14],
                     V16=newdata[,15]
                     )
newdata1
```

The available types of metric are weighted, orthonormalized, and plain
Plain is used here.
Newdata1 now has 15 dimensions excluding output :



```
newdata1
 V1          V2          V3          V4         V5        V6          V7          V8         V9        V10        V11        V12
  T  0.40362124 -14.5907078  -5.6693462 -10.910677  8.686916 -1.90734032 -10.5715673 3.42480568 -2.3344787 -10.701260 -2.9991695
  I  3.20053049  -6.6303832 -11.5832841  -8.135315 15.956937 -1.56899525  -7.3273246 2.90449286 -4.8951155  -7.454048 -0.7768272
  D  3.42654892  -7.9095582 -13.0684541  -7.385716 11.679776 -2.25648912  -8.0547774 0.62118895 -2.4255953  -8.631240 -3.2305850
  N -0.54523288  -6.8704372 -11.8604020 -10.296309  6.324049  2.87712964  -9.3885499 0.98817182 -6.5231256  -9.823322 -2.1680926
  G  2.74004300  -3.9473357  -4.1360021 -12.483865 10.263198 -4.77679319  -8.3470925 0.57786587 -3.9965465  -7.792591 -0.7043499
```



```
          V13      V14       V15          V16
1  -1.53482325 1.897140 -3.062714 -0.080504107
2  -2.90269034 4.638527 -5.061259  0.995871237
3  -0.57656395 3.259347 -5.673829  0.132815438
4  -4.68620807 4.677999 -4.086111 -0.310841814
5  -1.63607314 3.651466 -4.376403 -1.838938962
```

Newdata1 is then split into training and test sets 75-25% respectively
And then lda is used to classify and find the accuracy of the model.

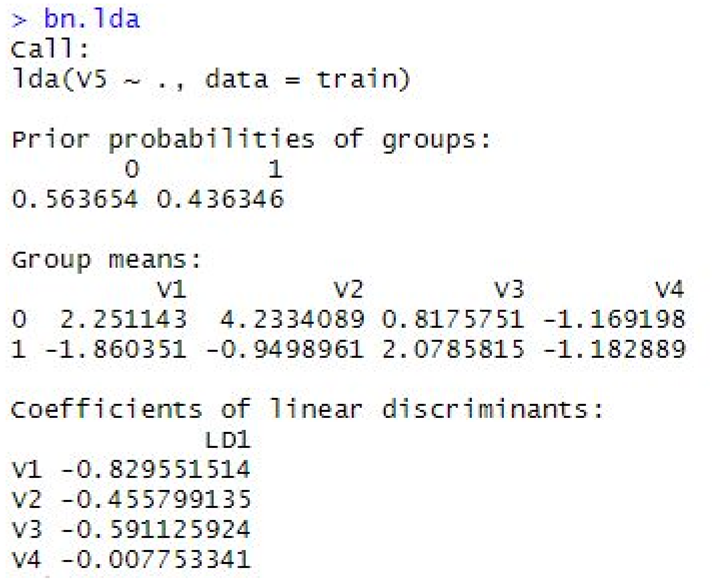
ii) LDA

One of the most commonly used supervised dimensionality reduction methods is linear discriminant analysis(LDA), It is designed to find low-dimensional projection that maximizes class separation,it uses it to perform classification.
R provides a library “MASS” which contains the function lda(formula,data….).The function tries to detect if the within-class covariance matrix is singular. If any variable has within-group variance less than tol^2 it will stop and report the variable as constant ; where tol is

a tolerance to decide if a matrix is singular.Specifying the prior(the prior probabilities of class membership.) will affect the classification : if unspecified, the class proportions for the training set are used. If specified, the probabilities should be in the order of the factor levels.
For each individual, lda determines group means and computes the probability of belonging to the different groups. The individual is then put to the group with the highest probability.
To measure the capacity of separation for each new feature of space candidate . In 1988, a statistician called Ronald Fisher proposed to maximize the function that represents the difference between the means, normalized by a measure of the within-class variability.hence they came up with two measures: the within class and the between class. However, this formulation is only possible if Normal distribution of the dataset is assumed.

a)LDA as a dimensionality reduction algorithm :

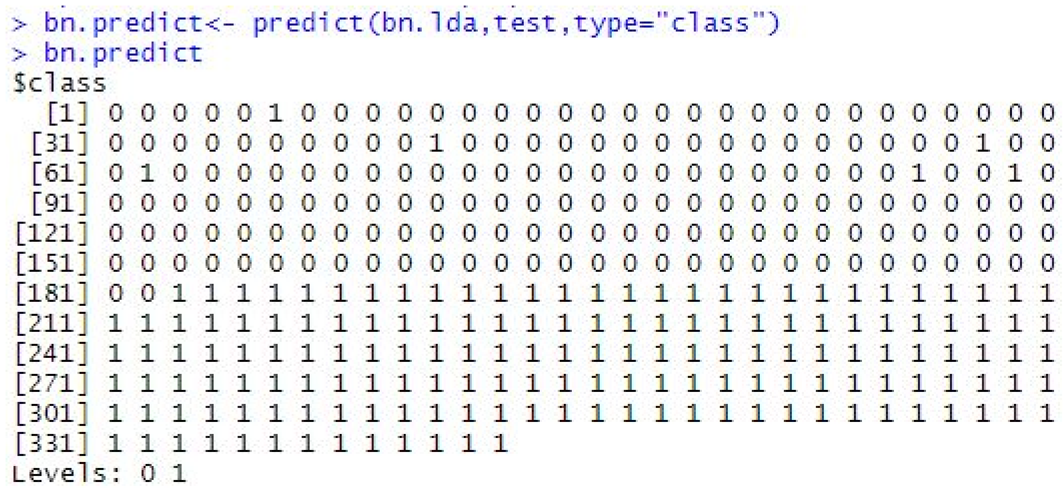
LDA reduces dimensionality from original number of attributes to C — 1 features, where C is the number of classes.The group means and the coefficients of linear discriminant describe the new feature space where the data will be projected in.
for example if one has 3 classes the new feature space will contain 2 attributes.
In the dataset data_bank_authentication LDA is used for both classification and dimensionality reduction.
bn.lda<-lda(V5 ~.,data=train)
bn.lda
This is the lda function in which V5 is the attribute which specifies the class in which the data is classified.and the data used for lda classification is train(which is 75% of the actual dataset used for supervised learning).the next line bn.lda gives this as output



