# Diabetes data analysis- clustering in R

## Data Description:

Data collected from diabetes patients has been widely investigated nowadays. Data set is aggregated, labeled and relatively straightforward to do further machine learning tasks. However, in the real world, diabetes data are often collected from healthcare instruments attached to patients. The raw data can be sporadic and messy.
The data is from the UCI archive. It is collected from electronic recording devices as well as paper records for 70 diabetes patients. For each patient, there is a file that contains 3-4 months of glucose level measurements and insulin dosages, as well as other special events (exercise, meal consumption, etc).

## Dataset information:

Diabetes patient records were obtained from two sources: an automatic electronic recording device and paper records. The automatic device had an internal clock to timestamp events, whereas the paper records only provided "logical time" slots (breakfast, lunch, dinner, bedtime). For paper records, fixed times were assigned to breakfast (08:00), lunch (12:00), dinner (18:00), and bedtime (22:00). Thus paper records have fictitious uniform recording times whereas electronic records have more realistic time stamps.
File Names and format:
(1) Date in MM-DD-YYYY format
(2) Time in XX:YY format
(3) Code
(4) Value

The Code field is deciphered as follows:

33 = Regular insulin dose
34 = NPH insulin dose
35 = UltraLente insulin dose
48 = Unspecified blood glucose measurement
57 = Unspecified blood glucose measurement
58 = Pre-breakfast blood glucose measurement
59 = Post-breakfast blood glucose measurement
60 = Pre-lunch blood glucose measurement

61 = Post-lunch blood glucose measurement
62 = Pre-supper blood glucose measurement
63 = Post-supper blood glucose measurement
64 = Pre-snack blood glucose measurement
65 = Hypoglycemic symptoms
66 = Typical meal ingestion
67 = More-than-usual meal ingestion
68 = Less-than-usual meal ingestion
69 = Typical exercise activity
70 = More-than-usual exercise activity
71 = Less-than-usual exercise activity
72 = Unspecified special event

## 1) Cluster analysis:

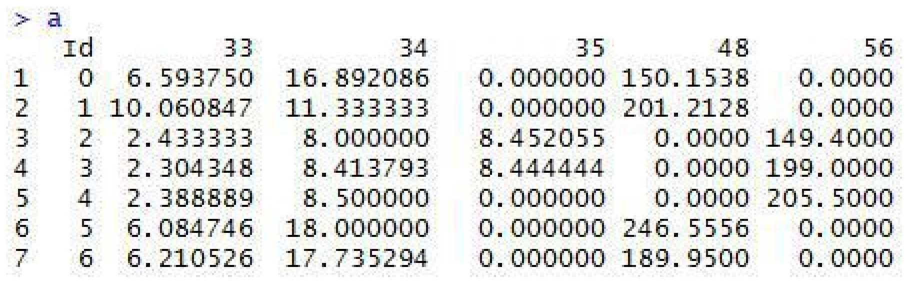
Since the patients may have different levels of symptoms and also vary in treatment (such as insulin dose), we first conduct a clustering analysis to see if there are underlying groups
.

### Data Preparation procedure:

First, we need to construct a data frame from the 70 separate files. In the raw data file, data points are recorded in a 'transaction' style and the time interval is irregular. Also the data set does not have any clear label or indicator. This makes the data at hand difficult to work with. Therefore, we need to preprocess the data for machine learning tasks.

### (1) Aggregation-

For now, we ignore the timestamps and just do an aggregation on the patient level. We calculate the average value for each code and thus each average code value can be used as a feature. For each patient, we combine the information and form a feature vector. Here, we need to do a transpose of the data frame, which means that we want to convert CODE as separate columns.
We can have a look at the obtained data set shown below:



```
> a
  Id        33        34        35       48       56
1  0  6.593750 16.892086  0.000000 150.1538   0.0000
2  1 10.060847 11.333333  0.000000 201.2128   0.0000
3  2  2.433333  8.000000  8.452055   0.0000 149.4000
4  3  2.304348  8.413793  8.444444   0.0000 199.0000
5  4  2.388889  8.500000  0.000000   0.0000 205.5000
6  5  6.084746 18.000000  0.000000 246.5556   0.0000
7  6  6.210526 17.735294  0.000000 189.9500   0.0000
```

For ease of presentation, we omitted other CODE values. In total, there are 20 types of CODE. The next job is to cluster this data. Usually, clustering on data with high dimensionality is not ideal since the distance of each data point tends be large. It would be great if we can do a PCA analysis and cluster on principle components (PC), which can indicate directions of the features that have the highest variations.

