McDonald's Nutrition Facts Analysis using Unsupervised Learning

MSc. Data Science and Economics

Tommaso Pessina*
Department of Economics, Management and Quantitative Methods
University of Milan
Milan, Italy
tommaso.pessina@studenti.unimi.it

September 1, 2020

Abstract

The aim of this project is to analyze and discuss the nutrition fact of the McDonald's American menu. We will simply cluster the menu in order to find some interesting result and will see what are the effect of some micro/macro nutrient on calories (in this particular example).
All of this will be done using unsupervised learning statistical methods, like clustering and Principal Component Analysis, and some data visualization to correctly understand our result.

***K*eywords** Statistical Learning · Cluster · Principal Component Analysis · Unsupervised Learning · More

1 Introduction

Firstly, we can say few word about the dataset that we used. It is an official dataset published by McDonald's that can easly find online.
We have in total 24 column that identify: category, item, serving size, calories, calories from fat, Total Fat, Total Fat (% Daily Value), Saturated Fat, Saturated Fat (% Daily Value), Trans Fat, Cholesterol, Cholesterol (% Daily Value), Sodium, Sodium (% Daily Value), Carbohydrates, Carbohydrates (% Daily Value), Dietary Fiber, Dietary Fiber (% Daily Value), Sugars, Protein, Vitamin A (% Daily Value), Vitamin C (% Daily Value), Calcium (% Daily Value), Iron (% Daily Value).
The fist thing that we do, it is to remove the first three column because, for our purpose, are not relevant (although the column "item" will be added again after the clusterization); actually, for the clusterization, these 3 column would rise some problem.
Notice that go into nutritional details (in a biological sense) is not our purpose. Despite that, we can answer to some question anyway, like:

- Can we divide (i.e. clusterize) the McDonald's menu in a good way? Or better, the clusterization method is able to clusterize correctly a menu with respect to some nutritional exigence?
- Can we suggest dishes to people with dietary problems using this method?
- What is the micro or macro nutrient present in the food menu with higher quantity?
- There is any correlation between micro/macro nutrient with respect of each other (especially with calories)?

2 Clusterization method and statistical analysis

As said, the first thing the we can do is to clusterize our dataset, but firstly we may analyze what are (in general) the mean value of each nutritional fact, as shown in Figure 1.

*MSc. Data Science and Economics student | Bsc. Computer Science

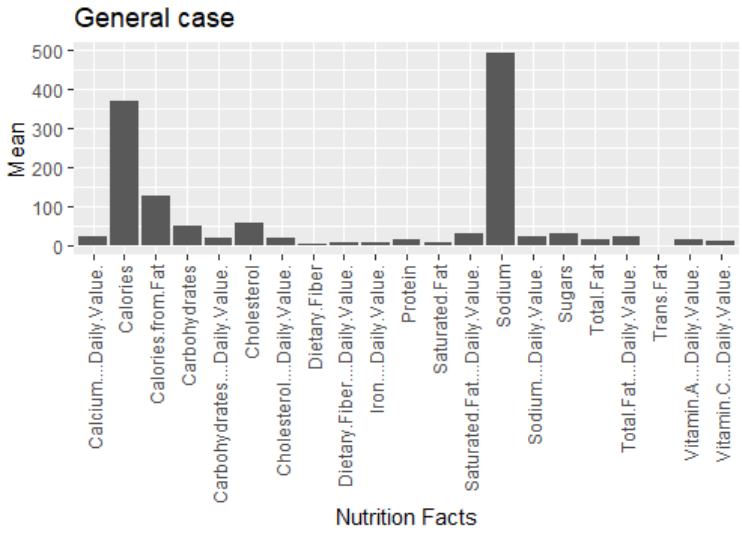

Figure 1: Mean nurtrional facts

Despite the fact that the mean level of Sodium are at a critical value, probably to induce customer to drink more, we can move on and remove the columns that contain textual value in order to compute our clusterization analysis.
The first, and more important, question here is: what is the correct number of cluster? Unfortunately there is not a correct or wrong value, but we can run several time the method that give us the number of cluster that we should use and choose the one that appear, as minimum value, more often. But let's discuss more about this method.
Firstly, we choose to use the k-means method because it can partition our data into n cluster according to the distance between the observation and the mean of a cluster, so it's sound appealing for this kind of analysis (although we may not obtain the "first best" optimal solution).
The problem of the kmeans algorithm is that it is computationally difficult (NP-hard); however, efficient heuristic algorithms converge quickly to a local optimum.
After said that, we still not know how to select the number of cluster. This can be done by running several times the kmeans algorithm, each time with a different number of cluster, and extract the vector of within-cluster sum of squares, one component per cluster (that the R method will provide to us automatically). Now we can sum up all of this value and we will select the number of cluster that gives us the minimum of this latter value. It turns out that 12 is a good number (again, if we run several time the method, we can obtain different value) as we can see in Figure 2.