```
> bn.lda
Call:
lda(V5 ~ ., data = train)

Prior probabilities of groups:
       0        1 
0.563654 0.436346 

Group means:
         V1         V2        V3        V4
0  2.251143  4.2334089 0.8175751 -1.169198
1 -1.860351 -0.9498961 2.0785815 -1.182889

Coefficients of linear discriminants:
            LD1
V1 -0.829551514
V2 -0.455799135
V3 -0.591125924
V4 -0.007753341
```

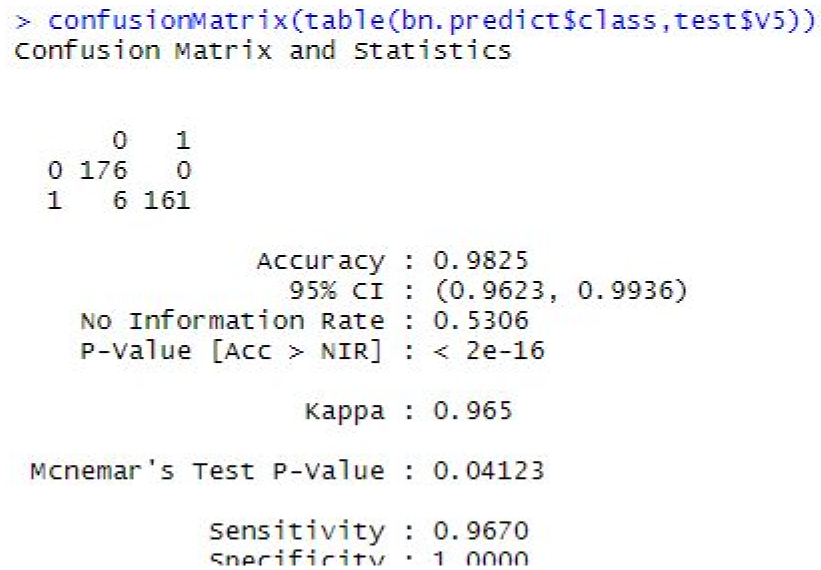
LD1 is only one discriminant function (i.e C -1)=(2-1)=1) this reduces the number of attributes to 1 from 4.Then the predict() function is used to predict the result from the model(bn.lda).it takes model ,data,and the type as input respectively.Test is used as data and the type is class because classification is used.

bn.predict<- predict(bn.lda,test,type="class")
bn.predict



```
> bn.predict<- predict(bn.lda,test,type="class")
> bn.predict
$class
  [1] 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 [31] 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
 [61] 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0
 [91] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[121] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[151] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[181] 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[211] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[241] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[271] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[301] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[331] 1 1 1 1 1 1 1 1 1 1 1 1 1
Levels: 0 1
```

It predicts the output from the data test.
The accuracy is predicted using the confusion matrix:



```
> confusionMatrix(table(bn.predict$class,test$V5))
Confusion Matrix and Statistics


      0   1
  0 176   0
  1   6 161

               Accuracy : 0.9825
                 95% CI : (0.9623, 0.9936)
    No Information Rate : 0.5306
    P-Value [Acc > NIR] : < 2e-16

