Segmentacion de Clientes

## 

Contenido

[Intro 1](#_Toc523242077)

[desarrollo 2](#_Toc523242078)

[Cargar el archivo de datos: 2](#_Toc523242079)

[Anlizar los percentiles 3](#_Toc523242080)

[Transformacion 3](#_Toc523242081)

[Analisis de outliers 4](#_Toc523242082)

[Analisis de segmentos 6](#_Toc523242083)

## Intro

En este taller, analizaremos un conjunto de datos que contiene datos sobre los montos de gastos anuales de varios clientes (informados en unidades monetarias) de diversas categorías de productos. Uno de los objetivos de este proyecto es describir mejor la variación en los diferentes tipos de clientes con los que interactúa un distribuidor mayorista. Hacerlo equiparía al distribuidor con una idea de cómo estructurar mejor su servicio de entrega para satisfacer las necesidades de cada cliente.  
  
El conjunto de datos para este proyecto se puede encontrar en el Repositorio de aprendizaje automático de UCI (<https://archive.ics.uci.edu/ml/datasets/Wholesale+customers> ).Se excluirán en el análisis los atributos **Channel** y **Region**, centrándose en cambio en las seis categorías de productos registradas para los clientes.

## desarrollo

Ejecutar un notebook

Importar las librerías:

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from IPython.display import display

## Cargar el archivo de datos:

data = pd.read\_csv("customers.csv")

data.drop(['Region', 'Channel'], axis = 1, inplace = True)

Obtener estadisticos basicos

display(data.describe())

Seleccione tres indices (al azar) los cuales va a revisar en particular

indices = [22,154,398]

Crear un data frame con los ejemplos de muestra señalados

samples = pd.DataFrame(data.loc[indices], columns = data.keys()).reset\_index(drop = True)

print("Ejemplos sleccionados:")

display(samples)

## Anlizar los percentiles

pcts = 100. \* data.rank(axis=0, pct=True).iloc[indices].round(decimals=3)

sns.heatmap(pcts.reset\_index(drop=True), annot=True, vmin=1, vmax=99, fmt='.1f', cmap='YlGnBu')

plt.title('rangos de percentiles de la muestra’)

plt.xticks(rotation=45, ha='center');

* Actividad: indique su interpretacion del grafico obtenido

# Genere una matrix de graficos de dispersion para cada par de atributos del conjunto de datos.

pd.plotting.scatter\_matrix(data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');

* Actividad: Que analisis hace de las graficas generadas? Que patrón puede señalar?

## Transformacion

Escale los datos, usando logaritmo natural

log\_data = np.log(data.copy())

grafique los datos en una nueva matriz

pd.plotting.scatter\_matrix(log\_data, alpha = 0.3, figsize = (14,8), diagonal = 'kde');

Compare la muestra original, con la generada a partir de la transformacion log

print("ejemplos Originales: ")

display(samples)

Muestre los datos con la transformacion log

print("ejemplos Log-transformados:")

display(log\_samples)

* Actividad:
* En que consiste la transformacion log? (ref: <http://onlinestatbook.com/2/transformations/log.html>
* Cual fue el resultado luego de la transformacion?

## Analisis de outliers

# Para cada atributo, encontrar los puntos con valores extremos muy altos o muy bajos

for feature in log\_data.keys():

Calcular Q1 ( percentil 25) para el atributo dado

Q1 = np.percentile(log\_data, 25)

Calcular Q3 ( percentile 75) para el atributo dado

Q3 = np.percentile(log\_data, 75)

Usa el rango rango interquartile para calcular un paso outlier (1.5 veces el rango interquartil)

step = (Q3 - Q1) \* 1.5

Mostrar los outliers

print("Puntos considerados outlier para el atributo '{}':".format(feature))

display(log\_data[~((log\_data[feature] >= Q1 - step) & (log\_data[feature] <= Q3 + step))])

Seleccione los indices para los puntos que desea remover

outliers = [66, 75, 338, 142, 154, 289]

# Remueva los puntos

good\_data = log\_data.drop(log\_data.index[outliers]).reset\_index(drop = True)

🡪 Actividad :

Entonces en que consiste el rango interquartile para deteccion de outliers?

## Analisis de segmentos

from sklearn.metrics import silhouette\_score

from sklearn.cluster import KMeans

n\_clusters = 2

Aplicar el algoritmo de clustering k-means

clusterer = KMeans(n\_clusters=n\_clusters, random\_state=1)

clusterer.fit(log\_data)

Predecir el cluster para cada ejemplo

preds = clusterer.predict(log\_data)

Obtener los centros de cluster

centers = clusterer.cluster\_centers\_

Calcular el coeficiente de silueta para los clusters dados

score = silhouette\_score(log\_data, preds, random\_state=1)

print("{0} clusters: {1:.4f}".format(n\_clusters, score))

* Actividad:
* En que consiste el coeficiente de silueta? Ref: <http://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html>
* Pruebe con distintos valores para n\_cluster, e indique con cual obtuvo el mejor coeficiente

(2,4,6,8,10)

Cada grupo cluster generado tiene un punto central. Estos centros (o medios) no son específicamente puntos de datos de los datos, sino más bien los promedios de todos los puntos de datos predichos en los grupos respectivos. Para el problema de crear segmentos de clientes, el punto central de un clúster corresponde al cliente promedio de ese segmento. Como actualmente los datos se escalan mediante un logaritmo, podemos recuperar el gasto representativo de los clientes a partir de estos puntos de datos aplicando las transformaciones inversas.

# Exponenciacion de los centros

true\_centers = np.exp(centers)

# Mostrar los centros reales

segments = ['Segmento {}'.format(i) for i in range(0,len(centers))]

true\_centers = pd.DataFrame(np.round(true\_centers), columns = data.keys())

true\_centers.index = segments

display(true\_centers)

* Actividad:
* índique su análisis apartir de los segmentos generados