UNIVERSITÀ DEGLI STUDI DI MILANO

Module Statistical Learning, Deep Learning and Artificial Intelligence

Final Exam ~ Unsupervised Project

June, 2020 – 2021

Nazli Begum CIRPANLI

942345

Table of Contents

Aim of the Analysis

This analysis aims at uncovering some hidden patterns by applying three different unsupervised learning methods: PCA, K-Means Clustering and Hierarchical clustering

Data Description

Dataset chosen for the analysis is Wisconsin Diagnostic Breast Cancer dataset from https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29

It contains 568 observations and 32 variables without any missing values. 30 features are measurements on cells in suspicious lumps in a woman's breast. Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image.

There are 30 continuous features and a binary target variable. Observations are classified as either *benign* or *malignant*. Continuous features are measurements computed for each cell nucleus.

(columns 1-2)
ID number
Diagnosis (M = malignant, B = benign)
(columns 3-32)
radius
texture
perimeter
area
smoothness
compactness
concavity
concave points
symmetry
fractal dimension

Means and standard deviations of each variable is shown on the right:

	means	stand.dev.
radius_mean	14.12	3.52
texture_mean	19.31	4.29
perimeter_mean	91.91	24.29
area_mean	654.28	351.92
smoothness_mean	0.10	0.01
compactness_mean	0.10	0.05
concavity_mean	0.09	0.08
concave_points_mean	0.05	0.04
symmetry_mean	0.18	0.03
fractal_dimension_mean	0.06	0.01
radius_se	0.40	0.28
texture_se	1.22	0.55
perimeter_se	2.86	2.01
area_se	40.14	45.28
smoothness_se	0.01	0.00
compactness_se	0.03	0.02
concavity_se	0.03	0.03
concave_points_se	0.01	0.01
symmetry_se	0.02	0.01
fractal_dimension_se	0.00	0.00
radius_worst	16.25	4.82
texture_worst	25.69	6.14
perimeter_worst	107.13	33.47
area_worst	878.58	567.85
smoothness_worst	0.13	0.02
compactness_worst	0.25	0.16
concavity_worst	0.27	0.21
concave_points_worst	0.11	0.07
symmetry_worst	0.29	0.06
fractal_dimension_worst	0.08	0.02

Principal Component Analysis

PCA is an unsupervised learning method which serves as an dimension reduction method as well as an exploratory data analysis tool.

Since our dataset is quite high dimensional, it is reasonable and useful to apply PCA and try to reduce dimensionality. Moreover, PCA also gives an idea about whether the data is suitable for clustering.

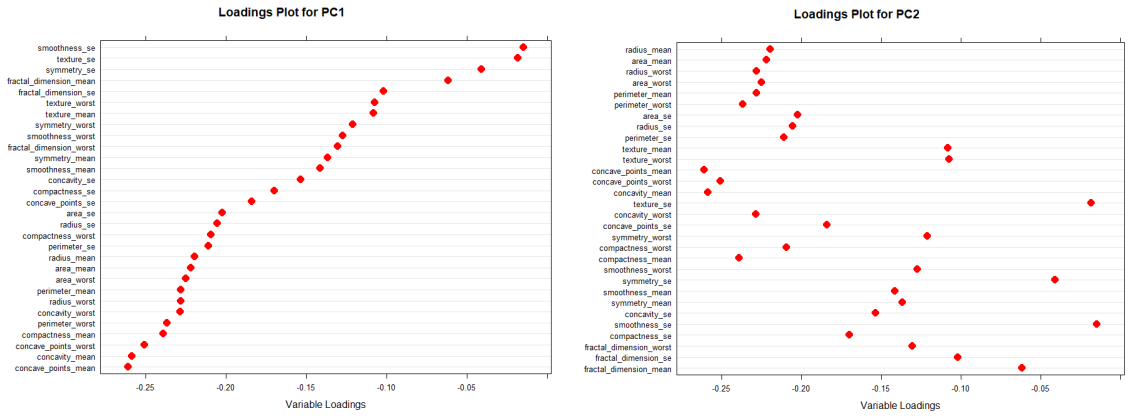
Before applying PCA, data was scaled and standardized. Variables with very large/low variance can dominate others and appear like more important in explaning the variance, when it is not.