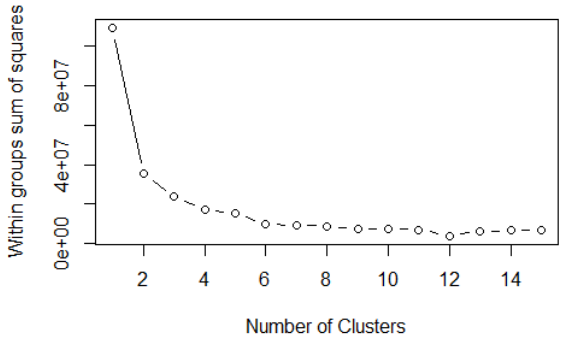

Figure 2: Number of cluster

Now, by running the kmeans method offered by R, we will obtain the cluster. Saving them as a factor in our dataset and adding again the name of the item, we can now analyze for each element the cluster to which it belongs.

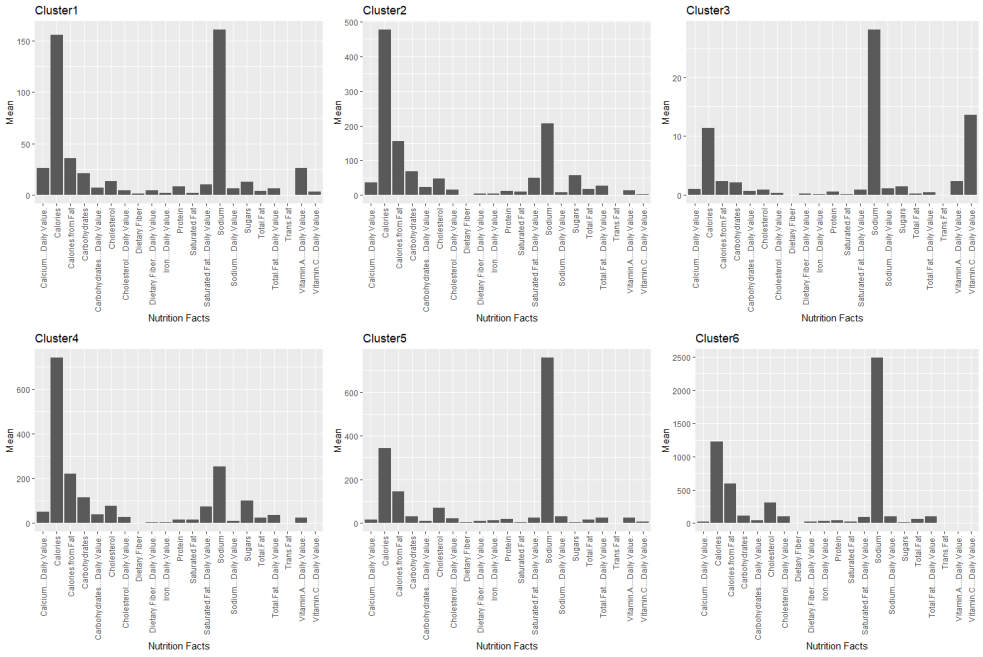

Figure 3: Cluster 1 to 6 plot

2.1 Cluster analysis

Firstly, in Figure 3 (cluster1 to cluster6) and Figure 4 (cluster7 to cluster12) we can see graphically the mean distribution of nutritional facts for each cluster.

Just from this, we can say something, like:

- if a person suffers from hypertension or for other reasons requires a hyposodic diet, he/she should go for an item belonging to cluster 3, cluster 10 or cluster 11 (pay attention to the scale value at the right side of each plot) and he/she should surely avoid food from cluster 6, cluster 9 or cluster 12;
- if you want to stay light you should order an item from cluster 3 but also cluster 10 and cluster 11 would be good. For cluster 1 we should pay attention to the carbohydrates from fat;
- if you have cholesterol problem, you should prefer item from cluster 1, cluster 3, even better from cluster 10 or cluster 11;

But let's go deeper and see if our model can be really used for this purpose. From the above, seems that cluster 3 are the more "healthier" (less sodium, less calories and less cholesterol) so let's see which kind of food it contains:

- Side salad;
- Diet Coke (various serving size);
- Iced Tea (various serving size);
- Coffee (various serving size);

So, basically it does not contains main dishes.
Let's take a counterexample from cluster 9, that should be avoided by a person that requires a hyposodic diet. Here we can find food item like the "Bacon Clubhouse Crispy Chicken Sandwich" that contains a Sodium value of 1720 (i.e. 72 of daily value suggested). Cluster 9 seems to be the worst and in fact it contains all big serving size (e.g. 40 chicken nuggets).

