UL Assignment 2

Gareth Edwards

12/11/2020

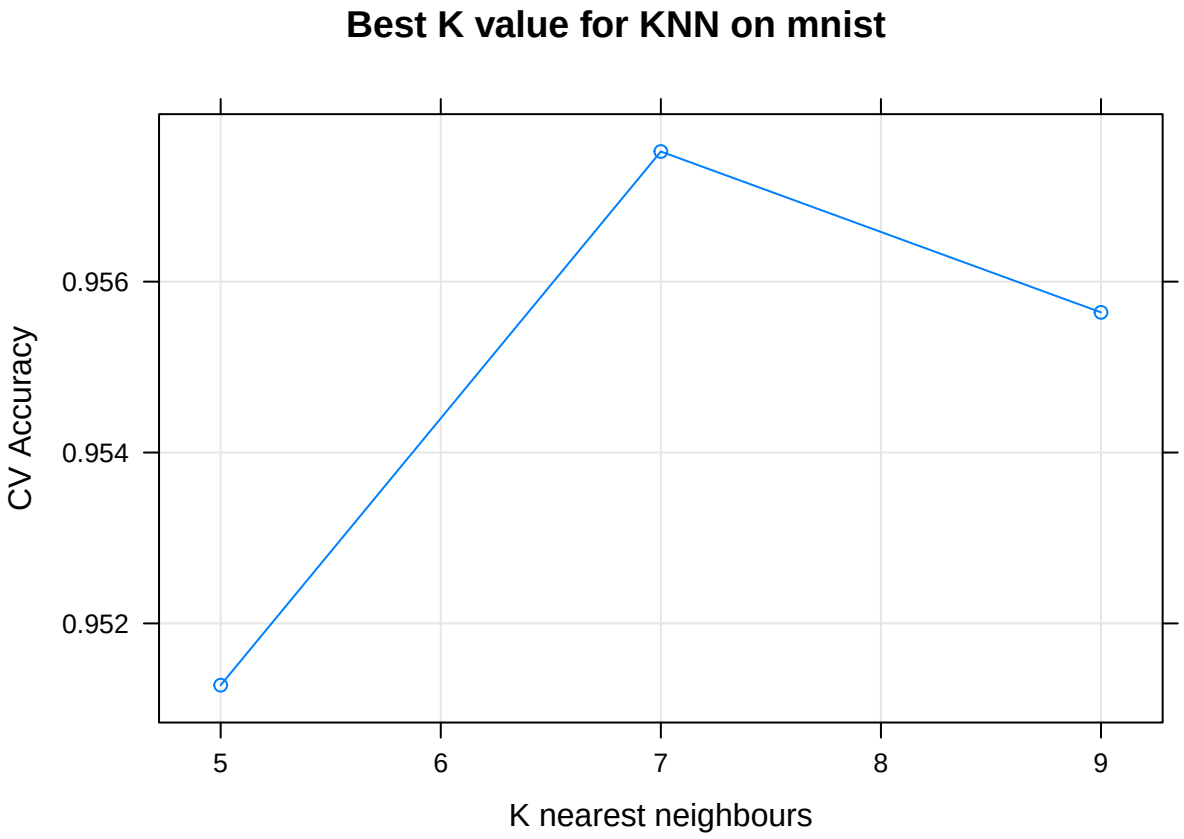
Introduction

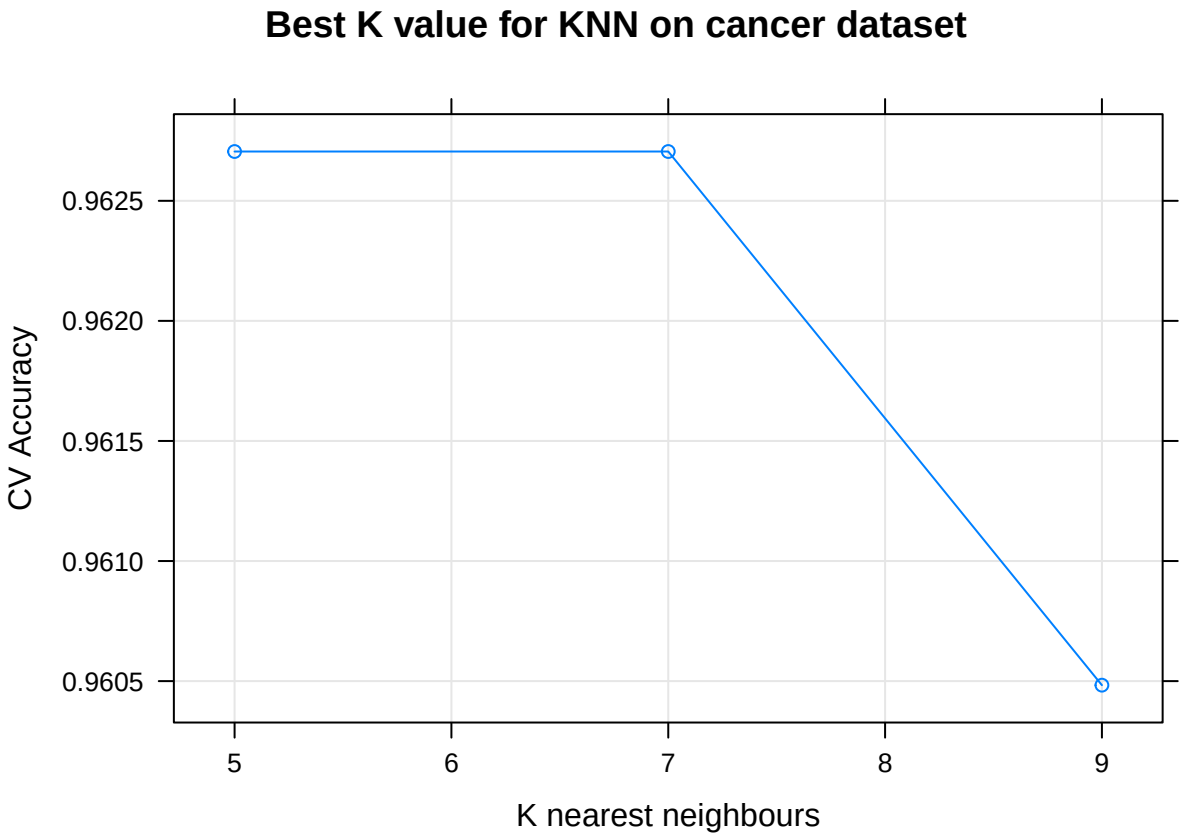
Two different datasets were used in this assignment. The mnist dataset containing pixel values of hand written digits 5 and 8, and a breast cancer dataset with image information on a breast mass. The mnist dataset contains 10000 observations. 5000 observations being a hand written number 5 and the other 5000 a hand written number 8. For the purposes of reducing computation time, 2000 random observations of the mnist dataset is used and split into the train and test sets. Each cell in the mnist dataset represents the information of a pixel from a 28x28 image of the hand written digit. The breast cancer dataset consists of 569 observations with 32 variables. Each variable being information gleened from an image of a breast mass. These variables are used to determing whether the breast mass is malignant or benign.