                  Kappa : 0.965

 Mcnemar's Test P-Value : 0.04123

            Sensitivity : 0.9670
            Specificity : 1.0000
```

The accuracy of the given model is 98.2%.

b) LDA as a classifier algorithm:

LDA on its own can be used to classify.We are modeling the data as a set of multivariate normal distributions.Classification uses the Bayes approach.
In the dataset letter_recognition.data LDA is used for classification only.

LDA cannot reduce dimensions from this data set because the number of dimensions are already less then c-1.

The classification is as follows:

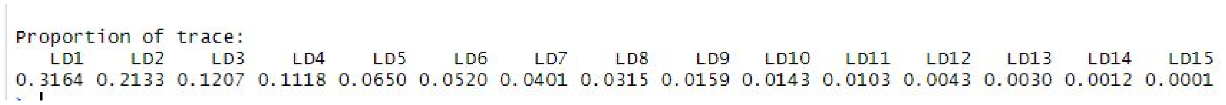
library(MASS)

lr.lda<-lda(V1 ~.,data=train)

lr.lda

The explanation for the lda function is same as above

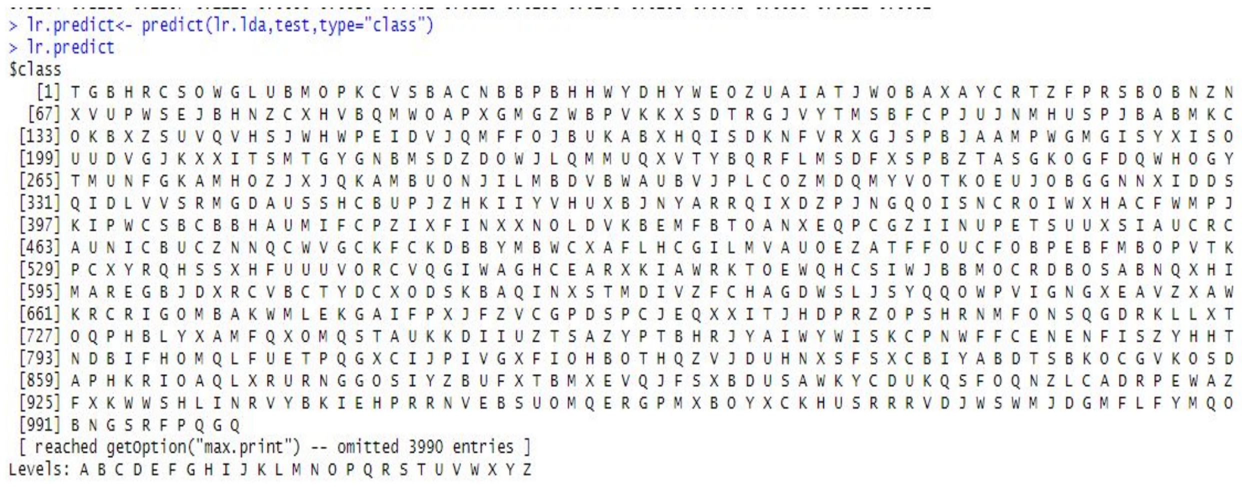
The output of lr.lda gives:



```
Proportion of trace:
   LD1    LD2    LD3    LD4    LD5    LD6    LD7    LD8    LD9   LD10   LD11   LD12   LD13   LD14   LD15
0.3164 0.2133 0.1207 0.1118 0.0650 0.0520 0.0401 0.0315 0.0159 0.0143 0.0103 0.0043 0.0030 0.0012 0.0001
```

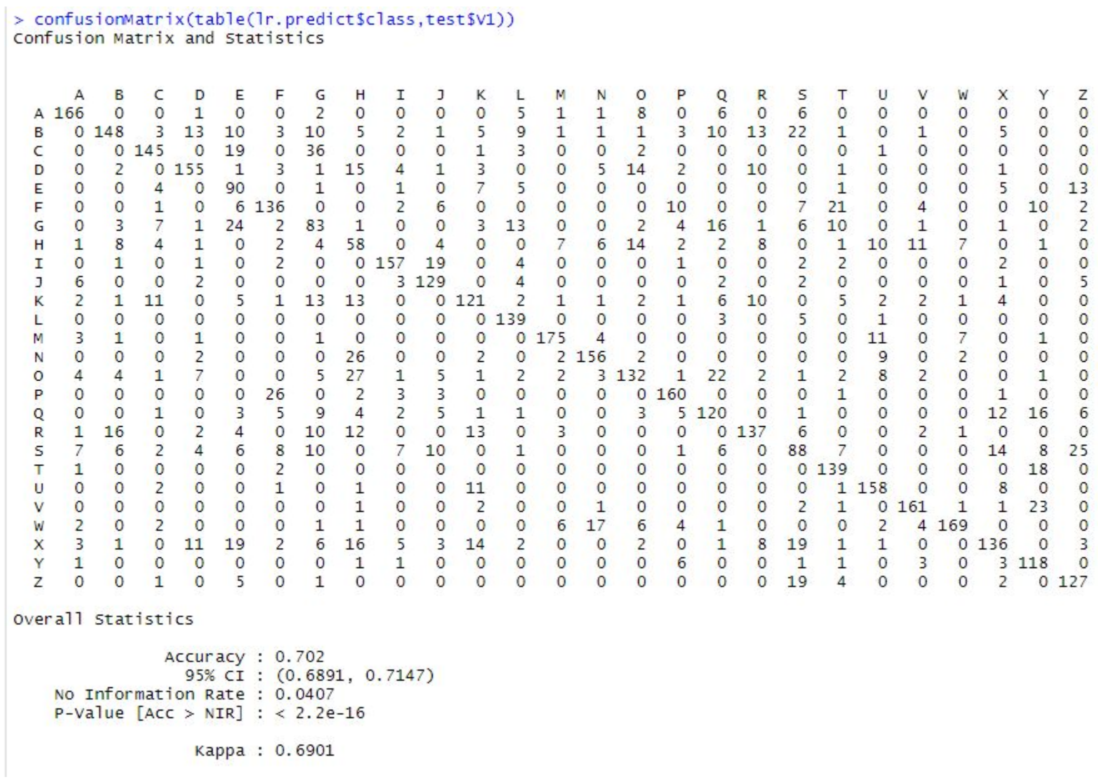
It gives till LD15 because the dimensions were reduced from another method.

Then predict method is used to test the model(lr.lda)from the test dataset.