2.2 Hierarchical clustering

Instead of using the k-mean method, we can also clusterize our dataset with the hierarchical clusterization. First we need to calculate the dissimilarity matrix, i.e. a matrix which contains the distance (here we use the Euclidean distance)

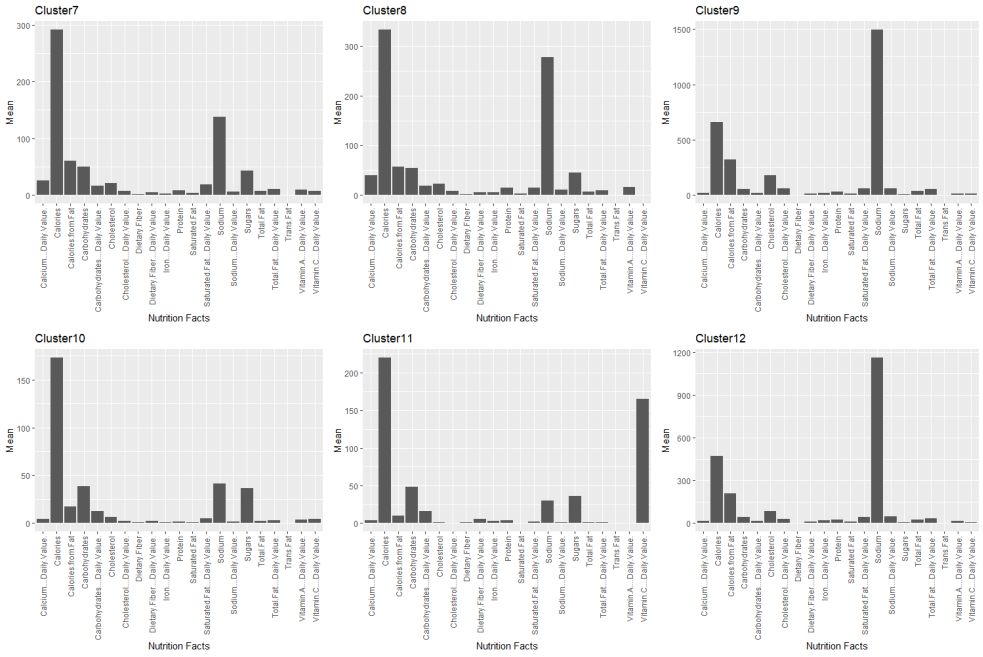

Figure 4: Cluster 7 to 12 plot

between each element. Next we compute the hierarchical clustering using Complete Linkage.
What we obtain here is something not so informative, so we decide to compute a divisive hierarchical clustering of the dataset with the help of the function diana. In particular here we choose to use the Manhattan distance and set the flag "stand" to false, that means that our dataset is not already standardized. We also choose to cut the tree according to the number of cluster that we choosed before: 12.
Finally, a graphical clusterization of our food items can be the one in Figure 5.

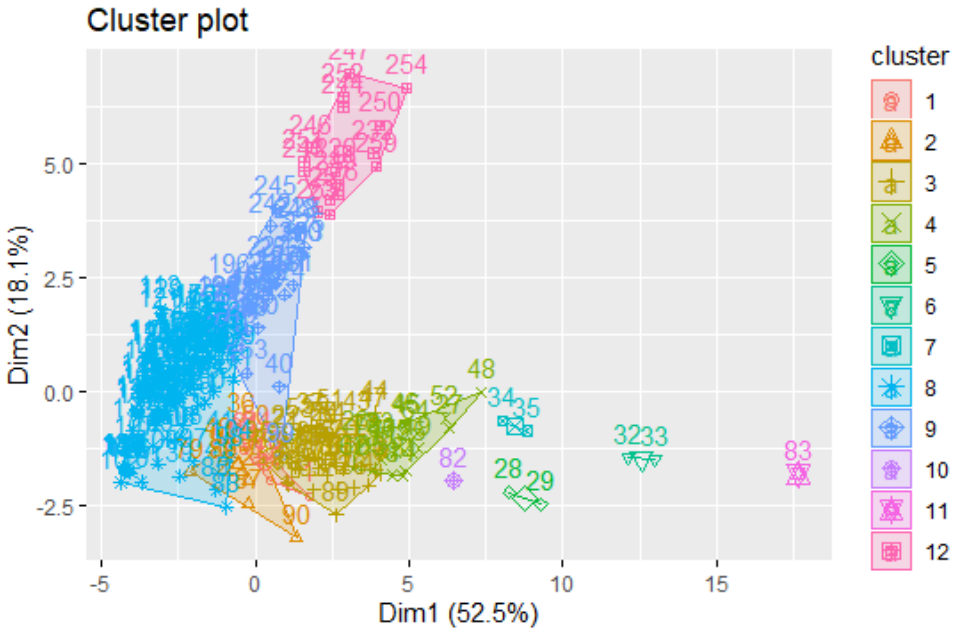

Figure 5: Cluster plot

For completeness we include also the dendrogram of our hierarchical clusterization showing also the cluster in the "leaf", which can be seen in Figure 6.