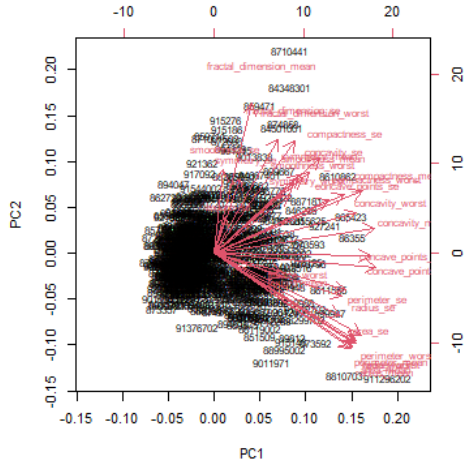
In R, prcomp function was used for principal component analysis. This function uses correlation matrix and the outputs returned by this function are as follow:

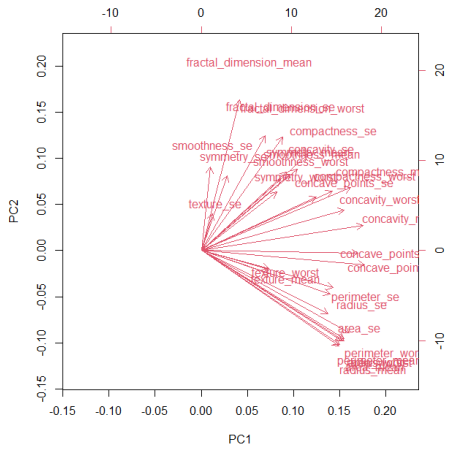
- Sdev - the standard deviations of PCs
- Rotation – loadings (eigenvectors)
- Center - variable mean
- Scale - variable standard deviations
- X – scores

<u>Loadings Plots</u>

Loadings plots for first and second principal component are depicted in the figures below. Red points represents the amount each variable contributes to that component.