```
> lr.predict<- predict(lr.lda,test,type="class")
> lr.predict
$class
   [1] T G B H R C S O W G L U B M O P K C V S B A C N B B P B H H W Y D H Y W E O Z U A I A T J W O B A X A Y C R T Z F P R S B O B N Z N
  [67] X V U P W S E J B H N Z C X H V B Q M W O A P X G M G Z W B P V K K X S D T R G J V Y T M S B F C P J U J N M H U S P J B A B M K C
 [133] O K B X Z S U V Q V H S J W H W P E I D V J Q M F F O J B U K A B X H Q I S D K N F V R X G J S P B J A A M P W G M G I S Y X I S O
 [199] U U D V G J K X X I T S M T G Y G N B M S D Z D O W J L Q M M U Q X V T Y B Q R F L M S D F X S P B Z T A S G K O G F D Q W H O G Y
 [265] T M U N F G K A M H O Z J X J Q K A M B U O N J I L M B D V B W A U B V J P L C O Z M D Q M Y V O T K O E U J O B G G N N X I D D S
 [331] Q I D L V V S R M G D A U S S H C B U P J Z H K I I Y V H U X B J N Y A R R Q I X D Z P J N G Q O I S N C R O I W X H A C F W M P J
 [397] K I P W C S B C B B H A U M I F C P Z I X F I N X X N O L D V K B E M F B T O A N X E Q P C G Z I I N U P E T S U U X S I A U C R C
 [463] A U N I C B U C Z N N Q C W V G C K F C K D B B Y M B W C X A F L H C G I L M V A U O E Z A T F F O U C F O B P E B F M B O P V T K
 [529] P C X Y R Q H S S X H F U U U V O R C V Q G I W A G H C E A R X K I A W R K T O E W Q H C S I W J B B M O C R D B O S A B N Q X H I
 [595] M A R E G B J D X R C V B C T Y D C X O D S K B A Q I N X S T M D I V Z F C H A G D W S L J S Y Q Q O W P V I G N G X E A V Z X A W
 [661] K R C R I G O M B A K W M L E K G A I F P X J F Z V C G P D S P C J E Q X X I T J H D P R Z O P S H R N M F O N S Q G D R K L L X T
 [727] O Q P H B L Y X A M F Q X O M Q S T A U K K D I I U Z T S A Z Y P T B H R J Y A I W Y W I S K C P N W F F C E N E N F I S Z Y H H T
 [793] N D B I F H O M Q L F U E T P Q G X C I J P I V G X F I O H B O T H Q Z V J D U H N X S F S X C B I Y A B D T S B K O C G V K O S D
 [859] A P H K R I O A Q L X R U R N G G O S I Y Z B U F X T B M X E V Q J F S X B D U S A W K Y C D U K Q S F O Q N Z L C A D R P E W A Z
 [925] F X K W W S H L I N R V Y B K I E H P R R N V E B S U O M Q E R G P M X B O Y X C K H U S R R R V D J W S W M J D G M F L F Y M Q O
 [991] B N G S R F P Q G Q
 [ reached getOption("max.print") -- omitted 3990 entries ]
Levels: A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
```

Then to calculate the accuracy confusion matrix is used as follows:



```
> confusionMatrix(table(lr.predict$class,test$V1))
Confusion Matrix and Statistics