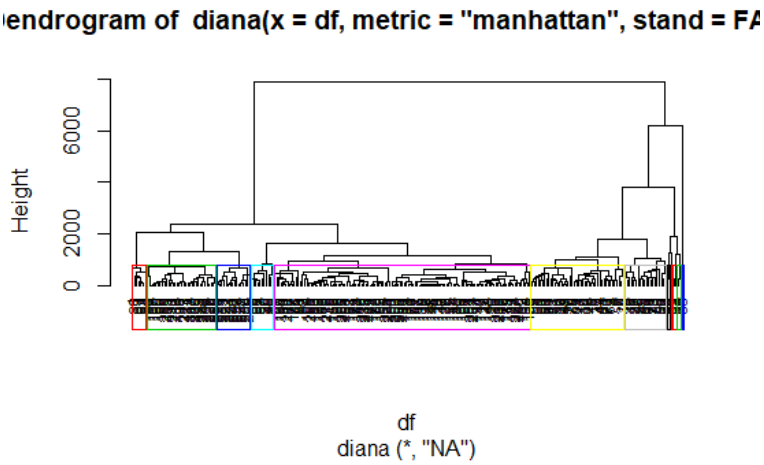

Figure 6: Dendrogram of diana

Now we can explain a little bit the difference between the Euclidean and the Manhattan distance, both used to calculating dissimilarities between observations. The Euclidean distance take the root of the overall sum-of-squares of the differences (of two observation), whereas the Manhattan take the sum of the absolute differences.

2.3 Distances between observation

Finally, we can analyse the distance between few observation to see if the clusterization of the item are correct from a nutritional point-of-view.
Firstly, we can analyze a counterexample, we calculate the Manhattan distance between the Big Mac and the side salad, dishes that are obviously of a different type and they are dived correspondingly as cluster 12 and cluster 3. Turns out that they have a distance of 2193, pretty high. Now, we takes two different dishes: McChicken and Cheeseburger, which have been both classified as cluster 5, and they have a distance of 303.5, not so much.
So, basically, we can use this method to calculate "how far we are" from our "ideal" dish. For instance, if we fix a dish that we know that we can eat (according to our diet for example) we can calculate the distance of all other food item form our "starting dish" and we can select the ones with smaller distance (i.e. that should be similar by a nutritional point of view). Alternatively, this can be easily done by selecting food item from the same cluster or from nearby cluster.

3 Principal Component Analysis (PCA)

Now we can analyze the implication (and maybe correlation) of nutrient on each other for each cluster.
Principal component analysis (PCA) can be useful to analyze deeper the clusters. Here we use the prcomp function which performs a principal components analysis on a given numeric data matrix and returns the results as an object of class prcomp. We choose to set the flag scale to True, that is a logical value indicating whether the variables should be scaled to have unit variance before the analysis takes place.
For our analysis we take a random sample of 23 food items from our dataset and (accordingly to the position) we associate to each food the correspondingly item name.
Firslty, we may say few word about the Principal Component Analysis (PCA) that is basically a technique that allows us to project data onto a lower-dimensional space without losing too much information. PCA can be either done by Singular Value Decomposition (SVD) of a design matrix, what the prcomp function do in background, or by doing 2 steps:

- calculating the data correlation (or covariance) matrix of the original data;
- performing the eigenvalue decomposition.

We also try the princomp function, that use the second method described above, and the unique big difference is that the result (in particular the plot) are reversed (specular).
We have to keep in mind that the second method (the one that uses the eigenvalue decomposition) require a number of observation higher than the number of value (e.g. 20 here would not work).
For each method, after compute the PCA, we can see the percentage of explained variance by each principal component. As we can see in Figure 7, the percentage of explained variance decrease as the number of the component increase.

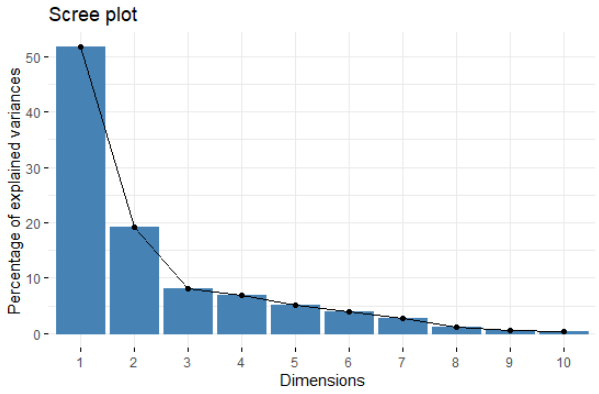

Figure 7: Percentage of explained variance

Now, in Figure 8 we can see the plot of the Principal Component Analysis for 23 random sample dishes from the McDonald's menu.

