Tarea 4 algoritmos de clustering

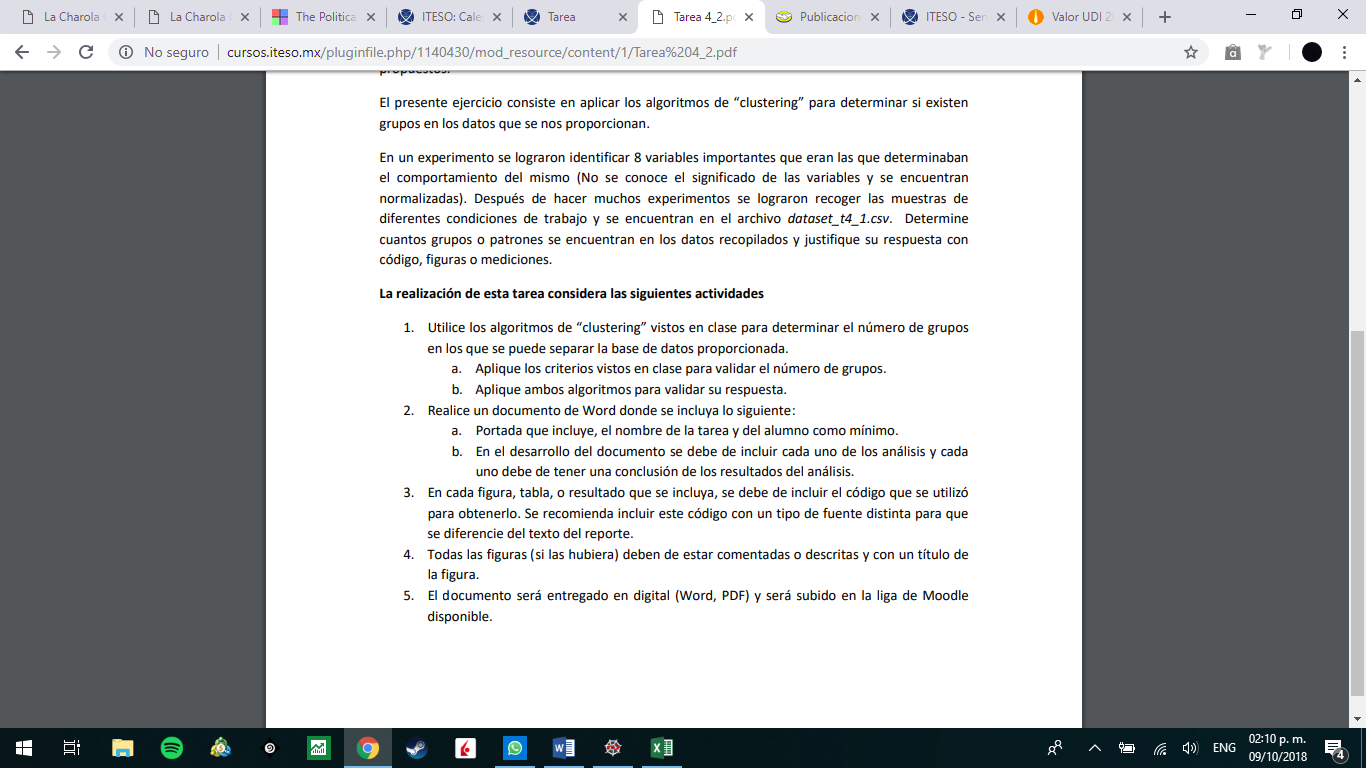
Juan Pablo Estavillo Urrea

Ciencia de Datos

9 de octubre de 2018

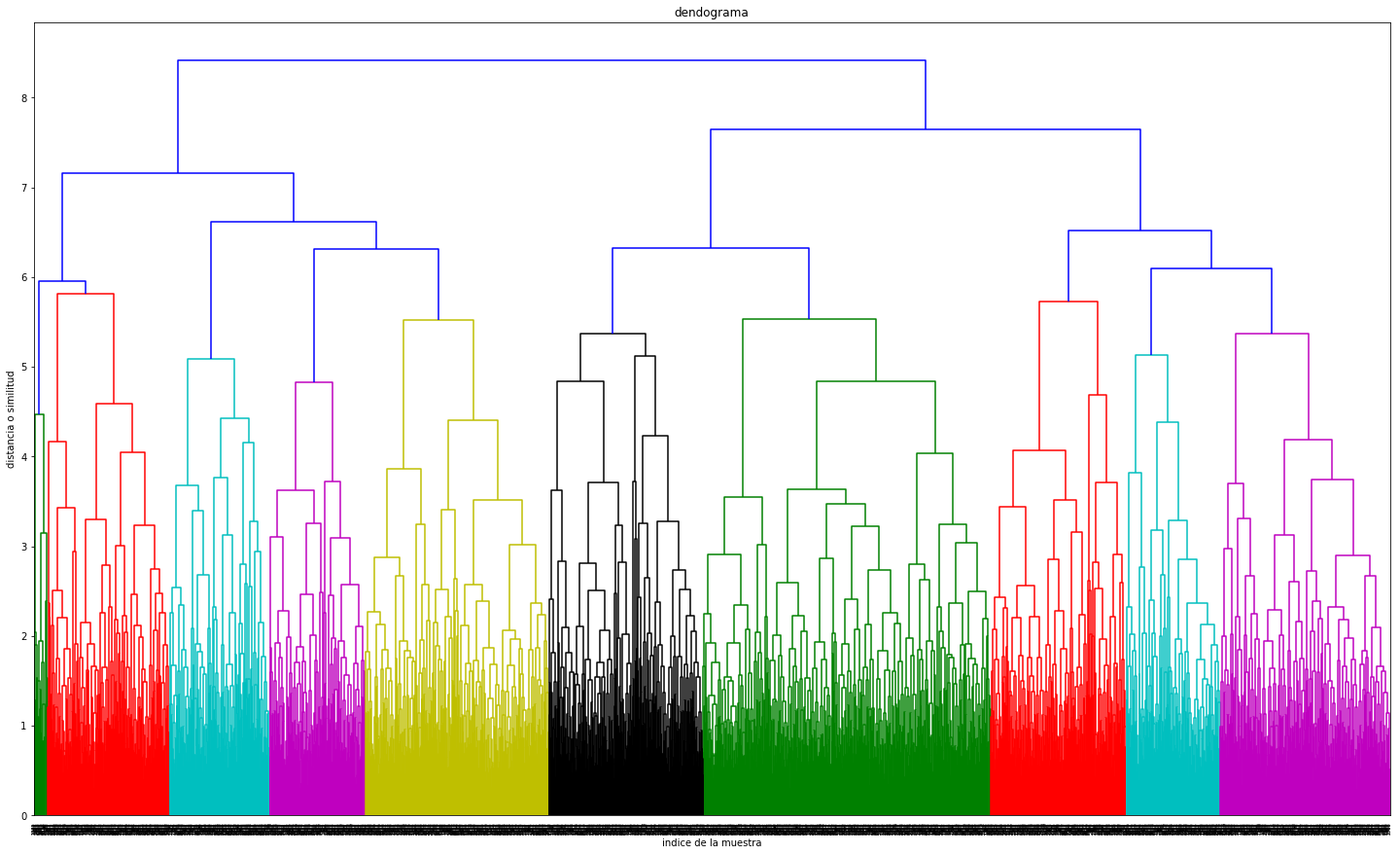
ITESO A.C

1

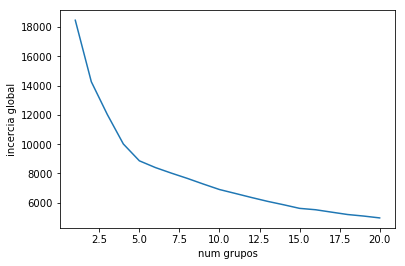


1. **Hierarchy**

En este modelo de Clustering se usa para agrupar los datos por distancias, en este caso se agruparon por distancias euclidianas. En el eje x de la siguiente grafica tenemos a los índices de todos los datos encontrados en el archivo de Excel, mientras que en el eje y se puede apreciar la distancia que existen entre los datos, conforme avanza el modelo, se agrupan mas datos aumentando así la distancia entre ellos.