Three different dimensionality reduction techniques were used on each dataset. Principal Component Analysis (PCA), Multidimensional Scaling (MDS) and isoMaps. The aim is to identify which dimensionality reduction technique is best, for each dataset, with respects to classification accuracy. The classification algorithm used is K Nearest Neighbours (KNN). KNN is first performed on the initial datasets and then on the dimensionally reduced datasets and the accuracy of each is compared. A further comparison is made between differing amounts of dimensions the data is reduced to. Each dataset is reduced to 2, 3, 4, 5 and 6 dimensions. This is done for each dimensionality reduction technique. KNN is then performed on each level of dimensionality reduction. By performing these experiments we note which methods are best in classifying data in terms of accuracy and computation time.

Each dataset contained no missing data values. This increases the accuracy of the KNN model and increases the realiabilty of the dimensionality reduction techniques. The cancer dataset was first scaled before performing KNN or any of the dimensionality reduction techniques.

The mnist subset and cancer dataset was split into a train and a test set (80/20 split). Initially KNN is applied to the original train datasets. The plot below shows the optimal K value to use in the clustering algorithm. In this case, for the mnist and cancer dataset the optimal K value is both 7. This means that each observation will be classified according to K of its nearest neighbours. For example, if the majority of the 7 nearest neighbours are classified as the digit 5 then the current digit observation will also be classified as 5. Similarily, if the majority of the 7 nearest neighbours of an observation from the cancer dataset is malignant, then the current observation will also be classified as malignant.


Best K value for KNN on mnist
CV Accuracy
0.956
0.954
0.952
5
6
7
8
9
K nearest neighbours

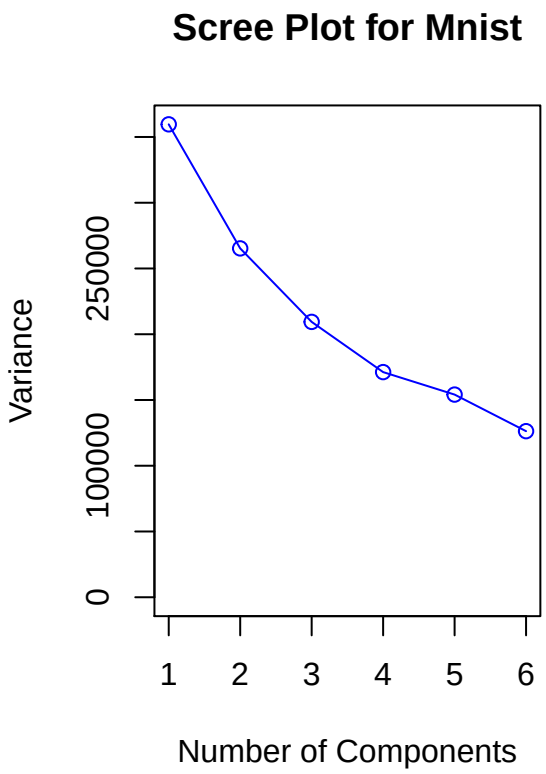


The test accuracy for the KNN on the original mnist and cancer datasets are 0.97 an 0.96 respectively. Each of the 3 dimensionality reduction algorithms will be applied to each dataset and KNN applied to the reduced dimensions. The accuracies of each KNN output will be compared to that of the original dataset. Further discussion will then take place regarding the implication of these results.