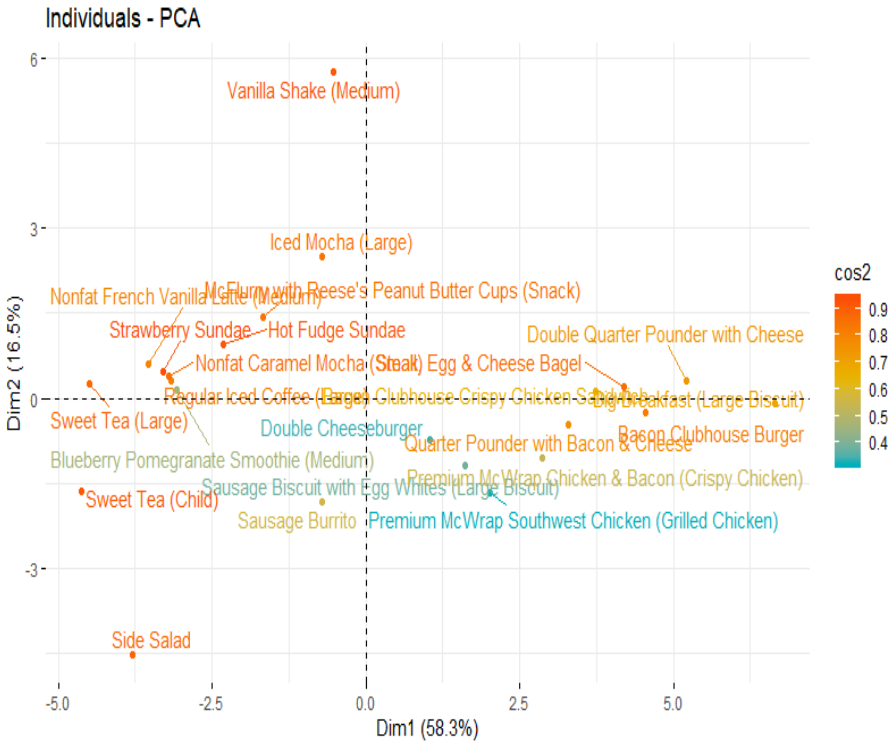

Figure 8: PCA plot

Finally, we can analyse in Figure 9 the correspondence between the random food item and the nutritional facts, always by using PCA.

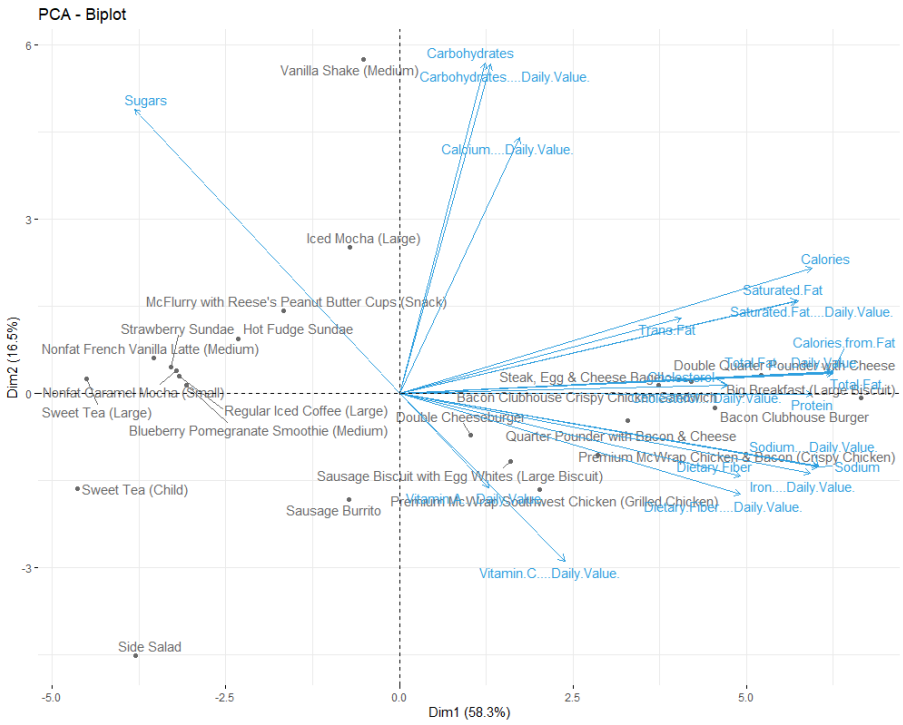

Figure 9: PCA plot

If, instead, we used the eigenvalue decomposition, we need the correlation matrix. This can be useful also to understand some correlation between nutrient. The only one problem is that we must remove the column "Trans fat" because it contain only 0 (so the computation would not works). If we analyse the correlation matrix we can see few interesting things, described in Table 1, in which the number are the correlation value between the row/column item.

Table 1: Correlation matrix

	Carbohydrates	Dietary.Fiber	Sugars	Protein
Calories	0.7815395	0.5388935	0.2595981	0.7878475
Total.Fat	0.4612135	0.5808373	-0.1154457	0.8077730

Moreover we can say that:

- Calories and Total.fat have a correlation value of 90%;
- Carbohydrates and Sugars have a correlation value of 76%;
- Protein/fiber and Sugars are negatively correlated;
- Protein and fiber have a correlation value of 64%;

Finally, in the code appendix we include also a way to compute the PCA with the eigenvalue decomposition without using the princomp function, but since this will lead to similar result we does not discuss this in details.