## (2) Mean normalization-

Replace each $x_j^i$ with $x_j$ - $\mu_j$, Here there is no need to normalize data.

## (3)Feature scaling (depending on data) -

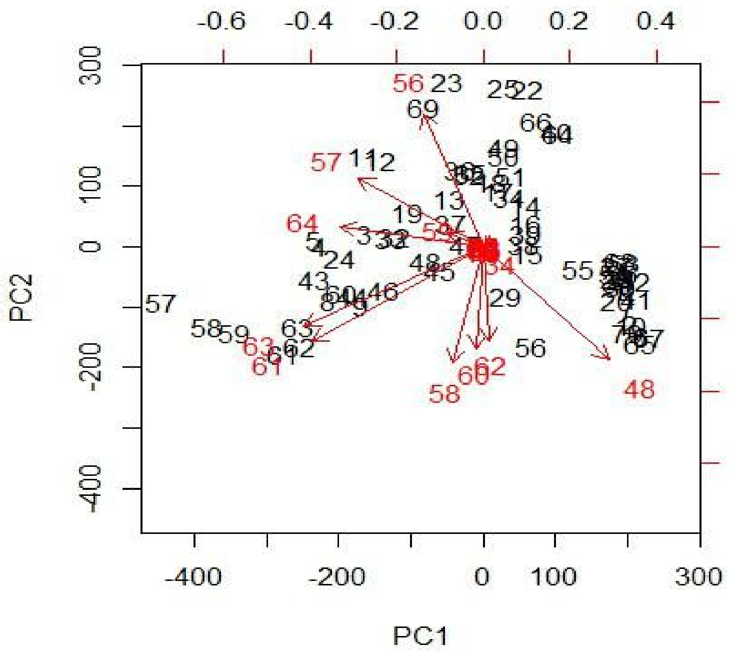
If features have very different scales then scale so they all have a comparable range of values. Standard deviation (more commonly).
Here there is no need to scale a data.

## (4) Dimensionality reduction- Principal Component Analysis

It is a type of unsupervised learning problem. It Speeds up algorithms. It's hard to visualize highly dimensional data.

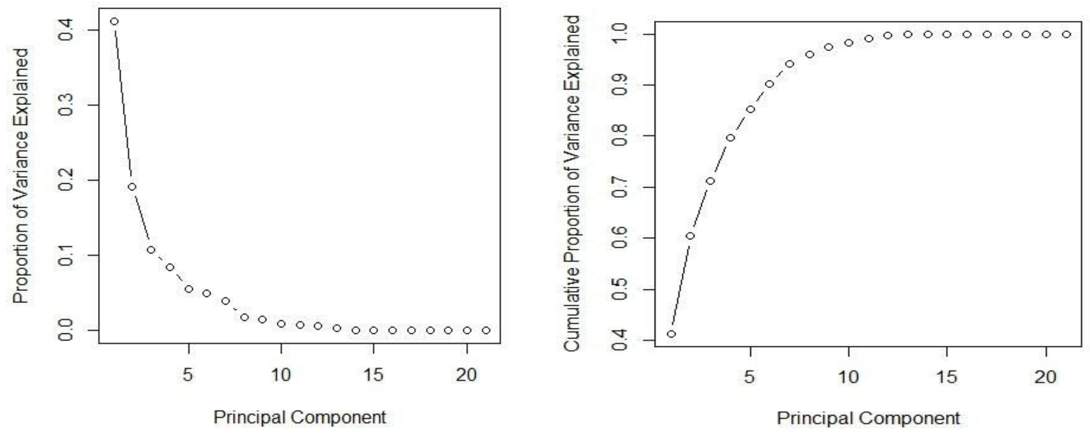
Typically you don't generally ascribe meaning to the new features (so we have to determine what these summary values mean). So despite having 20 features, there may be two "dimensions" of information, with features associated with each of those dimensions. PCA tries to find a lower dimensional surface so the sum of squares onto that surface is minimized.
In our data set have lots of zero values so we can’t replace those values with median hence we do further analysis using the scale=0.