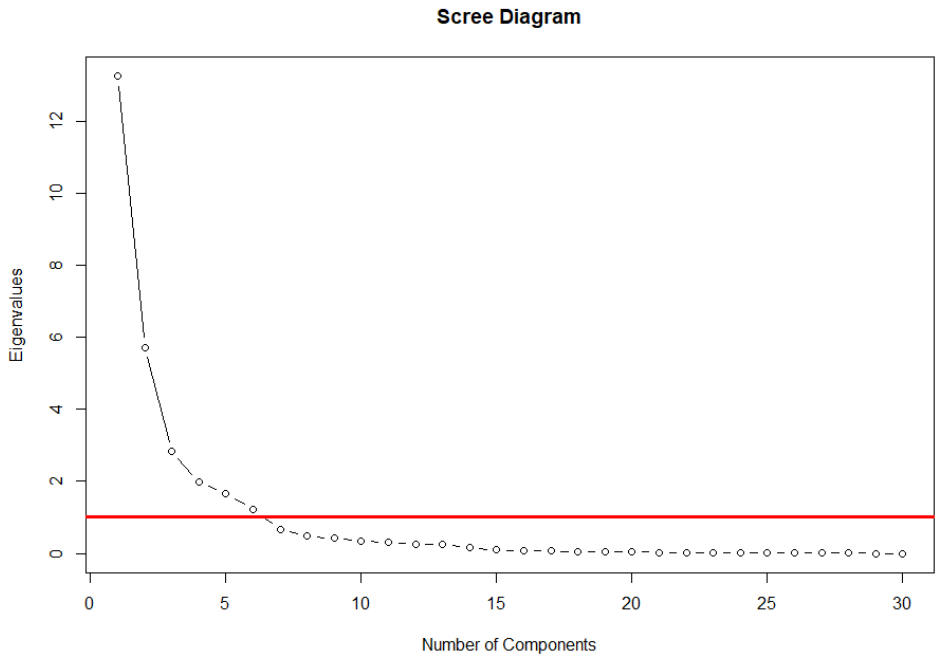


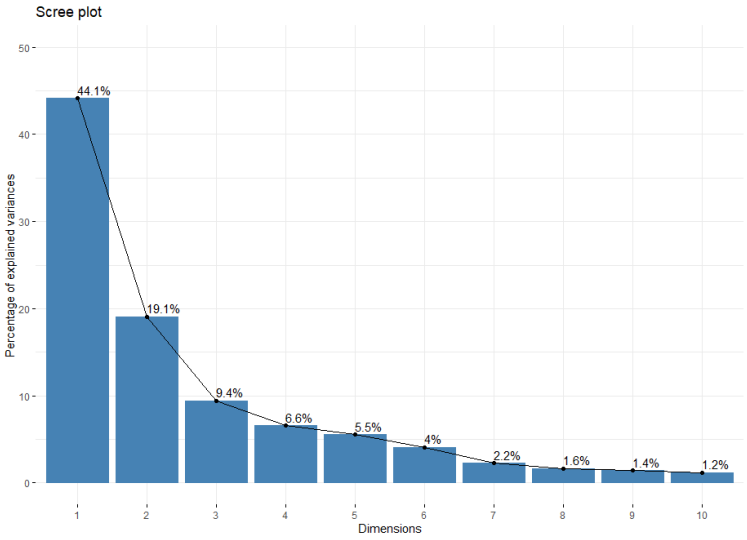


Biplots

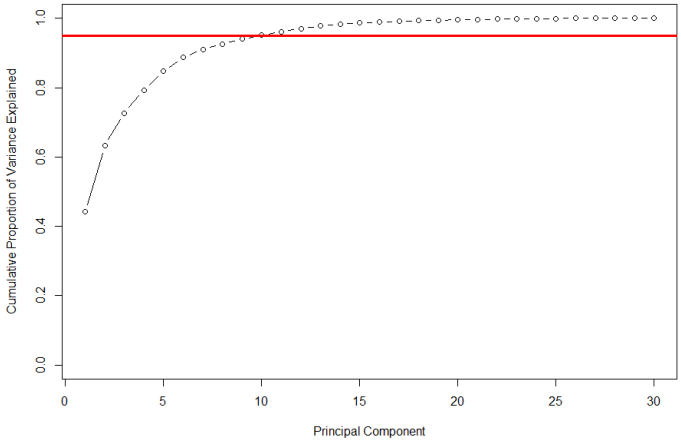
Biplots bring together loadings and scores. Variables are represented by the vectors . Having the same direction means that those variables are similar to each other. Correlation amount between a variable and a component is represented by the length of the corresponding vector. In other words, influence that a vector has on a principal component is large when it is further away that PC's origin.

Eigenvalues represent the variance explained by each principal component and are obtained by squaring the standard deviations of principal components. Standard deviations can be reached with $sd of prcomp function. Eigenvalues greater than 1 were selected and threshold was shown by the red line on the scree diagram below.