3.1 PCA prediction

Now we can, basing on our PCA result, predict the classification of new food item like: Big Mac, Diet Coke (medium), Cheeseburger and McFlurry with Oreo Cookies (medium). This will be done by using the function predict of Principal Component Analysis and the result can viewed in Figure 10.

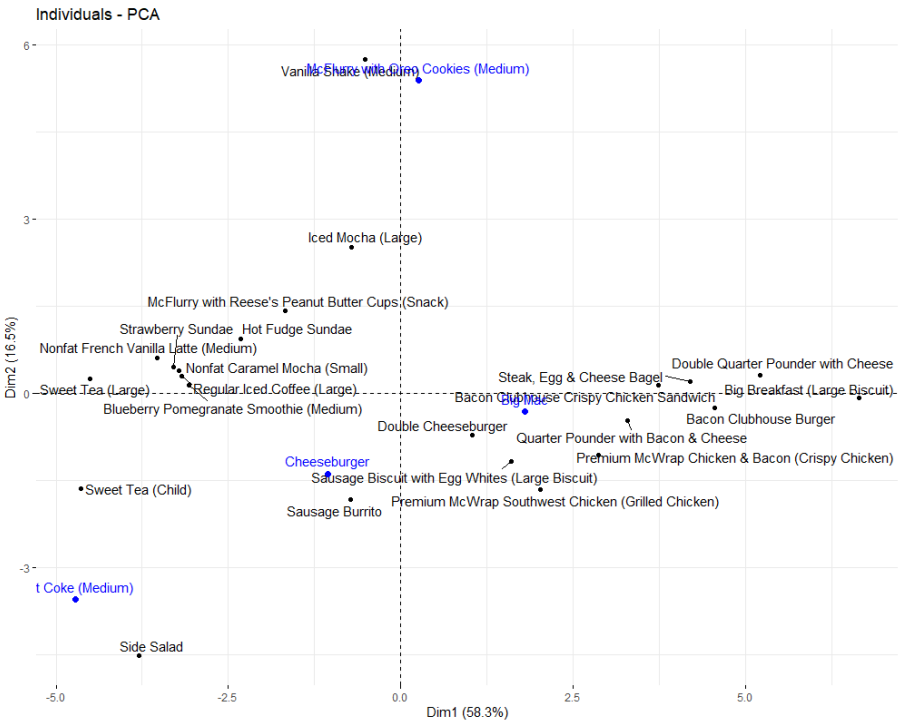

Figure 10: PCA prediction

As we can see, the diet coke is all to the left and in fact it was classified in cluster 3 (apparently the healthier one) and the McFlurry with Oreo Cookies is near to the Vanilla shake (so a food item that is not too different).
So, what we have here? We have a model for predict and classify the McDonald's food items, according to their similar dishes. As matter of fact, K-means clustering and PCA appear to have very different goals and at first sight do not seem to be related. However, as explained in the Ding & He 2004 paper "K-means Clustering via Principal Component Analysis", there is a deep connection between them.
With this model we can, again, find new dishes that are similar (from a nutritional point of view) to our starting set of items.

4 Conclusion

From our analysis turns out that the clusterization method are efficient also for food clustering, in particular menu clusterization, and it can be useful in order to help a customer to choose what to eat according to his/her nutritional exigence.
Finally, in this particular case, we learnt that:

- in mean, the sodium is near always the highest value;
- with 12 cluster we can divide the McDonald's menu in different group according to some common feature;
- the clusterization of this menu is correct and we prove that;
- we use another method (PCA) for analysing food item and create a sort of classification according to the nutritional fact of each food item.
- we use the method predict of PCA for testing if we can predict the classification of food item and it works.

So, with this few method, we got a way to suggest and/or help customer in his/her food ordering, in particular according to food similarity and nutritional facts. This, of course, can be extended and integrated with a system that can automatically suggest a menu according to the same thing. Moreover we can also think in a biological point of view and suggest food item also according to some disease like diabetes or anemia; this can be done with some supervised statistical learning method that are not the aim of this paper (here we think only in term of unsupervised learning).