      A   B   C   D   E   F   G   H   I   J   K   L   M   N   O   P   Q   R   S   T   U   V   W   X   Y   Z
  A 166   0   0   1   0   0   2   0   0   0   0   5   1   1   8   0   6   0   6   0   0   0   0   0   0   0
  B   0 148   3  13  10   3  10   5   2   1   5   9   1   1   1   3  10  13  22   1   0   1   0   5   0   0
  C   0   0 145   0  19   0  36   0   0   0   1   3   0   0   2   0   0   0   0   0   1   0   0   0   0   0
  D   0   2   0 155   1   3   1  15   4   1   3   0   0   5  14   2   0  10   0   1   0   0   0   1   0   0
  E   0   0   4   0  90   0   1   0   1   0   7   5   0   0   0   0   0   0   0   1   0   0   0   5   0  13
  F   0   0   1   0   6 136   0   0   2   6   0   0   0   0   0  10   0   0   7  21   0   4   0   0  10   2
  G   0   3   7   1  24   2  83   1   0   0   3  13   0   0   2   4  16   1   6  10   0   1   0   1   0   2
  H   1   8   4   1   0   2   4  58   0   4   0   0   7   6  14   2   2   8   0   1  10  11   7   0   1   0
  I   0   1   0   1   0   2   0   0 157  19   0   4   0   0   0   1   0   0   2   2   0   0   0   2   0   0
  J   6   0   0   2   0   0   0   0   3 129   0   4   0   0   0   0   2   0   2   0   0   0   0   1   0   5
  K   2   1  11   0   5   1  13  13   0   0 121   2   1   1   2   1   6  10   0   5   2   2   1   4   0   0
  L   0   0   0   0   0   0   0   0   0   0   0 139   0   0   0   0   3   0   5   0   1   0   0   0   0   0
  M   3   1   0   1   0   0   1   0   0   0   0   0 175   4   0   0   0   0   0   0  11   0   7   0   1   0
  N   0   0   0   2   0   0   0  26   0   0   2   0   2 156   2   0   0   0   0   0   9   0   2   0   0   0
  O   4   4   1   7   0   0   5  27   1   5   1   2   2   3 132   1  22   2   1   2   8   2   0   0   1   0
  P   0   0   0   0   0  26   0   2   3   3   0   0   0   0   0 160   0   0   0   1   0   0   0   1   0   0
  Q   0   0   1   0   3   5   9   4   2   5   1   1   0   0   3   5 120   0   1   0   0   0   0  12  16   6
  R   1  16   0   2   4   0  10  12   0   0  13   0   3   0   0   0   0 137   6   0   0   2   1   0   0   0
  S   7   6   2   4   6   8  10   0   7  10   0   1   0   0   0   1   6   0  88   7   0   0   0  14   8  25
  T   1   0   0   0   0   2   0   0   0   0   0   0   0   0   0   0   0   0   0 139   0   0   0   0  18   0
  U   0   0   2   0   0   1   0   1   0   0  11   0   0   0   0   0   0   0   0   1 158   0   0   8   0   0
  V   0   0   0   0   0   0   0   1   0   0   2   0   0   1   0   0   0   0   2   1   0 161   1   1  23   0
  W   2   0   2   0   0   0   1   1   0   0   0   0   6  17   6   4   1   0   0   0   2   4 169   0   0   0
  X   3   1   0  11  19   2   6  16   5   3  14   2   0   0   2   0   1   8  19   1   1   0   0 136   0   3
  Y   1   0   0   0   0   0   0   1   1   0   0   0   0   0   0   6   0   0   1   1   0   3   0   3 118   0
  Z   0   0   1   0   5   0   1   0   0   0   0   0   0   0   0   0   0   0  19   4   0   0   0   2   0 127

Overall Statistics

               Accuracy : 0.702
                 95% CI : (0.6891, 0.7147)
    No Information Rate : 0.0407
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6901
```

And the accuracy of the given model comes out to be 70.2%

CODE FOR FLD OF LETTER RECOGNITION

```
#import the dataset letter-recognition.data

head(letter.recognition)

#using the function lfda() to reduce the dimension from 16 to 15
library(lfda)
x<-letter.recognition[,-1]
y<-letter.recognition[,1]
r<-15
model<-lfda(x,y,r,metric = "plain")
newdata<-model$Z
newdata
newdata1<-data.frame(V1=y ,
                     V2=newdata[,1],
                     V3=newdata[,2],
                     V4=newdata[,3],
                     V5=newdata[,4],
                     V6=newdata[,5],
                     V7=newdata[,6] ,
                     V8=newdata[,7],
                     V9=newdata[,8],
                     V10=newdata[,9],
                     V11=newdata[,10],
                     V12=newdata[,11],
                     V13=newdata[,12],
                     V14=newdata[,13] ,
                    V15=newdata[,14],
                    V16=newdata[,15]
                    )
newdata1

#plot(x=model,labels=y)

#splitting data into training and test sets
library(caret)
set.seed(1234)
train_index<-createDataPartition(newdata1$V1,p=0.75,list=FALSE)
train<-newdata1[train_index,]
test<-newdata1[-train_index,]

summary(test)
summary(train)