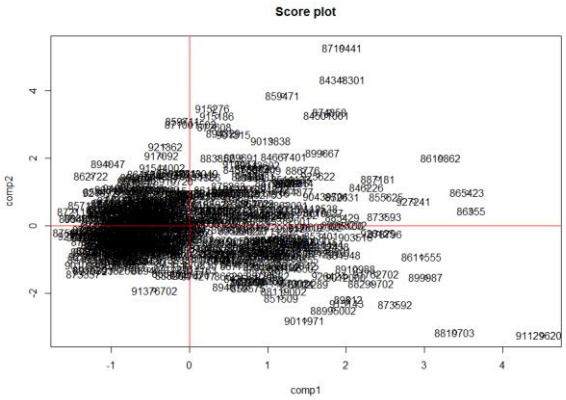


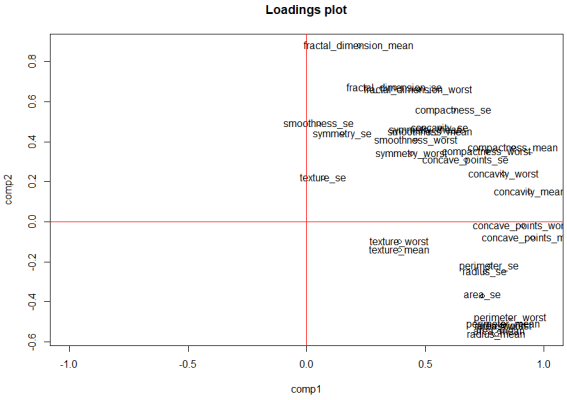


PVE Explained by Each Component



Cumulative Variance Explained



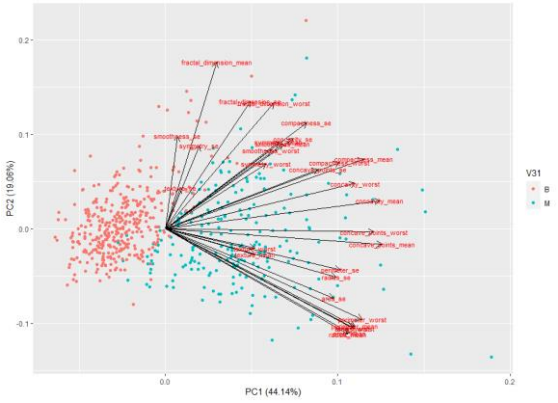


Score Plot **Loadings Plot**

First 7 components explain 95% of the total variance

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	comunality	communality
radius_mean	0.80	-0.56	0.02	-0.06	0.05	-0.02	0.10	0.9705	0.9705
texture_mean	0.39	-0.14	-0.10	0.84	-0.05	0.04	-0.01	0.8915	0.8915
perimeter_mean	0.83	-0.51	0.02	-0.06	0.05	-0.02	0.09	0.9640	0.9640
area_mean	0.81	-0.55	-0.05	-0.08	0.01	0.00	0.04	0.9692	0.9692
smoothness_mean	0.52	0.45	0.17	-0.23	-0.48	0.31	0.11	0.8933	0.8933
compactness_mean	0.87	0.37	0.12	-0.05	0.01	0.02	-0.02	0.9116	0.9116
concavity_mean	0.94	0.15	0.00	-0.03	0.11	0.01	0.09	0.9273	0.9273
concave_points_mean	0.95	-0.08	0.04	-0.09	-0.06	0.06	0.13	0.9427	0.9427
symmetry_mean	0.50	0.46	0.06	-0.09	-0.40	-0.40	0.07	0.7982	0.7982
fractal_dimension_mean	0.22	0.88	0.03	-0.07	-0.06	0.14	-0.24	0.9094	0.9094
radius_se	0.75	-0.25	-0.46	-0.13	-0.20	0.03	-0.26	0.9620	0.9620
texture_se	0.07	0.22	-0.62	0.52	-0.23	0.04	0.09	0.7707	0.7707
perimeter_se	0.77	-0.22	-0.45	-0.11	-0.16	0.00	-0.26	0.9491	0.9491
area_se	0.74	-0.37	-0.37	-0.14	-0.16	0.05	-0.29	0.9532	0.9532
smoothness_se	0.05	0.49	-0.52	-0.06	-0.30	0.37	0.20	0.7835	0.7835
compactness_se	0.62	0.56	-0.26	0.03	0.36	-0.08	-0.02	0.9029	0.9029
concavity_se	0.56	0.47	-0.29	-0.01	0.45	-0.07	0.17	0.8550	0.8550
concave_points_se	0.67	0.31	-0.38	-0.11	0.25	0.03	0.30	0.8549	0.8549
symmetry_se	0.15	0.44	-0.49	-0.04	-0.32	-0.54	0.06	0.8554	0.8554
fractal_dimension_se	0.37	0.67	-0.36	-0.03	0.34	0.06	-0.16	0.8611	0.8611
radius_worst	0.83	-0.52	0.08	-0.02	-0.01	0.00	0.01	0.9663	0.9663
texture_worst	0.39	-0.10	0.08	0.89	-0.10	0.06	0.00	0.9742	0.9742
perimeter_worst	0.86	-0.48	0.08	-0.02	0.01	-0.01	0.00	0.9770	0.9770
area_worst	0.82	-0.52	0.02	-0.04	-0.04	0.03	-0.06	0.9509	0.9509
smoothness_worst	0.46	0.41	0.44	-0.03	-0.42	0.40	0.09	0.9187	0.9187
compactness_worst	0.76	0.35	0.40	0.12	0.16	-0.05	-0.11	0.9147	0.9147
concavity_worst	0.83	0.24	0.29	0.09	0.24	-0.03	0.05	0.8997	0.8997
concave_points_worst	0.91	-0.02	0.29	-0.02	0.05	0.03	0.14	0.9360	0.9360
symmetry_worst	0.44	0.34	0.46	0.06	-0.31	-0.55	0.01	0.9231	0.9231
fractal_dimension_worst	0.47	0.66	0.39	0.10	0.12	0.09	-0.31	0.9372	0.9372

Here it can be seen that the data is separable by first 2 principal components. PC1 explains 44.14 % of the total variation in the data, whereas, PC2 accounts for 19%.



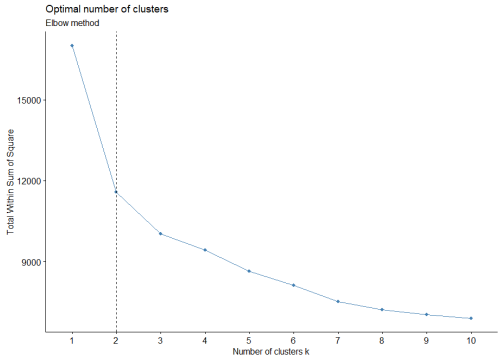
K-Means Clustering

It has been already confirmed by PCA algorithm that the data is separable. Therefore, in order to further exploit this discovery K-means clustering was performed. K-means is a simple and effective clustering algorithm with one drawback which is the necessity to specify the number of clusters beforehand. Once the number of clusters **k** has been decided, algorithm assigns each data points to one of those clusters using Euclidean distance.

There are several techniques which give idea about the optimal number of clusters for data at hand. Of those techniques Elbow method and Average Silhouette method were employed in order to decide the number of clusters.

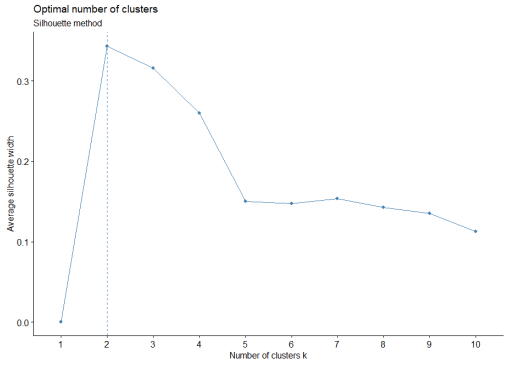
Elbow Method

Elbow method looks at total within-cluster sum of squares for different numbers of clusters and chooses the number after which adding another cluster does not improve the total within-cluster sum of squares. In other words, within-cluster variation is minimized. In this analysis, total wss was compared for varying numbers of clusters within the range (0,10). The number of clusters chosen by Elbow method for this data is 2.