Principal Component Analysis

PCA is a process that decomposes data (using eigen decomposition). That is, it reduces the dimensions of the data by calculating the axes within the data subspace. PCA essentially describes data using fewer dimensions based on the subspace of the data. PCA thus works best if data lies on or close to a linear subspace. Each component that the data is broken up into contains more of the total variability of the data compared to the component before it. Therefore, if a dataset is broken up into 5 components, or variables, then the first component will contain more variablility of the original data than the second component. The third component will contain more variability than the fourth but less than the second such that the combination of all 5 components accounts for all the variability in the original dataset. The scree plots below show the amount of original variability contained by each component for both the mnist and cancer datasets.

PCA was applied to both the mnist subset and cancer datasets. Each dataset was decomposed into 6 components. KNN was then applied to these components. First using 2, 3, 4, 5 and then all 6 components. The test set accuracies was calculated after KNN was applied to each group of components and a table of the results was produced.



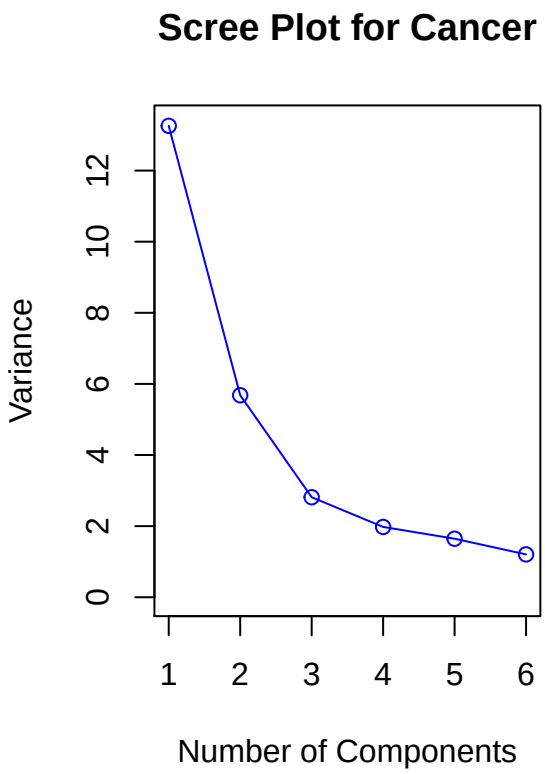


Table 1: Test Accuracy of KNN applied to PCA of Mnist dataset

Components	Accuracy
2	0.6390977
3	0.9423559
4	0.9624060
5	0.9473684
6	0.9598997

Table 2: Test Accuracy of KNN applied to PCA of Cancer dataset

Components	Accuracy
2	0.9380531
3	0.9380531
4	0.9469027
5	0.9203540
6	0.9469027

In both cases the KNN test accuracy was higher when the dataset is reduced to 4 components. With the maximum accuracy being 0.96 and 0.95 for the mnist and cancer datasets respectively. The accuracy in this clustering shows that the data for both datasets could possibly lie on a lower dimensional subspace. This makes sense especially for the mnist dataset as the pixel values are 0 in most areas and actual values take up a smaller portion of the dataset. Therefore, it will be highly beneficial to reduce dimensions on the mnist

dataset, or any other similar dataset, before performing a clustering algorithm such as KNN.

Multidimensional Scaling

The second dimensionality reduction technique used on the two datasets is Multidimensional Scaling (MDS). MDS starts with a proximity matrix of the original data and uses it to derive an n x p data matrix with less dimensions. The proximity matrix of this derived n x p data matrix should be as similar as possible compared to the initial proximity matrix.

For both datasets we calculate the proximity matrix by finding the euclidean distance from each point to every other point. The result is a n x n proximity matrix of distances between observations. This proximity matrix is then used to compute a data set with 6 dimensions. This is done using various matrix calculations. Below are the tables showing the accuracy of the KNN algorithm on the reduced dimensions, for each dataset, using MDS.

Table 3: Test Accuracy of KNN applied to MDS of digit dataset

Components	Accuracy
2	0.6666667
3	0.9022556
4	0.9373434
5	0.9323308
6	0.9674185

Table 4: Test Accuracy of KNN applied to MDS of cancer dataset

Components	Accuracy
2	0.9203540
3	0.9469027
4	0.9380531
5	0.9380531
6	0.9823009

Using MDS we can see that KNN performs relatively well on 4 components like in the case of PCA. However, for the mnist dataset the KNN clustering accuracy was at a maximum when applied to 6 components. The clustering accuracies are relatively good for the MDS approach. MDS is, however, computationally expensive. The generation of a proximity matrix alone is computationally expensive. The complexity of calculating the distance matrix is N(N - 1)/2 * 3P. This means the time increases significantly as P (the number of variables) and N (the number of observations) increase. For large datasets it is therefore not efficient to use MDS for dimensionality reduction.

MDS works well for the cancer dataset. Applying KNN on 6 components of the cancer data gives us a test accuracy of 0.98. This is higher than the KNN test accuracy on the original cancer dataset.

IsoMap

While PCA and MDS discover the linear subspace of a dataset, IsoMap discovers the non-linear subspace. If data is lying on a non-linear subspace, PCA and MDS would not be able to accurately summarise the data into components. This is especially true for MDS as the distances used for the proximity matrix are shortest euclidean distances between each point. In a non-linear subspace we can no longer make use of straight line euclidean distances.