Dimension of pca is:
70*21
Biplot of pca:



The parameter scale = 0 ensures that arrows are scaled to represent the loadings. To make inference from image above, focus on the extreme ends (top, bottom, left, right) of this graph.

## Scree plot and cumulative scree plot:

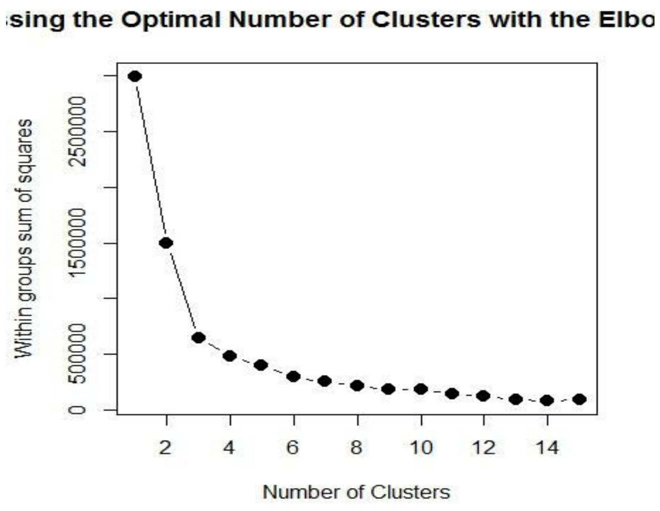


A scree plot is used to access components or factors which explains the most of variability in the data. It represents values in descending order. The plot above shows that ~ 6 components explains around 93.2% variance in the data set. Let's do a confirmation check, by plotting a cumulative variance plot. This will give us a clear picture of number of components. This plot shows that 6 components results in variance close to ~ 93%.

**K Means clustering:**

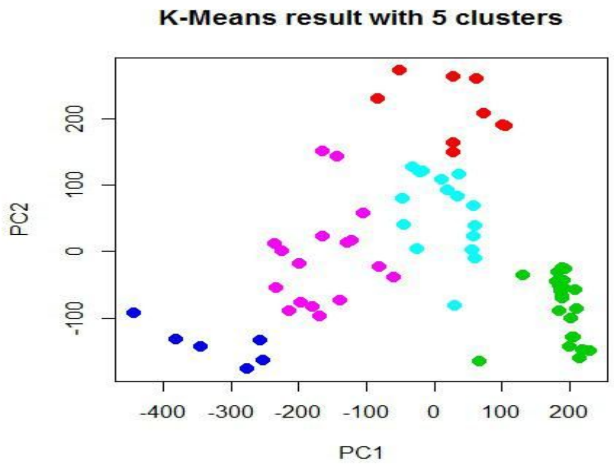
In the principal component space, we use a k-means clustering method to do the clustering over the first two PCs.

We find optimal number of clusters using Elbow plot:



From above graph we can say that there are.

**The clusters are illustrated in the plot:**



## Classification

## Naive bayes:

```
Confusion Matrix and Statistics

          Reference
Prediction 1    2    3
         1       24 0 1
         2       0 21 0
         3       0 0   24

Overall Statistics

               Accuracy : 0.97
                 95% CI : (0.7684, 1)
    No Information Rate : 0.4286
    P-Value [Acc > NIR] : 7.052e-06

                  Kappa : 0.95
 Mcnemar's Test P-Value :    7.052e-16
```

## KNN:

### Confusion matrix:

```
predictions  1  2  3
          1      26  1  0
          2      0 23  0
          3      0  0 20

accuracy:
[1] 0.9857143
```

## R code:

```
#load data
dd=read.csv("D:/shweta/Project_5 Hypoglecemia/dd.csv", header=T)
datetime.vec = paste(dd$Date, dd$Time)
dd$datetime = as.POSIXct(strptime(datetime.vec, "%m-%d-%Y %H:%M"))
str(dd)
head(dd)
dd$Code = as.factor(dd$Code)
#names(dd)
d1=dd[,c(-2:-3,-6)]

#########Aggrigation

library("reshape2")
library(plyr)

q<- ddply(d1, .(Id, Code), summarize, Value=mean(Value))

p=melt(q, id=c("Id", "Code"), Code.name="var")

a=dcast(p, Id~Code, Value.var="Value", na.rm=T)

a[is.na(a)] <- 0.0000


#######data prepration
########pca
pca = prcomp(a[,-1], scale. =F, center = T)
names(pca)
pca$scale
pca$center
pca$rotation
pca$rotation[1:5,1:4]

dim(pca$x)

#plot
biplot(pca, scale = 0)

#compute standard deviation of each principal component
```

```
std_dev <- pca$sdev

#compute variance
pca_var <- std_dev^2

#proportion of variance explained
prop_var <- pca_var/sum(pca_var)

#scree plot
plot(prop_var, xlab = "Principal Component",
    ylab = "Proportion of Variance Explained",
    type = "b")
#cumulative scree plot
plot(cumsum(prop_var), xlab = "Principal Component",
    ylab = "Cumulative Proportion of Variance Explained",
    type = "b")

#add a data with principal components
a = data.frame(Id = a$Id, pca$x)
#we are interested in first 6 PCAs
a = a[,1:7]
dim(a)
```