Average Silhouette method

Another method employed to decide on the number of clusters is average Silhouette method. It measures how much a point is similar to its own cluster compared to other clusters. Unlike elbow method, a large number is an indicator of good clustering with this method. In our case, average silhouette is at its maximum when the number of clusters is 2.



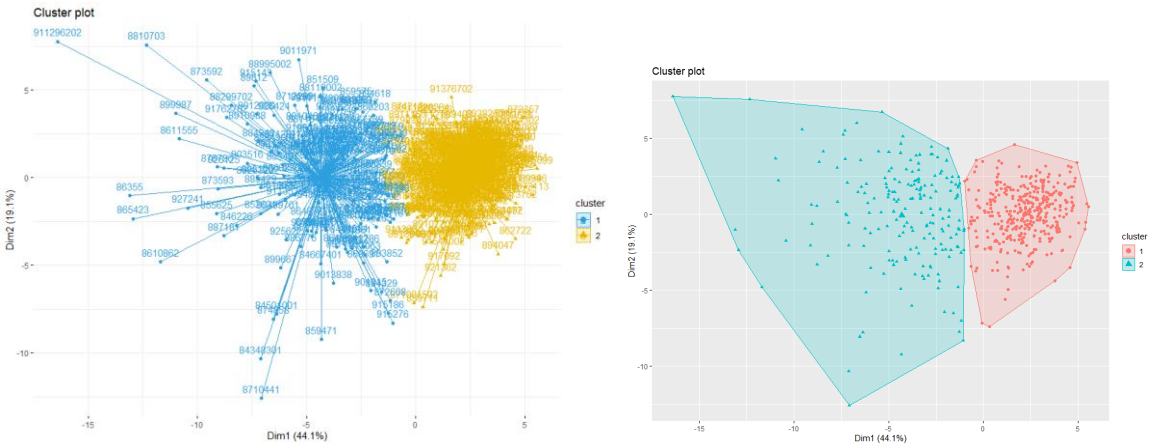
K-Means Algorithm

K-means clustering algorithm works as follows:

Step 1: Randomly assign each data point to the clusters from 1 to K.
Step 2: For each cluster k cluster centroids are calculated. Centroids are the vectors containing the means of the observations for each feature.
Step 3: data points then are assigned to the cluster whose centroid is the closest to that data point in terms of Euclidean distance.
Step 4: Step 2 and 3 are repeated until cluster assignments stop changing.

Elbow method and average Silhoutte method both pointed out that 2 is the optimal number of clusters for this data and should be selected as **k** for k means algorithm. Hence, k –means algorithm was performed with k=2

According to the results of two methods described above, it seems like 2 clusters should be selected for k means algorithm. You can see the visualizations of k-means algorithm run with k=2 and nstart=100 below. with nstart=100, the algorithm go over the steps described above for 100 times. This is to make the results of the algorithm more stable.



Data was clustered into 2 different groups.

Hierarchical Clustering

There are two types of hierarchical clustering: agglomerative (bottom-up) and divisive (top-down). In this analysis, agglomerative approach was performed.

Hierarchical clustering is another method for clustering data. Main advantage of this method is that the number of clusters do not have to be given the algorithm apriori. Outcome of this algorithm is a tree , called dendrogram, consisting of leaves and branches. Leaves correspond to every single data points. Similar data points , leaves, fuse into branches and similar branches fuse until there no leaves or branches left to fuse. Height of the tree represents how similar/dissimilar the data points are to each other. Therefore, data points that fuse at the lower part of the tree are very similar to each other.

Hierarchical Clustering Algorithm

Step 1: Treat each data point as if they are themselves clusters, and then compute pairwise dissimilarities.
Step 2: Fuse the pairs(clusters) that are most similar
Step 3: Compute the new pairwise dissimilarities between clusters and fuse the most similar ones until the algorithm reaches the root node.

Hierarchical clustering was applied to reduced data which was produced by PCA. Distances between data points were calculated using Euclidean distance.

There are several types of linkage that determine which clusters should be fused. In this analysis complete, average and ward linkage were employed.

Cutting the dendrogram into different groups

Dendrogram produce by hierarchical clustering with ward linkage was cut into first 2 and then, 4 groups. Numbers of observations in each cluster were examined and it is clear that clusters represent the different groups of patients. Cutting the tree into 2 groups reveals that patients who have been diagnosed with M belong to 1st cluster and patients who have been diagnosed as B belong to 2nd cluster.

The tree was also cut into 4 groups, however, 2 clusters represent the data in a much better and useful way, than 4 clusters. When compared to the clusters produced by 2-means clustering algorithm , it can be seen that the clusters obtained by cutting the tree into 2 are consistent with 2-means results.