IsoMaps are very similar to MDS. The only difference is the method in which the proximity matrix is formed. In MDS the proximity matrix was calculated using the shortes euclidean distances between observations. For IsoMaps the proximity matrix is calculated by initially finding a certain number of k closest neighbours to each observation and finding the distance to each of these neighbours (k = 10 in this case). This is repeated for all the observations in the dataset. Naturally there will be missing values in this proximity matrix. These missing distances are computed by using a shortest path algorithm. The resulting proximity matrix is known as a weighted graph. This shortest path algorithm ensures that all the distances calculated remain in the non-linear subspace and is not just straight line distances calculated from one point to all other points. MDS is then performed on the weighted graph

Table 5: Test Accuracy of KNN applied to IsoMap of mnist dataset

Components	Accuracy
2	0.9649123
3	0.9774436
4	0.9774436
5	0.9774436
6	0.9699248

Table 6: Test Accuracy of KNN applied to IsoMap of cancer dataset

Components	Accuracy
2	0.9469027
3	0.9203540
4	0.9646018
5	0.9469027
6	0.9292035

The KNN test accuracy on the components of the mnist dataset, after applying IsoMap, has a very high test accuracy with the maximum being at 6 components with a value of 0.97. This implies that the data in the mnist dataset lie on a non-linear subspace as the data was accurately summarize in each component. For a dataset such as the mnist dataset the best dimension reduction to use for clustering purposes would be a non-linear dimensionality reduction technique such as IsoMap. Again, however, the computation time of IsoMap is greater than PCA's as, like with MDS, a proximity matrix is created which is generally computationally expensive.

The results for the Cancer dataset, after applying IsoMap, showed that applying KNN to 4 components produced a test accuracy of 0.965 which is slightly better than the KNN test accuracy of the original.

Discussion

Reducing the dimensions for the mnist dataset proved essential to improve the clustering of the dataset. The best dimension reduction technique for the mnist dataset, with respects to consequent KNN application, was the IsoMap approach. This shows us the nature of the mnist data. The mnist dataset is fairly sparse and the actual values seem to lie on a non-linear subspace that is accurately observed by the IsoMap approach. The IsoMap components are thus accurate summaries of the mnist data and thus can also be used for reliable clustering.

The original cancer dataset clustered well without dimensionality reduction. There was, however, improvement when KNN was applied to the reduced cancer dataset. The improvement showed especially when the cancer dataset was reduced using MDS. The test accuracy after applying MDS on the cancer dataset was 0.98. Interestingly, even when KNN was applied to 2 components the test accuracy was still higher than the original. Applying KNN to 2 components, as apposed to the original 30 variables, is more efficient and reduces computation time. However, one is also to factor in the computation time of MDS which, in itself, takes more time than KNN on the original cancer dataset. Therefore, if you are looking for maximum accuracy and computation time is not a constraint, MDS should be used to summarise the cancer data before clustering.

Based on the results of the tests performed, one would recommend using IsoMap for the mnist dimension reduction and MDS for the cancer dataset reduction. This is, however, dependent on the context of the problem at hand. Especially in the case of the cancer dataset. The mnist dataset is the most necessary to reduce as the computation time of KNN on the full dataset is immense. Summarising the data into components is therefore necessary to increase clustering accuracy and decrease computation time. The cancer dataset need only be reduced if one requires the maximum possible accuracy. If one is satisfied with an accuracy of, approximately, 0.9 then there is no need to reduce dimensionality prior to clustering. However, as the dataset increases in size the computation time of KNN on the original dataset will become too high and will thus require one to reduce the dimensions first before clustering.

An interesting point of information was seen in the test accuracy on the various numbers of components. When KNN was applied to four components it seemed to produce either the best or second best test accuracy. This was true for PCA (for both datasets), MDS (on both datasets) and IsoMap on the cancer dataset. This could suggest that, on average, reducing data to 4 components could be a sufficient summary of the dataset.

Conclusion

Dimensioanlity reduction is a powerful tool in summarising a dataset. This is especially important for large datasets. Dimension reduction not only improves clustering accuracy for some datasets but also reduces computation time of the clustering algorithm on large datasets. Reducing dimensions comes with a drawback in that it is time consuming. MDS and IsoMap is the most time consuming due to the calculation of the proximity matrix. But while it is time consuming, it also reduces the data more precisley than PCA. In the end one should carefully consider whether it is necessary to take more time to achieve greater accuracy or is the faster methods, like PCA, accurate enough for the task at hand.