#performimg linear discriminant analysis using the function lda() to classify into different classes
library(MASS)
lr.lda<-lda(V1 ~.,data=train)
lr.lda

#prediction of test data oputput
lr.predict<- predict(lr.lda,test,type="class")
lr.predict

#finding the accuracy of the original output with the test output
table(lr.predict$class,test$V1)
confusionMatrix(table(lr.predict$class,test$V1))

#the accuracy of the model is 70.2%
```

2. Metric Learning

In metric learning, the code is on Supervised Global Distance Metric Learning. The objective of this is to attempt to learn metrics that keep all points within the same classes close while separating data points from different classes separate. It does this by learning a global distance parameter which minimizes the distance between the data pairs in the equivalence constraints subject to the constraint that the data pairs are well separated.

#INFORMATION ABOUT Haberman's Survival DATASET

The dataset we have taken for observing the performance of Global metric learning algorithm is the Haberman's Survival Data Set. This dataset contains cases from a study that was conducted between 1958 and 1970 at the University of Chicago's Billings Hospital on the survival of patients who had undergone surgery for breast cancer. The attributes of this dataset are:

1. Age of patient at time of operation (numerical)
2. Patient's year of operation (year - 1900, numerical)
3. Number of positive axillary nodes detected (numerical)

4. Survival status (class attribute)
-- 1 = the patient survived 5 years or longer
-- 2 = the patient died within 5 year

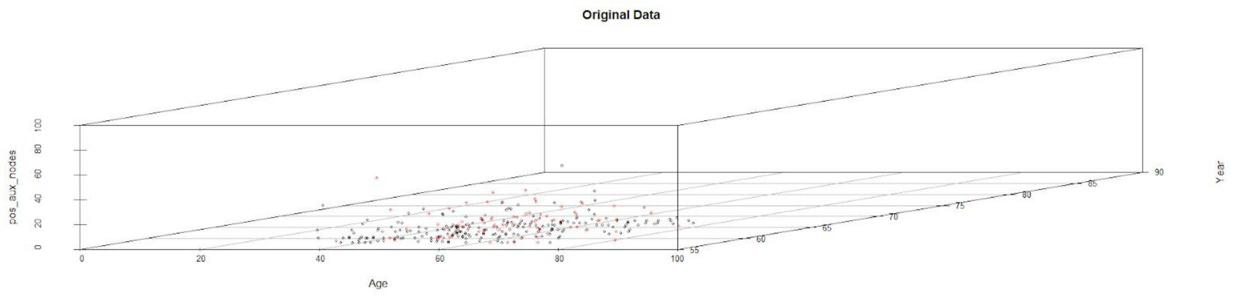
#EXPLANATION OF THE CODE

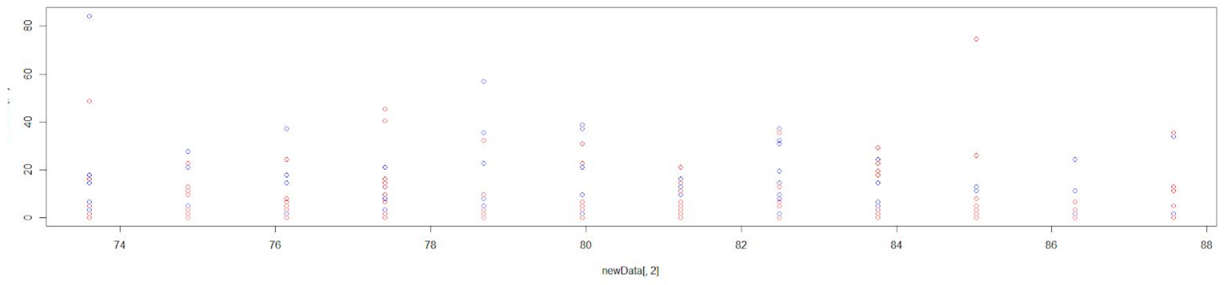
Initially in Section 1 of code the data from the .DATA file is read and stored in data variable. The names of the columns are also initialized.
In Section 2, the data is divided into similar and dissimilar classes according to their survival status and saved in two variables: alive and dead. And all their possible combination are stored in simi and dism variables. These along with the data variable will be used to do the global distance metric dimensionality reduction.
In section 3, data, simi and dism are used in GdmDiag and a dimension is reduced and their graphs are plotted using the scatterplot3d function.
In section 4, the dimension reduced data is stored in red_data.
In section 5, the data is divided into train and test data and is trained in linear discriminative analysis and is tested on the test data and confusion matrix and accuracy levels are observed in section 6.

CODE FOR HABERMAN'S DATA SET

```
library("plot3D")
library("dml")
library("scatterplot3d")
library("MASS")

#Section 1

rnames<-c("Age","Year","pos_aux_nodes","survived")
data <- read.csv("haberman.DATA",header=FALSE,col.names = rnames)

l=nrow(data)