Cut tree into 2 groups

	Diagnosis	
grp	B	M
1	18	169
2	339	42

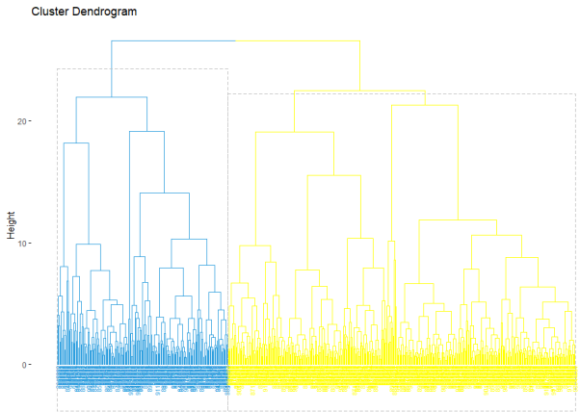
Cut tree into 4 groups

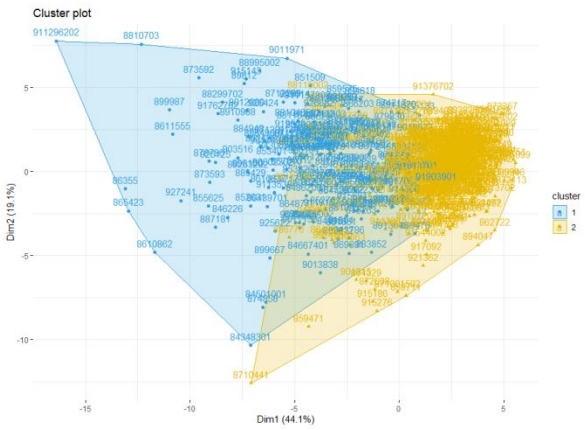
	Diagnosis	
grp	B	M
1	8	101
2	10	68
3	141	33
4	198	9

Comparison with 2-means

grp		
	1	2
1	160	33
2	27	348

As it can be seen below figures that the data can be well-separated by 2 clusters.





Validation of Clusters

In order to understand how well these clustering methods partitioned the data, cluster numbers assigned to each data points were checked according to diagnosis label. Number of patients belong to each cluster appeared to be meaningful when examining those numbers based on diagnosis class; hence it has been clear that the data was clustered well enough to reflect these classes. Additionally, mean of all observations whose true labels and cluster numbers are equal is 0.90 for 2-means and 0.89 for hierarchical clustering with Ward linkage.

Conclusion

- ✓ Wisconsin diagnostic breast cancer data can be reduced to 7 principal components which explain .95 % of the total variance.
- ✓ PCA proves that this dataset is separable and suitable for clustering.
- ✓ 2-means and hierarchical clustering divides the data into 2 clusters in a way that each cluster represents one diagnosis class in the data.
- ✓ Applying k-means clustering to reduced data did not separate the observations as well as clustering applied to the original data. Hence, reduced data was only used with hierarchical clustering for visualization purposes.