Appendix

```
knitr::opts_chunk$set(echo = FALSE)
knitr::opts_chunk$set(message = FALSE)
knitr::opts_chunk$set(results = 'hide')
knitr::opts_chunk$set(warning = FALSE)
knitr::opts_chunk$set(fig.keep = 'all')
```

```
library(tidyverse)
library(tidyr)
library(dplyr)
library(caret)
library(smacof)
library(RDRToolbox)
library(knitr)

# loading data
digit.dat <- read.csv("data/min_mnist.csv")
cancer.dat <- read.csv("data/WDBC.csv")

digit.dat$X5 <- as.factor(digit.dat$X5)
cancer.dat$diagnosis <- as.factor(cancer.dat$diagnosis)

set.seed(2020)
digit.subset <- sample(nrow(digit.dat), size = 2000) # created to be used in models as 10000 observatio
                                                     # are computaionally expensive
digit.subset <- digit.dat[digit.subset,]

# split into train and test set
set.seed(2020)
train.index <- createDataPartition(digit.subset$X5, p = 0.8, list = F) # 80 20 split
dig.train <- digit.subset[train.index,]
dig.test <- digit.subset[-train.index,]

# split the data (cancer data)
set.seed(2020)
train.index <- createDataPartition(cancer.dat$diagnosis, p = 0.8, list = F) # 80 20 split
cancer.train <- cancer.dat[train.index,]
cancer.test <- cancer.dat[-train.index,]

par(mfrow = c(1,2))
# further info for the train method: Using 10 fold cv and repeating that once
# aggregate the validation results
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
set.seed(2020)
digit.knn <- train(as.factor(X5) ~ ., data = dig.train,
                   method = "knn",
                   trControl = trctrl,
                   preProcess = c("center", "scale"))
```

```r
plot(digit.knn, main = 'Best K value for KNN on mnist',
     xlab = 'K nearest neighbours',
     ylab = 'CV Accuracy') # viewing the effects on CV of the different K values

# further info for the train method. Using 10 fold cv and repeating that once
# aggregate the validation results
trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
set.seed(2020)
cancer.knn <- train(as.factor(diagnosis) ~ ., data = cancer.train[,-1],
                    method = "knn",
                    trControl = trctrl,
                    preProcess = c("center", "scale"))

plot(cancer.knn, main = 'Best K value for KNN on cancer dataset',
     xlab = 'K nearest neighbours',
     ylab = 'CV Accuracy') # viewing the effects on CV of the different K values


# predictions on the test set (mnist)
dig.test.preds <- predict(digit.knn, dig.test[,-1])
dig.orig.test.acc <- sum(dig.test.preds == dig.test$X5)/nrow(dig.test) # accuracy

#confusionMatrix(dig.test.preds, dig.test$X5)

# predictions on the test set (cancer)
cancer.test.preds <- predict(cancer.knn, cancer.test[,-c(1,2)])
can.orig.test.acc <- sum(cancer.test.preds == cancer.test$diagnosis)/nrow(cancer.test) # accuracy

#confusionMatrix(as.factor(cancer.test.preds), as.factor(cancer.test$diagnosis))




par(mfrow = c(1,2))
# PCA on Mist
set.seed(2020)
digit.pca <- princomp(digit.subset[,-1], scores = T)

# extracting the components and adding labels
dat.labels <- as.data.frame(digit.subset$X5)
pca.comps.dig <- as.data.frame(digit.pca$scores[,1:6])
pca.comps.dig <- cbind.data.frame(dat.labels, pca.comps.dig)
names(pca.comps.dig)[1] <- "labels"
pca.comps.dig$labels <- as.factor(pca.comps.dig$labels)


# creating the scree plot
variance <- (digit.pca$sdev)^2
max.var <- round(max(variance), 1)
comps <- as.integer(c(1,2,3,4, 5, 6))
plot(comps, variance[1:6], main = "Scree Plot for Mnist", xlab = "Number of Components",
     ylab = "Variance", type = "o", col = "blue", ylim = c(0, max.var))
```

```r
# PCA on cancer data
cancer.pca <- princomp(scale(cancer.dat[,-c(1,2)]), scores = T)

# extracting components
cancer.labs <- as.data.frame(cancer.dat$diagnosis)
pca.comps.can <- as.data.frame(cancer.pca$scores[,1:6])
pca.comps.can <- cbind.data.frame(cancer.labs, pca.comps.can)
names(pca.comps.can)[1] <- "diagnosis"
pca.comps.can$diagnosis <- as.factor(pca.comps.can$diagnosis)


# creating the scree plot
variance <- (cancer.pca$sdev)^2
max.var <- round(max(variance), 1)
comps <- as.integer(c(1,2,3,4, 5, 6))
plot(comps, variance[1:6], main = "Scree Plot for Cancer", xlab = "Number of Components",
     ylab = "Variance", type = "o", col = "blue", ylim = c(0, max.var))


# KNN on the digits.pca

# function to run KNN on the components and then return the accuracies using confusionMatrix()
knnOnPca <- function(compData){
  # splitting data
  set.seed(2020)
  index <- createDataPartition(compData$labels, p = 0.8, list = F)
  compData.train <- compData[index,]