5 Code appendix

In this final chapter we will include all the R code.

```
library(tidyverse)

Dataset <- read.csv("menu.csv")
Dataset <- na.omit(Dataset)

n <- nrow(Dataset)
p <- ncol(Dataset)

df <- Dataset[,4:p]

q <- ggplot(data.frame(mean = colMeans(df[,1:21]), nutrfact = names(df[,1:21]))) +
  geom_col(aes(nutrfact, mean)) + labs(x="Nutrition Facts", y="Mean")+
  labs(title="General case")
q + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

#CLUSTERING#

library(cluster)    # clustering algorithms
library(factoextra) # clustering algorithms & visualization

library(ggplot2)

wss <- (nrow(df)-1)*sum(apply(df,2,var))
for (i in 2:15) wss[i] <- sum(kmeans(df,centers=i)$withinss)
plot(1:15, wss, type="b", xlab="Number of Clusters",
     ylab="Within groups sum of squares")

min <- min(wss)
minPos <- match(min,wss)
minPos

#Runned 20 times we obtain a mean of 12.47 -> 12 cluster (more likely 12,14,15)

set.seed(20)
clusters <- kmeans(df, 12)

# Save the cluster number in the dataset as column 'cluster'
df$cluster <- as.factor(clusters$cluster)
str(clusters)

df$Item <- Dataset$Item

library(gridExtra)

# create a df for each cluster
cluster1 <- subset(df, cluster==1)
cluster2 <- subset(df, cluster==2)
cluster3 <- subset(df, cluster==3)
cluster4 <- subset(df, cluster==4)
cluster5 <- subset(df, cluster==5)
cluster6 <- subset(df, cluster==6)
cluster7 <- subset(df, cluster==7)
cluster8 <- subset(df, cluster==8)
cluster9 <- subset(df, cluster==9)
cluster10 <- subset(df, cluster==10)
```

```
cluster11 <- subset(df, cluster==11)
cluster12 <- subset(df, cluster==12)

#create the plot mean for each cluster
q1 <- ggplot(data.frame(mean = colMeans(cluster1[,1:21]),
                        nutrfact = names(cluster1[,1:21]))) +
  geom_col(aes(nutrfact, mean)) + labs(x="Nutrition␣Facts", y="Mean")+
  labs(title="Cluster1") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q2 <- ggplot(data.frame(mean = colMeans(cluster2[,1:21]),
                        nutrfact = names(cluster2[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster2")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q3 <- ggplot(data.frame(mean = colMeans(cluster3[,1:21]),
                        nutrfact = names(cluster3[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster3") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q4 <- ggplot(data.frame(mean = colMeans(cluster4[,1:21]),
                        nutrfact = names(cluster4[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster4") +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q5 <- ggplot(data.frame(mean = colMeans(cluster5[,1:21]),
                        nutrfact = names(cluster5[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster5")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q6 <- ggplot(data.frame(mean = colMeans(cluster6[,1:21]),
                        nutrfact = names(cluster6[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster6")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q7 <- ggplot(data.frame(mean = colMeans(cluster7[,1:21]),
                        nutrfact = names(cluster7[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster7")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q8 <- ggplot(data.frame(mean = colMeans(cluster8[,1:21]),
                        nutrfact = names(cluster8[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster8")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q9 <- ggplot(data.frame(mean = colMeans(cluster9[,1:21]),
                        nutrfact = names(cluster9[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition␣Facts", y="Mean")+ labs(title="Cluster9")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q10 <- ggplot(data.frame(mean = colMeans(cluster10[,1:21]),
```

```
                          nutrfact = names(cluster10[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition Facts", y="Mean")+ labs(title="Cluster10")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q11 <- ggplot(data.frame(mean = colMeans(cluster11[,1:21]),
                          nutrfact = names(cluster11[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition Facts", y="Mean")+ labs(title="Cluster11")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

q12 <- ggplot(data.frame(mean = colMeans(cluster12[,1:21]),
                          nutrfact = names(cluster12[,1:21]))) +
  geom_col(aes(nutrfact, mean)) +
  labs(x="Nutrition Facts", y="Mean")+ labs(title="Cluster12")+
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

#Plot cluster1-6 and cluster7-12 aggreagated
grid.arrange(q1,q2,q3,q4,q5,q6, ncol=3)
grid.arrange(q7,q8,q9,q10,q11,q12, ncol=3)

###HIERACHICAL CLUSTERING

library(cluster)
library(factoextra)

# Dissimilarity matrix
d <- dist(df[,1:21], method = "euclidean")
# Hierarchical clustering using Complete Linkage
hc1 <- hclust(d, method = "complete" )
# Plot the obtained dendrogram
plot(hc1, cex = 0.6, hang = -1)

hc4 <- diana(df[,1:21],metric = "manhattan", stand = FALSE)