Appendix : R Code

```
# loading the libraries
library(car)
library(ggplot2)
library(tidyverse)
library(hrbrthemes)
library(dplyr)
library(tidyr)
library(viridis)
library(gplots)


data=read.csv("Wdbc.data")
fix(data)
dim(data)

# Data Preprocessing

#setting the colnames
names=c('id', 'diagnosis', 'radius_mean',
  'texture_mean', 'perimeter_mean', 'area_mean',
  'smoothness_mean', 'compactness_mean',
  'concavity_mean','concave_points_mean',
  'symmetry_mean', 'fractal_dimension_mean',
  'radius_se', 'texture_se', 'perimeter_se',
  'area_se', 'smoothness_se', 'compactness_se',
  'concavity_se', 'concave_points_se',
  'symmetry_se', 'fractal_dimension_se',
  'radius_worst', 'texture_worst',
  'perimeter_worst', 'area_worst',
  'smoothness_worst', 'compactness_worst',
  'concavity_worst', 'concave_points_worst',
  'symmetry_worst', 'fractal_dimension_worst')

colnames(data)=names
rownames(data)=data$id
data=data[,-1]
fix(data)

#inspection of missing values in the dataset
sum(is.na(data)) # number of records with N/A values


########  Descriptives ##########
#descriptives:
means=apply(data[,-1],2,mean)
stand.dev.=apply(data[,-1],2,sd)
descriptives=round(cbind(means,stand.dev.),2)
descriptives
summary(data)
```

```
#correlations # check if the variables correlated
library(GGally)
ggcorr(data, method = c("everything", "pearson"),
     label_alpha= TRUE,
     label = TRUE, label_size = 2, layout.exp= 0)

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

# Principal Component Analysis


wdbc=data[,-c(1)] # dropping the categorical variables
wdbc= scale(wdbc)  # To standarize the variables
summary(wdbc)


wdbc_pc=prcomp(wdbc,scale. = TRUE,center=TRUE)
summary(wdbc_pc)

wdbc_pc$center #means
wdbc_pc$scale #sd

round(wdbc_pc$rotation,2) # loadings
round(wdbc_pc$x,4) # scores


############################
library(lattice)
# see which variable contributes to PC1 the most
load    <- wdbc_pc$rotation
sorted.loadings <- load[order(load[, 1]), 1]
myTitle <- "Loadings Plot for PC1"
myXlab  <- "Variable Loadings"
dotplot(sorted.loadings, main=myTitle, xlab=myXlab, cex=1.5, col="red")

sorted.loadings <- load[order(load[, 2]), 1]
myTitle <- "Loadings Plot for PC2"
dotplot(sorted.loadings, main=myTitle, xlab=myXlab, cex=1.5, col="red")

sorted.loadings <- load[order(load[, 3]), 1]
myTitle <- "Loadings Plot for PC3"
dotplot(sorted.loadings, main=myTitle, xlab=myXlab, cex=1.5, col="red")

#  BiPlot
cex.before <- par("cex")
par(cex = 0.7)
biplot(wdbc_pc)
par(cex = cex.before)

# Change the direction
wdbc_pc$rotation=-wdbc_pc$rotation
wdbc_pc$x=-wdbc_pc$x
```

```
cex.before <- par("cex")
par(cex = 0.7)
biplot(wdbc_pc)
par(cex = cex.before)

#####################
plot(wdbc_pc)
#The variance explained by each principal component is obtained by squaring
#these:
pr.var=wdbc_pc$sdev^2
round(pr.var,4)
#proportion of variance explained by each principal component,
pve=pr.var/sum(pr.var)
round(pve,2)

plot(pr.var,main="Scree Diagram",xlab = "Number of Components",
   ylab="Eigenvalues",
   type = 'b')
abline(h=1, lwd=3, col="red")

#plot the PVE explained by each component
plot(pve,xlab = "Principal Component",ylab="Proportion of Variance Explained",ylim = c(0,1),
   type = 'b')
abline(h=0, lwd=3, col="red")
#cumulative
plot(cumsum(pve),xlab = "Principal Component",ylab="Cumulative Proportion of Variance
Explained",ylim = c(0,1),
   type = 'b')
abline(h=0.95, lwd=3, col="red")


biplot(wdbc_pc, xlabs = rep("", nrow(wdbc))) # to make it easier to show the vectors

#select how many components
screeplot(wdbc_pc)
pca_var <- wdbc_pc$sdev^2
pca_var_perc <- round(pca_var/sum(pca_var) * 100, 1)
barplot(pca_var_perc, main = "Variation Plot", xlab = "PCs",
      ylab = "Percentage Variance", ylim = c(0, 100))

library("factoextra")
fviz_screeplot(wdbc_pc, addlabels = TRUE, ylim = c(0, 50))

#first 7 components explain 95% of the total variance
#components
components=round(cbind(wdbc_pc$rotation[,1]*wdbc_pc$sd[1],wdbc_pc$rotation[,2]*wdbc_pc$sd
[2],
                wdbc_pc$rotation[,3]*wdbc_pc$sd[3],wdbc_pc$rotation[,4]*wdbc_pc$sd[4],
                wdbc_pc$rotation[,5]*wdbc_pc$sd[5],wdbc_pc$rotation[,6]*wdbc_pc$sd[6],
                wdbc_pc$rotation[,7]*wdbc_pc$sd[7])
            ,2)
colnames(components)=c("PC1","PC2","PC3","PC4","PC5","PC6","PC7")
```

```
communality<-components[,1]^2+components[,2]^2+components[,3]^2+
  components[,4]^2 + components[,5]^2 + components[,6]^2+components[,7]^2

components<-cbind(components,communality)
components

# standardized scores


sd <- wdbc_pc$sdev