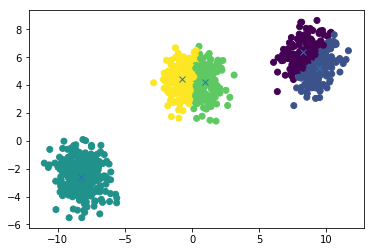
**Seleccionando el numero ideal de grupos para el cluster (inersias)**



En esta grafica de brazo codo se puede ver claramente que el numero de grupos óptimos para realizar un clustering es de 5, debido a que la pendiente negativa disminuye.

**KMeans**

En este tipo de clustering, se denomina un grupo n de Clusters prestablecidos como centroides, el cual ya calculamos y es 5 grupos en este caso, de donde el código ira creando diferentes grupos dependiendo de la distancia que los datos tengan con respecto al centroide, ósea que el código inicia con 5 datos aleatorios y va agrupándose con los mas cercanos que encuentra a su paso, creando así 5 grupos de Clustering.



**Código usado para la tarea**

*import pandas as pd*

*import numpy as np*

*import matplotlib.pyplot as plt*

*from scipy.cluster import hierarchy*

*import scipy.spatial.distance as sc*

*from sklearn.cluster import KMeans*

*#%% importamos datos*

*data=pd.read\_csv('../data/dataset\_t4\_1.csv')*

*#%% random seed*

*np.random.seed(seed=69)*

*semilla=69*

*#%% clustering using eucliean*

*Z= hierarchy.linkage(data,metric='euclidean',method='complete')*

*plt.figure(figsize=(25,15))*

*plt.title('dendograma')*

*plt.xlabel('indice de la muestra')*

*plt.ylabel('distancia o similitud')*

*dn=hierarchy.dendrogram(Z)*

*plt.show()*

*#%% decidiendo el numero en el cluster*

*inersias = np.zeros(20) # crea un arreglo de 10 espacios*

*for k in np.arange(20)+1:*

*model = KMeans(n\_clusters=k, random\_state=semilla, init='random')*

*model= model.fit(data)*

*inersias[k-1]=model.inertia\_*

*plt.plot(np.arange(1,21),inersias)*

*plt.xlabel('num grupos')*

*plt.ylabel('incercia global')*

*plt.show()*

*#%%*

*n\_optimo=5*

*#%% clustering with kMeans*

*model = KMeans(n\_clusters=n\_optimo, random\_state=69, init='random') #Random la manera en que inicializan los centroides*

*model= model.fit(data) # fit ejecuta todo el algoritmo. El de las iteraciones.*

*centroides = model.cluster\_centers\_ ## Para ver donde están los centroides. Te da las coordenadas*

*Ypredict = model.predict(data) #predict te dice que dato pertence a que grupo.*

*#El número de variables es el número de dimensiones*

*#?KMeans*

*plt.scatter(Z[:,0],Z[:,1], c= Ypredict)*

*plt.show()*

*J = model.inertia\_ # Promedio de la distancia euclidiana.*

*#%% Criterio de desicioón el n´mero de clusters*

*inersias = np.zeros(10) # crea un arreglo de 10 espacios*

*for k in np.arange(10)+1:*

*model = KMeans(n\_clusters=k, random\_state=69, init='random')*

*model= model.fit(data)*

*inersias[k-1]=model.inertia\_*

*plt.plot(np.arange(1,11),inersias)*

*plt.xlabel('num grupos')*

*plt.ylabel('incercia global')*

*plt.show()*