  compData.test <- compData[-index,]

  # performing KNN on the PCAed data
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  compData.knn <- train(labels ~ ., data = compData.train,
                        method = "knn",
                        trControl = trctrl,
                        preProcess = c("center", "scale"))

  compData.preds <- predict(compData.knn, compData.test)
  compData.acc <- sum(compData.preds == compData.test$labels)/nrow(compData.test) # accuracy

  returnList <- list("digit.knn.model" = compData.knn,
                     "digit.knn.acc" = compData.acc)
  return(returnList)
}

dig.test.accs <- c()
# using 2 components
digit.2.comps.knn <- knnOnPca(pca.comps.dig[1:3])
#digit.2.comps.knn$digit.knn.model # the knn model
dig.test.accs <- c(dig.test.accs, digit.2.comps.knn$digit.knn.acc) # model accuracy
```

```r
# using 3 components
digit.3.comps.knn <- knnOnPca(pca.comps.dig[1:4])
#digit.3.comps.knn$digit.knn.model # the knn model
#digit.3.comps.knn$digit.knn.acc # model accuracy
dig.test.accs <- c(dig.test.accs, digit.3.comps.knn$digit.knn.acc)

# using 4 components
digit.4.comps.knn <- knnOnPca(pca.comps.dig[1:5])
#digit.4.comps.knn$digit.knn.model # the knn model
#digit.4.comps.knn$digit.knn.acc # model accuracy
dig.test.accs <- c(dig.test.accs, digit.4.comps.knn$digit.knn.acc)

# Using 5 components
digit.5.comps.knn <- knnOnPca(pca.comps.dig[1:6])
#digit.5.comps.knn$digit.knn.model # the knn model
#digit.5.comps.knn$digit.knn.acc # model accuracy
dig.test.accs <- c(dig.test.accs, digit.5.comps.knn$digit.knn.acc)

# using 6 components
digit.6.comps.knn <- knnOnPca(pca.comps.dig[1:7])
#digit.6.comps.knn$digit.knn.model # the knn model
#digit.6.comps.knn$digit.knn.acc # model accuracy
dig.test.accs <- c(dig.test.accs, digit.6.comps.knn$digit.knn.acc)

# create table to show test accuarcies compared to components
dig.pca.tab <- data.frame(Components = c(2:6),
                          Accuracy = dig.test.accs)
kable(dig.pca.tab, caption = 'Test Accuracy of KNN applied to PCA of Mnist dataset')

# function to run KNN on the components and then return the accuracies using confusionMatrix()
knnOnPca <- function(compData){
  # splitting data
  compData$diagnosis <- as.factor(compData$diagnosis)
  set.seed(2020)
  index <- createDataPartition(compData$diagnosis, p = 0.8, list = F)
  compData.train <- compData[index,]
  compData.test <- compData[-index,]

  # performing KNN on the PCAed data
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  compData.knn <- train(diagnosis ~ ., data = compData.train,
                        method = "knn",
                        trControl = trctrl,
                        preProcess = c("center", "scale"))

  compData.preds <- predict(compData.knn, compData.test)
  compData.acc <- sum(compData.preds == compData.test$diagnosis)/nrow(compData.test)

  returnList <- list("cancer.knn.model" = compData.knn,
                     "cancer.knn.acc" = compData.acc)
  return(returnList)
}
```

```r
cPCA.test.acc <- c()
# 2 components
# using 2 components
cancer.2.comps.knn <- knnOnPca(pca.comps.can[1:3])
#cancer.2.comps.knn$cancer.knn.model # the knn model
cPCA.test.acc <- c(cPCA.test.acc, cancer.2.comps.knn$cancer.knn.acc) # model accuracy

# 3 components
cancer.3.comps.knn <- knnOnPca(pca.comps.can[1:4])
#cancer.3.comps.knn$cancer.knn.model # the knn model
cPCA.test.acc <- c(cPCA.test.acc, cancer.3.comps.knn$cancer.knn.acc) # model accuracy

# 4 components
cancer.4.comps.knn <- knnOnPca(pca.comps.can[1:5])
#cancer.4.comps.knn$cancer.knn.model # the knn model
cPCA.test.acc <- c(cPCA.test.acc, cancer.4.comps.knn$cancer.knn.acc) # model accuracy

# 5 components
cancer.5.comps.knn <- knnOnPca(pca.comps.can[1:6])
#cancer.5.comps.knn$cancer.knn.model # the knn model
cPCA.test.acc <- c(cPCA.test.acc, cancer.5.comps.knn$cancer.knn.acc) # model accuracy

# 6 components
cancer.6.comps.knn <- knnOnPca(pca.comps.can[1:7])
#cancer.6.comps.knn$cancer.knn.model # the knn model
cPCA.test.acc <- c(cPCA.test.acc, cancer.6.comps.knn$cancer.knn.acc) # model accuracy

# create table to show test accuarcies compared to components
can.pca.tab <- data.frame(Components = c(2:6),
                          Accuracy = cPCA.test.acc)
kable(can.pca.tab, caption = 'Test Accuracy of KNN applied to PCA of Cancer dataset')


# creating proximity matrix from digit.dat
digit.dist <- dist(digit.subset)

# performing SMACOF metric MDS
digit.mds <- smacofSym(digit.dist, ndim = 6, type = "ratio")
digit.mds.comps <- as.data.frame(digit.mds$conf)