#Section 2

alive <- which(grepl(1,data$survived))
dead <- which(grepl(2,data$survived))

simi <- rbind(t(combn(alive, 2)), t(combn(dead, 2)))

temp <-  as.data.frame(t(simi))
tol <- as.data.frame(combn(1:l, 2))
dism <- t(as.matrix(tol[!tol %in% temp]))


#Section 3

result <- GdmDiag(data, simi, dism)
newData <- result$newData

color <- c(1:l)

for (value in color){
    if  (data$survived[value]==1){
        color[value] <- "red"
    }
    else {
        color[value] <- "blue"
    }
}



par(mfrow = c(2, 1), mar = rep(0, 4) + 0.1)
scatterplot3d(data, color = color, cex.symbols = 0.6,
              xlim = range(data[, 1], newData[, 1]),
              ylim = range(data[, 2], newData[, 2]),
              zlim = range(data[, 3], newData[, 3]),
              main = "Original Data")
# plot GdmDiag transformed data
scatterplot3d(newData, color = color, cex.symbols = 0.6,
              xlim = range(data[, 1], newData[, 1]),
              ylim = range(data[, 2], newData[, 2]),
              zlim = range(data[, 3], newData[, 3]),
              main = "Transformed Data")

plot(newData[,2],newData[,3],col=color)


#Section 4

set.seed(123)

red_data <- data.frame(V1=newData[,2],V2=newData[,3],V3=newData[,4])


#Section 5

smp_size <- floor(0.75 * nrow(red_data))

train_ind <- sample(seq_len(nrow(red_data)), size = smp_size)

train <- red_data[train_ind, ]
test <- red_data[-train_ind, ]

lda_out <- lda(V3 ~.,data=train)

lda_pred<- predict(lda_out,test,type="class")

table(lda_pred$class,test$V3)

confusionMatrix(table(lda_pred$class,test$V3))
```

GRAPHS





The image on top is the original data and the graph below is the one after dimensionality reduction.

ACCURACY LEVELS

```
               Accuracy : 0.6883
                 95% CI : (0.5726, 0.7891)
    No Information Rate : 0.7143
    P-Value [Acc > NIR] : 0.73931

                  Kappa : 0.087
 Mcnemar's Test P-Value : 0.02474

            Sensitivity : 0.8909
            Specificity : 0.1818
         Pos Pred Value : 0.7313
         Neg Pred Value : 0.4000
             Prevalence : 0.7143
         Detection Rate : 0.6364
   Detection Prevalence : 0.8701
      Balanced Accuracy : 0.5364

       'Positive' Class : 0.906395123004954
```

Accuracy of this model is 68.83%

3. Comparison with Unsupervised Dimensionality Reduction Techniques

In the following segment, the results obtained by performing FLD and Metric Learning on the same dataset, i.e., the Banknote Authentication Dataset and have been compared to the results after performing a supervised dimensionality reduction technique, namely PCA on the same dataset.

1. FLD

```
#import the dataset data_banknote_authentication.txt
head(data_banknote_authentication)

#using the function lfda() to reduce the dimension from 4 to 3

library(lfda)
x<-data_banknote_authentication[,-5]
y<-data_banknote_authentication[,5]
r<-3
model<-lfda(x,y,r,metric = "plain")
newdata<-model$Z
newdata
newdata1<-data.frame(V1=y ,
                V2=newdata[,1],
                V3=newdata[,2],
                V4=newdata[,3]
)
newdata1

plot(x=model,labels=y)


#splitting data into training and test sets
library(caret)
set.seed(123)
train_index<-createDataPartition(newdata1$V1,p=0.75,list=FALSE)
train<-newdata1[train_index,]
test<-newdata1[-train_index,]
summary(test)
summary(train)

#performimg fisher's linear discriminant analysis using the function lda()
library(MASS)
bn.lda<-lda(V1 ~.,data=train)
bn.lda

#prediction of test data oputput
bn.predict<- predict(bn.lda,test,type="class")
#plot(test,bn.predict)
bn.predict
```

```
#finding the accuracy of the original output with the test output
table(bn.predict$class,test$V1)
confusionMatrix(table(bn.predict$class,test$V1))

#the accuracy of the model is 98.25%
```

Accuracy of this model is 98.25%

2. Metric Learning

```
library("plot3D")

library("dml")

library("scatterplot3d")

library("MASS")


#Section 1


rnames<-c("Variance","Skewness","Curtosis","Entropy","Class")

data <- read.csv("banknote.csv",header=FALSE,col.names = rnames)

l=nrow(data)


#Section 2


real <- which(grepl(1,data$Class))

fake <- which(grepl(0,data$Class))

simi <- rbind(t(combn(real, 2)), t(combn(fake, 2)))


temp <-  as.data.frame(t(simi))

tol <- as.data.frame(combn(1:l, 2))

dism <- t(as.matrix(tol[!tol %in% temp]))


#Section 3


result <- GdmDiag(data, simi, dism)

newData <- result$newData


color <- c(1:l)


for (value in color){

  if  (data$Class[value]==1){
        color[value] <- "red"
    }
    else {
        color[value] <- "blue"
    }
}