# Divise coefficient
hc4$dc

# plot dendrogram
pltree(hc4, cex = 0.6, hang = -1, main = "Dendrogram of diana")

#Cut the tree according to the seleceted number of cluster
clust <- cutree(hc4, k = 12)
fviz_cluster(list(data = df, cluster = clust))

pltree(hc4, hang=-1, cex = 0.6)
rect.hclust(hc4, k = 12, border = 2:10)


### DISTANCE MANHATTAN ###
bigmac<-which(df$Item=='Big Mac') ## cluster 12
salad<-which(df$Item=='Side Salad') ## cluter 3

McChicken<-which(df$Item=='McChicken') ## cluter 5
Cheeseburger<-which(df$Item=='Cheeseburger') ## cluter 5

distance1 <- dist(df[c(bigmac,salad) , 1:22], method = "manhattan") #big distance

distance2 <- dist(df[c(McChicken,Cheeseburger) , 1:22], method = "manhattan")
```

```
#small distance

distance1
distance2

#### PCA WITH princomp ####

dfpc <- df[,1:21]
acp<-princomp(dfpc, cor=T) # use correaltion matrix
summary(princomp(dfpc, cor=T))
# select how many components:
screeplot(princomp(dfpc, cor=T))
# plot of the scores:
plot(princomp(dfpc, cor=T)$scores)
text(princomp(dfpc, cor=T)$scores, rownames(dfpc))
abline(h=0, v=0)
## biplot
biplot(acp)

# ----TWO TYPES OF PCA using RANDOM SAMPLE---- #

library(factoextra)

df.active <- df[sample(nrow(df), 23), 1:21] #df[1:23, c(1:9,12:21)]

#EIGENVALUE DECOMPOSITION
res.pca <- princomp(df.active, cor=T) #use correlation matrix
head(res.pca)

# Fetch and set the name of the random item
vector <- c()
for (y in rownames(res.pca$scores)){
  tmp<-as.character(df[y,"Item"])
  vector <- c(vector,tmp)
}
row.names(res.pca$scores)<-vector

fviz_eig(res.pca)
fviz_pca_ind(res.pca,
             col.ind = "cos2", # Color by the quality of representation
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
)

#SINGULAR VALUE DECOMPOSITION (SVD)

res.pca <- prcomp(df.active, scale = TRUE) #scale the data

# Fetch and set the name of the random item
vector <- c()
for (y in rownames(res.pca$x)){
  tmp<-as.character(df[y,"Item"])
  vector <- c(vector,tmp)
}
row.names(res.pca$x)<-vector

fviz_eig(res.pca)
fviz_pca_ind(res.pca,
             col.ind = "cos2", # Color by the quality of representation
```

```
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
)

# ———————— #

#GENERAL

fviz_pca_var(res.pca,
             col.var = "contrib", # Color by contributions to the PC
             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
             repel = TRUE     # Avoid text overlapping
)

fviz_pca_biplot(res.pca, repel = TRUE,
                col.var = "#2E9FDF", # Variables color
                col.ind = "#696969"  # Individuals color
)

#PREDICTION PCA
#Predict new value according to what PCA do before
ind.sup <- df[c(43,50,116,257), 1:21]
ind.sup.coord <- predict(res.pca, newdata = ind.sup)
rownames(ind.sup.coord)<-df$Item[c(43,50,116,257)]

# Add the new item to the prevoius plot
p <- fviz_pca_ind(res.pca, repel = TRUE)
fviz_add(p, ind.sup.coord, color ="blue")

#### PCA without princomp function
n <- nrow(df)
p <- ncol(df)

# mean and std:
M <- colMeans(df)
sigma <- apply(df, 2, sd)
descriptive<-round(cbind(M, sigma),2)
descriptive

## PCA start from correlation matrix
# calculate matrix R:
df1<-df[,1:21]
rho <- cor(df1)
round(rho,3)

# calculate eigenvalues and eigenvectors:
eigen(rho)
autoval <- eigen(rho)$values
autovec <- eigen(rho)$vectors

pvarsp = autoval/p
pvarspcum = cumsum(pvarsp)
tab<-round(cbind(autoval,pvarsp*100,pvarspcum*100),3)
colnames(tab)<-c("eigenvelues", "% variance","% cum variance")
tab

# Use Scree Diagram to select the components:
plot(autoval, type="b", main="Scree Diagram",
     xlab="Number of Component", ylab="Eigenvalues")
```

```
abline(h=1, lwd=3, col="red")

### We select two components
eigen(rho)$vectors[,1:2]

#Matrix of the components, obtained by multiplying the eigenvector by the root
##of the respective eigenvalue (if necessary we can change the sign
##for interpretative reasons)
comp<-round(cbind(-eigen(rho)$vectors[,1]*sqrt(autoval[1]),
                  -eigen(rho)$vectors[,2]*sqrt(autoval[2])),3)
rownames(comp)<-row.names(descriptive)
colnames(comp)<-c("Comp1","Comp2")
comp

# The sum of the squares of the values of each row of the component matrix
#  is the respective 'communality',
# The communality is the sum of the squared component
#  loadings up to the number of components you extract.
communality<-comp[,1]^2+comp[,2]
comp<-cbind(comp,communality)
comp


# Calculate the scores for the selected components and graph them:
df.scale <- scale(df[,1:21], T, T)
score <- df.scale%*%autovec[,1:2]
# normalized scores changed sign
# (non-normalized scores divided by square root of the respective eigenvalue)
## score chart
scorez<-round(cbind(-score[,1]/sqrt(autoval[1]),
                    -score[,2]/sqrt(autoval[2]),-score[,2]/sqrt(autoval[2])),2)
plot(scorez, main="Scores plot",
     xlab="comp1",ylab="comp2")
text(scorez, rownames(df))
abline(v=0,h=0,col="red")
# Loadings plot
plot(comp[,1:2], main="Loadings plot",
     xlab="comp1",ylab="comp2", xlim=range(-1,1))
text(comp, rownames(comp))
abline(v=0,h=0,col="red")
```