digit.mds.comps <- cbind.data.frame("labels" = digit.subset$X5, digit.mds.comps)

# run KNN and return on mds component data
# mds model and the KNN accuracies
knnOnMDS <- function(mds.dataset){

  # ensuring the label is a factor type
  mds.dataset$labels <- as.factor(mds.dataset$labels)

  # splitting the data
  mds.index <- createDataPartition(mds.dataset$labels, p = 0.8, list = F)
  mds.train <- mds.dataset[mds.index,]
  mds.test <- mds.dataset[-mds.index,]
```

```
  # performing KNN on the mds data
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  mdsData.knn <- train(labels ~ ., data = mds.train,
                       method = "knn",
                       trControl = trctrl,
                       preProcess = c("center", "scale"))

  # obtaining knn accuracies
  mds.preds <- predict(mdsData.knn, mds.test)
  mds.acc <- sum(mds.preds == mds.test$labels)/nrow(mds.test)

  # create return list
  returnList <- list("trainSet" = mds.train,
                     "knnModel" = mdsData.knn,
                     "modelAcc" = mds.acc)
  return(returnList)
}

mds.dig.testAcc <- c()
# MDS with 2 components
mds.2.comps <- knnOnMDS(digit.mds.comps[,1:3])
#mds.2.comps$knnModel
mds.dig.testAcc <- c(mds.dig.testAcc, mds.2.comps$modelAcc)

# MDS with 3 components
mds.3.comps <- knnOnMDS(digit.mds.comps[,1:4])
# mds.3.comps$knnModel
# mds.3.comps$modelAcc
mds.dig.testAcc <- c(mds.dig.testAcc, mds.3.comps$modelAcc)

# MDS with 4 components
mds.4.comps <- knnOnMDS(digit.mds.comps[,1:5])
# mds.4.comps$knnModel
# mds.4.comps$modelAcc
mds.dig.testAcc <- c(mds.dig.testAcc, mds.4.comps$modelAcc)

# MDS with 5 components
mds.5.comps <- knnOnMDS(digit.mds.comps[,1:6])
# mds.5.comps$knnModel
# mds.5.comps$modelAcc
mds.dig.testAcc <- c(mds.dig.testAcc, mds.5.comps$modelAcc)

# MDS with 6 components
mds.6.comps <- knnOnMDS(digit.mds.comps[,1:7])
# mds.6.comps$knnModel
# mds.6.comps$modelAcc
mds.dig.testAcc <- c(mds.dig.testAcc, mds.6.comps$modelAcc)

# create table to show test accuarcies compared to components
dig.mds.tab <- data.frame(Components = c(2:6),
                          Accuracy = mds.dig.testAcc)
kable(dig.mds.tab, caption = 'Test Accuracy of KNN applied to MDS of digit dataset')
```

```r
#### MDS on cancer data ####
# generating proximity data
cancer.dist <- dist(scale(cancer.dat[,-c(1,2)]))

# performing SMACOF MDS
cancer.mds <- smacofSym(cancer.dist, ndim = 6, type = "ratio")
cancer.mds.comps <- as.data.frame(cancer.mds$conf)
cancer.mds.comps <- cbind.data.frame("diagnosis" = cancer.dat$diagnosis, cancer.mds.comps)

# run KNN and return on mds component data
# mds model and the KNN accuracies
knnOnMDS <- function(mds.dataset){

  # splitting the data
  mds.index <- createDataPartition(mds.dataset$diagnosis, p = 0.8, list = F)
  mds.train <- mds.dataset[mds.index,]
  mds.test <- mds.dataset[-mds.index,]

  # performing KNN on the mds data
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  mdsData.knn <- train(diagnosis ~ ., data = mds.train,
                       method = "knn",
                       trControl = trctrl,
                       preProcess = c("center", "scale"))

  # obtaining knn accuracies
  mds.preds <- predict(mdsData.knn, mds.test)
  mds.acc <- sum(mds.preds == mds.test$diagnosis)/nrow(mds.test)

  # create return list
  returnList <- list("trainSet" = mds.train,
                     "knnModel" = mdsData.knn,
                     "modelAcc" = mds.acc)
  return(returnList)
}

mds.can.testAcc <- c()
# MDS with 2 components
mds.2.comps <- knnOnMDS(cancer.mds.comps[,1:3])
#mds.2.comps$knnModel
mds.can.testAcc <- c(mds.can.testAcc, mds.2.comps$modelAcc)

# MDS with 3 components
mds.3.comps <- knnOnMDS(cancer.mds.comps[,1:4])
#mds.3.comps$knnModel
mds.can.testAcc <- c(mds.can.testAcc, mds.3.comps$modelAcc)

# MDS with 4 components
mds.4.comps <- knnOnMDS(cancer.mds.comps[,1:5])
#mds.4.comps$knnModel
```

```r
mds.can.testAcc <- c(mds.can.testAcc, mds.4.comps$modelAcc)

# MDS with 5 components
mds.5.comps <- knnOnMDS(cancer.mds.comps[,1:6])
#mds.5.comps$knnModel
mds.can.testAcc <- c(mds.can.testAcc, mds.4.comps$modelAcc)

# MDS with 6 components
mds.6.comps <- knnOnMDS(cancer.mds.comps[,1:7])
#mds.6.comps$knnModel
mds.can.testAcc <- c(mds.can.testAcc, mds.5.comps$modelAcc)

# create table to show test accuarcies compared to components
mds.can.tab <- data.frame(Components = c(2:6),
                          Accuracy = mds.can.testAcc)
kable(mds.can.tab, caption = 'Test Accuracy of KNN applied to MDS of cancer dataset')