#Section 4


set.seed(123)


red_data <- data.frame(V1=newData[,1],V2=newData[,3],V3=newData[,5])


#Section 5
smp_size <- floor(0.75 * nrow(red_data))


train_ind <- sample(seq_len(nrow(red_data)), size = smp_size)


train <- red_data[train_ind, ]
test <- red_data[-train_ind, ]


lda_out <- lda(V3 ~.,data=train)


#Section 6
lda_pred<- predict(lda_out,test,type="class")


table(lda_pred$class,test$V3)


confusionMatrix(table(lda_pred$class,test$V3))
```

<u>The accuracy of this model is 86.59%.</u>

3. PCA

PCA is a supervised dimensionality reduction technique which uses orthogonal linear transformations to extract the most useful and least correlated components from the given dataset, thereby reducing the dimensions of the data without th

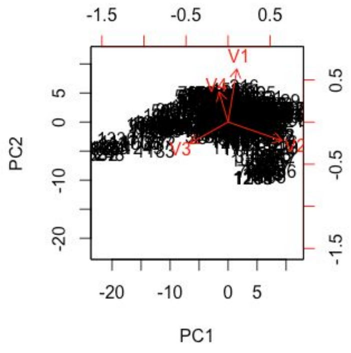
```
#importing the dataset data_banknote_authentication

head(data_banknote_authentication)



#splitting the data into training and test sets

#We should not combine the train and test set to obtain PCA components of whole data at once. Because, this would violate the entire assumption of generalization since test data would get ‘leaked’ into the training set. In other words, the test data set would no longer remain ‘unseen’. Eventually, this will hammer down the generalization capability of the model.

library(caret)

set.seed(123)

train_index<-createDataPartition(data_banknote_authentication$V5,p=0.75,list=FALSE)

train<-data_banknote_authentication[train_index,]

test<-data_banknote_authentication[-train_index,]

summary(test)

summary(train)



#V5 is categorical so we will drop it

test <- test[, c(1:4)]



#performing PCA on test using prcomp

test.pca <- prcomp(test, cor=TRUE, score=TRUE)



summary(test.pca)



#It can be noted that the first 3 components constitute 97.34% of the data

#They also have majority proportion of the variance



#plot of the components

plot(test.pca)
```

```
#getting loading variables which specify how individual attributes contribute to the components

test.pca$loadings


#obtaining transformed components using scores

test2 <- test.pca$scores

head(test2)


biplot(test.pca, scale=0)


#compute standard deviation of each principal component

std_dev <- test.pca$sdev


#compute variance

pr_var <- std_dev^2


#proportion of variance explained

propvar <- pr_var/sum(pr_var)


#scree plot

plot(propvar, xlab = "Principal Component",

     ylab = "Proportion of Variance Explained",

     type = "b")

#From scree plot it becomes apparent that we only need to take first 3 PCs


#add a training set with principal components

train.data <- data.frame(V5=data_banknote_authentication$V5, test.pca$x)


#we are interested in first 3 PCs

train.data <- train.data[,1:4]
```

```
#run a decision tree

library(rpart)

rpart.model <- rpart(V5 ~ .,data = train.data, method = "anova")

rpart.model


#transform test into PCA

test.data <- predict(test.pca, newdata = test)

test.data <- as.data.frame(test.data)
```

GRAPHS

1. This biplot depicts the degree of correlation between the principal components.
Since the SCALE=0, all arrows have been scaled to show their respective loadings.



2.This screeplot represents the share of variance of each principal component in a decreasing fashion. It tells us which components are sufficient to plot the entire data.

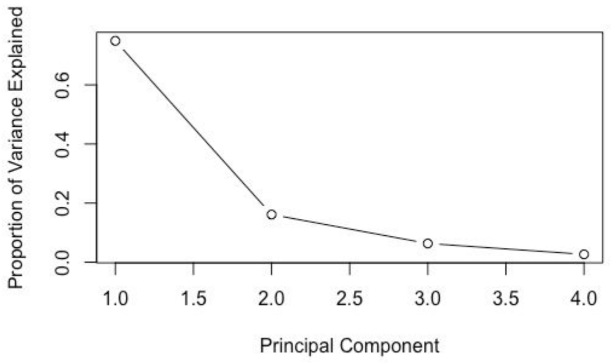
CONCLUSION

The comparison between these 3 learning algorithms can be done by applying them on one particular dataset and comparing the results. We have applied them on the banknote authentication dataset and the following results were observed.

FLD : 98.25%

METRIC LEARNING : 86.59%

From these accuracy levels we can conclude that FLD works best on the given dataset with an accuracy level of 98.25%. Also, metric learning gives an accuracy learning of 86.59%. These accuracies may vary from one dataset to another, but for the given dataset these are the accuracies obtained by the algorithms given above.


Proportion of Variance Explained
Principal Component