Importar Librerias

In [75]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn import preprocessing
from sklearn.cluster import KMeans
import warnings
warnings.filterwarnings('ignore')
from sklearn.decomposition import PCA
```

Cargar datos

In [90]:
```
df = df = pd.read_csv('./Wholesale customers data.csv')
df=df.dropna()
df.sample(10)
```

Out[90]:

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
425	1	3	11243	2408	2593	15348	108	1886
43	2	3	630	11095	23998	787	9529	72
97	1	3	403	254	610	774	54	63
218	2	1	18	7504	15205	1285	4797	6372
235	1	1	1838	6380	2824	1218	1216	295
285	1	3	40254	640	3600	1042	436	18
288	1	3	16260	594	1296	848	445	258
135	1	3	6300	1289	2591	1170	199	326
238	1	1	7363	475	585	1112	72	216
199	1	1	9670	2280	2112	520	402	347

Datos de Muestra

In [91]:
```
indices=[230,310,21]
df = df.drop(['Region','Channel'],axis=1)
muestra = pd.DataFrame(df.loc[indices],columns=df.keys()).reset_index(drop=True)
```

In [92]:
```
df=df.drop(indices,axis=0)
```

Procesamiento de los Datos

Escalamiento de datos

In [93]:
```
df_escalada= preprocessing.Normalizer().fit_transform(df)
muestra_escalada= preprocessing.Normalizer().fit_transform(muestra)
```

Analisis

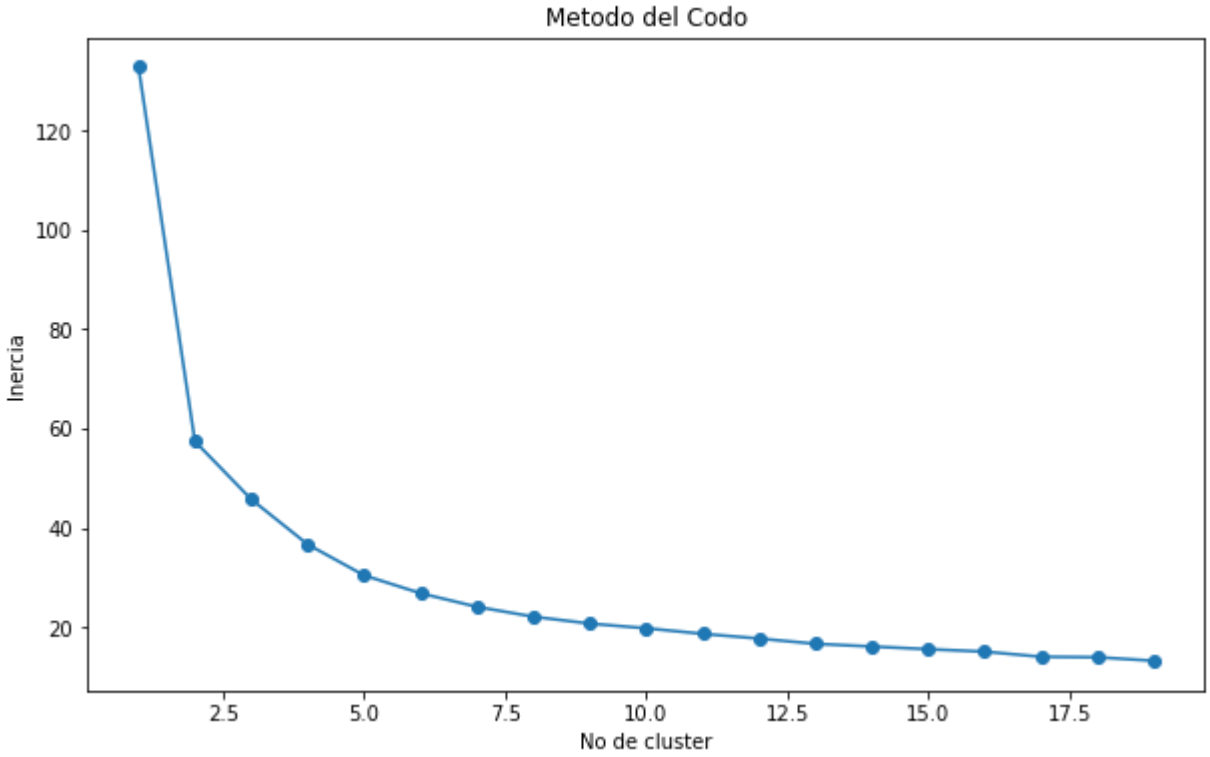
In [94]:
```
x= df_escalada.copy()
```

Hallo el valor de K optimo usando el metodo de Codo

In [95]:
```
inercia=[]
for i in range(1,20):
    algoritmo= KMeans(n_clusters=i,init= 'k-means++',max_iter=300,n_init=10)
    algoritmo.fit(x)
    inercia.append(algoritmo.inertia_)
```

Grafico el Codo

In [96]:
```
plt.figure(figsize=(10,6))
plt.title('Metodo del Codo')
plt.xlabel('No de cluster')
plt.ylabel('Inercia')
plt.plot(list(range(1,20)),inercia, marker='o')
plt.show()
```



Aplicamos el Algoritmo Kmeans

In [97]:
```
algoritmo= KMeans(n_clusters=6,init='k-means++',max_iter=300,n_init=10)
```

In [98]:
```
algoritmo.fit(x)
```

Out[98]:
```
KMeans(n_clusters=6)
```

In [99]:
```
centroides,etiquetas= algoritmo.cluster_centers_,algoritmo.labels_
```

Verficamos los datos de muestra

In [100…
```
muestra_prediccion= algoritmo.predict(muestra_escalada)
for i, pred in enumerate(muestra_prediccion):
    print('Muestra',i,' se encuentra en el cluster: ',pred)
```