#### IsoMap with digit data ####

# running IsoMap on the digit data and adding labels to resulting data frame
digit.iso <- Isomap(as.matrix(digit.subset[,-1]), dims = 6, k = 10)
iso.comp.data <- as.data.frame(digit.iso$dim6)
iso.comp.data <- cbind.data.frame(digit.subset$X5, iso.comp.data)
names(iso.comp.data)[1] <- "labels"

# function to run knn on the resulting IsoMapped model
knnOnIso <- function(isoData){

  # splitting the data
  iso.index <- createDataPartition(isoData$labels, p = 0.8, list = F)
  iso.train <- isoData[iso.index,]
  iso.test <- isoData[-iso.index,]

  # knn on the isoMaped data with 2 dimensions
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  isoData.knn <- train(labels ~ ., data = iso.train,
                       method = "knn",
                       trControl = trctrl,
                       preProcess = c("center", "scale"))

  # obtain accuracies
  iso.preds <- predict(isoData.knn, iso.test)
  iso.acc <- sum(iso.preds == iso.test$labels)/nrow(iso.test)

  # creating return items
  returnList <- list("isoTrain" = iso.train,
                     "knnModel" = isoData.knn,
                     "knnAcc" = iso.acc)
}
```

```r
iso.dig.testAcc <- c()

# isoMap to 2 dimensions
iso.2.comps <- knnOnIso(iso.comp.data[,1:3])
#iso.2.comps$isoTrain
#iso.2.comps$knnModel
iso.dig.testAcc <- c(iso.dig.testAcc, iso.2.comps$knnAcc)

# isoMap to 3 dims
iso.3.comps <- knnOnIso(iso.comp.data[,1:4])
# iso.3.comps$isoTrain
# iso.3.comps$knnModel
iso.dig.testAcc <- c(iso.dig.testAcc, iso.3.comps$knnAcc)

# isoMap to 4 dims
iso.4.comps <- knnOnIso(iso.comp.data[,1:5])
# iso.4.comps$isoTrain
# iso.4.comps$knnModel
iso.dig.testAcc <- c(iso.dig.testAcc, iso.4.comps$knnAcc)

# isoMap to 5 dims
iso.5.comps <- knnOnIso(iso.comp.data[,1:6])
# iso.5.comps$isoTrain
# iso.5.comps$knnModel
iso.dig.testAcc <- c(iso.dig.testAcc, iso.5.comps$knnAcc)

# isomap to 6 dims
iso.6.comps <- knnOnIso(iso.comp.data[,1:7])
# iso.6.comps$isoTrain
# iso.6.comps$knnModel
iso.dig.testAcc <- c(iso.dig.testAcc, iso.6.comps$knnAcc)

# create table to show test accuarcies compared to components
iso.dig.tab <- data.frame(Components = c(2:6),
                          Accuracy = iso.dig.testAcc)
kable(iso.dig.tab, caption = 'Test Accuracy of KNN applied to IsoMap of mnist dataset')


# running IsoMap on the cancer data and adding labels to resulting data frame
cancer.iso <- Isomap(as.matrix(scale(cancer.dat[,-c(1,2)])), dims = 6, k = 10)
iso.comp.data <- as.data.frame(cancer.iso$dim6)
iso.comp.data <- cbind.data.frame(cancer.dat$diagnosis, iso.comp.data)
names(iso.comp.data)[1] <- "diagnosis"

# function to run knn on the resulting IsoMapped model
knnOnIso <- function(isoData){

  # splitting the data
  iso.index <- createDataPartition(isoData$diagnosis, p = 0.8, list = F)
  iso.train <- isoData[iso.index,]
  iso.test <- isoData[-iso.index,]

  # knn on the isoMaped data with 2 dimensions
```

```
  trctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 1)
  set.seed(2020)
  isoData.knn <- train(diagnosis ~ ., data = iso.train,
                       method = "knn",
                       trControl = trctrl,
                       preProcess = c("center", "scale"))

  # obtain accuracies
  iso.preds <- predict(isoData.knn, iso.test)
  iso.acc <- sum(iso.preds == iso.test$diagnosis)/nrow(iso.test)

  # creating return items
  returnList <- list("isoTrain" = iso.train,
                     "knnModel" = isoData.knn,
                     "knnAcc" = iso.acc)
}

iso.can.testAcc <- c()
# isoMap to 2 dimensions
iso.2.comps <- knnOnIso(iso.comp.data[,1:3])
#iso.2.comps$isoTrain
iso.can.testAcc <- c(iso.can.testAcc, iso.2.comps$knnAcc)

# isoMap to 3 dimensions
iso.3.comps <- knnOnIso(iso.comp.data[,1:4])
#iso.3.comps$isoTrain
iso.can.testAcc <- c(iso.can.testAcc, iso.3.comps$knnAcc)

# isoMap to 4 dimensions
iso.4.comps <- knnOnIso(iso.comp.data[,1:5])
# iso.4.comps$isoTrain
# iso.4.comps$knnModel
iso.can.testAcc <- c(iso.can.testAcc, iso.4.comps$knnAcc)

# isoMap to 5 dimensions
iso.5.comps <- knnOnIso(iso.comp.data[,1:6])
# iso.5.comps$isoTrain
# iso.5.comps$knnModel
iso.can.testAcc <- c(iso.can.testAcc, iso.5.comps$knnAcc)

# isoMap to 6 dimensions
iso.6.comps <- knnOnIso(iso.comp.data[,1:7])
# iso.6.comps$isoTrain
# iso.6.comps$knnModel
iso.can.testAcc <- c(iso.can.testAcc, iso.6.comps$knnAcc)

# create table to show test accuarcies compared to components
iso.can.tab <- data.frame(Components = c(2:6),
                          Accuracy = iso.can.testAcc)
kable(iso.can.tab, caption = 'Test Accuracy of KNN applied to IsoMap of cancer dataset')
```