## #########Cluster method 1

```
# Subset the data
dat = a[,c(2,3)]
# Plot subset data
plot(dat, main = "% of favourable responses to
    Learning and Privilege", pch =20, cex =2)
# Perform K-Means with 2 clusters
set.seed(7)
km = kmeans(dat, 2, nstart=100)
summary(km)
# Plot results
plot(dat, col =(km$cluster +1) , main="K-Means result with 2 clusters", pch=20, cex=2)
# Check for the optimal number of clusters given the data
mydata <- dat
wss <- (nrow(mydata)-1)*sum(apply(mydata,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(mydata,
                                     centers=i)$withinss)
```

```
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares",
     main="Assessing the Optimal Number of Clusters with the Elbow Method",
     pch=20, cex=2)
# Perform K-Means with the optimal number of clusters identified from the Elbow method
set.seed(7)
km2 = kmeans(dat, 6, nstart=100)
# Plot results
plot(dat, col =(km2$cluster +1) , main="K-Means result with 6 clusters", pch=20, cex=2)
```

## #######cluster method 2

```
#K-Means:
library(robustHD)
library(NbClust)
k4=NbClust(dat, method="kmeans", min.nc=2,max.nc=15)
kmeans=kmeans(dat, 3, nstart=100)
kmeans
groups =kmeans$cluster
table(groups)
plot(dat[c("PC1","PC2")],col=kmeans$cluster,pch=c(1,2,3))
```

## #####classification method 1

```
a
outcome=
c(2,2,3,3,3,2,2,3,3,2,1,1,1,1,2,1,1,1,1,2,2,1,1,3,1,2,2,2,2,2,2,3,3,1,1,1,1,1,1,1,2,2,3,3,3,3,1,3,1,1,
1,1,2,2,2,2,3,3,3,3,3,3,3,1,2,1,2,2,1,2)
outcome
data =data.frame(a,outcome)
names(data)
str(data)
```

### #naive bayes

```
n=nrow(data)
s=sample(n,n*0.8,replace<-F,set.seed(78))
train_data<-data[s,]
train_data1<-train_data$outcome
test_data<-data[-s,]
test_data1<-test_data[,-9]
library(e1071)

# Fitting model
```

```
fit <-naiveBayes(train_data1 ~ ., data = train_data)
fit
summary(fit)
#Predict Output
predicted= predict(fit,test_data1)
t =table(test_data$outcome,predicted)
acc<-sum(diag(t))/sum(t)
acc
```

**#######classification using caret package**

**#naive bayes2**

```
# load the packages
library(caret)
library(klaR)

# define an 80%/20% train/test split of the dataset
trainIndex <- createDataPartition(data$outcome, p=0.80, list=FALSE)
dataTrain <- data[ trainIndex,]
dataTest <- data[-trainIndex,]
# train a naive Bayes model
data$outcome = as.factor(data$outcome)
dataTrain$outcome <- as.factor(dataTrain$outcome)
fit <- NaiveBayes(outcome~., data=dataTrain)

# make predictions
predictions <- predict(fit, dataTest[,1:7])
# summarize results
confusionMatrix(predictions$class, dataTest$outcome)
```

**###KNN**

```
# load the packages
library(caret)
library(mlbench)

# fit model
fit <- knn3(outcome~., data=data, k=3)
# summarize the fit
print(fit)
# make predictions
```

```
predictions <- predict(fit, data[,1:8], type="class")
# summarize accuracy
t=table(predictions, data$outcome)
t
acc<-sum(diag(t))/sum(t)
acc
```