scores<-round(cbind(wdbc_pc$x[,1]/sd[1],wdbc_pc$x[,2]/sd[2],wdbc_pc$x[,3]/sd[3],
                wdbc_pc$x[,4]/sd[4],wdbc_pc$x[,5]/sd[5],wdbc_pc$x[,6]/sd[6],
                wdbc_pc$x[,7]/sd[7]),2)
scores
plot(scores, main="Score plot",
    xlab="comp1",ylab="comp2")
text(scores, rownames(wdbc))
abline(v=0,h=0,col="red")
colnames(scores)=c("PC1","PC2","PC3","PC4","PC5","PC6","PC7")
scores

# loadings
par(mfrow=c(1,1))
plot(components[,1:2], main="Loadings plot",
    xlab="comp1",ylab="comp2", xlim=range(-1,1))
text(components, rownames(components))
abline(v=0,h=0,col="red")

plot(components[,2:3], main="Loadings plot",
    xlab="comp2",ylab="comp3", xlim=range(-1,1))
text(components, rownames(components))
abline(v=0,h=0,col="red")

plot(components[,1:3], main="Loadings plot",
    xlab="comp1",ylab="comp3", xlim=range(-1,1))
text(components, rownames(components))
abline(v=0,h=0,col="red")


#### different plottings
diagnosis <- factor(data$diagnosis)

pca_df <- as_tibble(wdbc_pc$x)
ggplot(pca_df, aes(x = PC1, y = PC2, col = data$diagnosis)) + geom_point()

library(ggfortify)
wdbc1=as.data.frame(cbind(wdbc,data$diagnosis),)
colnames(wdbc1['V31'])='diagnosis'

autoplot(wdbc_pc, data = wdbc1, colour ="V31", loadings = FALSE,loadings.label = TRUE,
       loadings.label.size = 3, loadings.colour="black")
```

```
##########

# k-means approach
wbdc=wdbc
summary(wbdc)
# K-Means
set.seed(123)
k.means.fit <- kmeans(wbdc, 2,nstart = 100) # k = 2
print(k.means.fit)
table(k.means.fit$cluster)

#cluster means
aggregate(wbdc, by=list(cluster=k.means.fit$cluster), mean)

#visualisation
library(factoextra)
fviz_cluster(k.means.fit, data = wbdc,
          palette = c("#2E9FDF", "#E7B800"),
          ellipse.type = "euclid",
          star.plot = TRUE,
          repel = FALSE,
          ggtheme = theme_minimal())

# how many clusters ?
## for K-means

summary(wdbc)
wssplot <- function(data, nc=10, seed=1234){
  wss <- (nrow(data)-1)*sum(apply(data,2,var))
  for (i in 2:nc){
    set.seed(seed)
    wss[i] <- sum(kmeans(data, centers=i)$withinss)}
  plot(1:nc, wss, type="b", xlab="Number of Clusters",
       ylab="Within groups sum of squares")}

wssplot(wbdc, nc=10)

library(factoextra)
fviz_nbclust(wbdc, kmeans, method = "wss") +
  geom_vline(xintercept = 2, linetype = 2)+
  labs(subtitle = "Elbow method")
fviz_cluster(kmeans(wbdc, centers = 2), geom = "point", data = wbdc)

# Elbow method
fviz_nbclust(wdbc, kmeans, method ="wss") +
  geom_vline(xintercept = 2, linetype = 2)+
  labs(subtitle = "Elbow method")
# Silhouette method
fviz_nbclust(wdbc, kmeans, method = "silhouette")+
  labs(subtitle = "Silhouette method")

##############################################
```

```
# hierarchical clustering
#As part of the preparation for hierarchical clustering, the distance between all pairs
# of observations are computed. Furthermore, there are different ways to link clusters
#together, with single, complete, and average being the most common linkage methods.
summary(wbdc)
#agglomerative clustering
d <- dist(scores, method = "euclidean") # Euclidean distance matrix.
d_matrix=as.matrix(d)
d_matrix
#complete linkage
H.fit <- hclust(d, method="complete")
fviz_dend(H.fit, cex = 0.5)
# Average linkage
H.fit.avg=hclust(d, method="average")
fviz_dend(H.fit.avg)
# Ward linkage
H.fit.ward=hclust(d, method="ward.D2")
fviz_dend(H.fit.ward)

# Cut tree into 2/4 groups
grp <- cutree(H.fit.ward, k = 2)
head(grp, n = 2)
# Number of members in each cluster
table(grp,diagnosis)
table(k.means.fit$cluster, grp)
fviz_dend(H.fit.ward, k = 2,
        cex = 0.5,
        k_colors = c("#2E9FDF", "yellow","red","pink"),
        color_labels_by_k = TRUE,
        rect = TRUE )
fviz_cluster(list(data = wbdc, cluster = grp),
          palette = c("#2E9FDF", "#E7B800", "#FC4E07"),
          ellipse.type = "convex",
          repel = FALSE,
          show.clust.cent = FALSE, ggtheme = theme_minimal())
############################################################
## validation
km2 = data.frame(wbdc, k.means.fit$cluster,data$diagnosis)
km2$data.diagnosis <- as.factor(mapvalues(km2$data.diagnosis,
                                  from=c("B", "M"),
                                  to=c("2","1")))
mean(km2$k.means.fit.cluster == km2$data.diagnosis)
hc2=data.frame(wbdc, grp,data$diagnosis)
hc2$data.diagnosis <- as.factor(mapvalues(hc2$data.diagnosis,
                                from=c("B", "M"),
                                to=c("2","1")))
mean(hc2$grp == hc2$data.diagnosis)
############################################################
```