```
Muestra 0  se encuentra en el cluster:  5
Muestra 1  se encuentra en el cluster:  3
Muestra 2  se encuentra en el cluster:  5
```

Grafico del cluster

In [101…
```
modelo_pca = PCA(n_components=2)
modelo_pca.fit(x)
pca=modelo_pca.transform(x)
```

In [102…
```
centroides_pca=modelo_pca.transform(centroides)
```

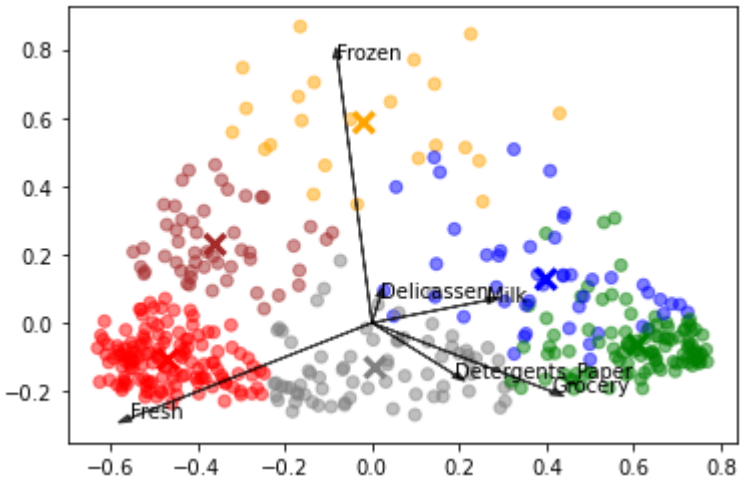
In [112…
```
colores=['blue','red','green','orange','gray','brown']
colores_cluster=[colores[etiquetas[i]] for i in range(len(pca))]
```

In [114…
```
plt.scatter(pca[:,0],pca[:,1],c=colores_cluster,marker='o',alpha=0.5)
plt.scatter(centroides_pca[:,0],centroides_pca[:,1],marker='x',s=100,linewidths=3,c=co

xvector=modelo_pca.components_[0] * max(pca[:,0])
yvector=modelo_pca.components_[1] * max(pca[:,1])
columnas=df.columns

for i in range(len(columnas)):
    plt.arrow(0,0,xvector[i],yvector[i],color='black',width = 0.0005,head_width=0.02,a
    plt.text(xvector[i],yvector[i],list(columnas)[i],color='black')

plt.show()
```



In [ ]:

In [ ]: