**Agrupación jerárquica**

En este ejemplo, realizaremos un agrupamiento jerárquico en datos del mundo real y veremos cómo se puede usar para resolver un problema real.

El problema que vamos a resolver en esta sección es segmentar a los clientes en diferentes grupos en función de sus tendencias de compra.

El conjunto de datos para este problema se puede descargar desde el siguiente enlace:

[compras-datos.csv](https://stackabuse.s3.amazonaws.com/files/hierarchical-clustering-with-python-and-scikit-learn-shopping-data.csv)

Coloque el archivo "shopping-data.csv" descargado en la carpeta "Conjuntos de datos" del directorio "D". Para agrupar estos datos en grupos seguiremos los mismos pasos que realizamos en el apartado anterior.

Ejecute el siguiente script para importar las bibliotecas deseadas:

import matplotlib.pyplot as plt

import pandas as pd

%matplotlib inline

import numpy as np

A continuación, para importar el conjunto de datos de este ejemplo, ejecute el siguiente código:

customer\_data = pd.read\_csv('D:\Datasets\shopping-data.csv')

Exploremos un poco nuestro conjunto de datos. Para verificar la cantidad de registros y atributos, ejecute el siguiente script:

customer\_data.shape

El script anterior devolverá (200, 5)lo que significa que el conjunto de datos contiene 200 registros y 5 atributos.

Para observar el conjunto de datos, ejecute la head()función del marco de datos. Echa un vistazo al siguiente script:

customer\_data.head()

La salida se verá así:

|  | **Identificación del cliente** | **Género** | **Años** | **Ingreso Anual (k$)** | **Puntuación de gastos (1-100)** |
| --- | --- | --- | --- | --- | --- |
| 0 | 1 | Masculino | 19 | 15 | 39 |
| 1 | 2 | Masculino | 21 | 15 | 81 |
| 2 | 3 | Mujer | 20 | dieciséis | 6 |
| 3 | 4 | Mujer | 23 | dieciséis | 77 |
| 4 | 5 | Mujer | 31 | 17 | 40 |

Nuestro conjunto de datos tiene cinco columnas: ID de cliente, Género, Edad, Ingreso anual y Puntuación de gastos. Para ver los resultados en un espacio de características bidimensional, conservaremos solo dos de estas cinco columnas. Podemos eliminar las columnas CustomerID, Género y Edad. Conservaremos las columnas Ingreso anual (en miles de dólares) y Puntuación de gastos (1-100). La columna Puntuación de gastos indica la frecuencia con la que una persona gasta dinero en un centro comercial en una escala del 1 al 100, siendo 100 el que más gasta. Ejecute el siguiente script para filtrar las tres primeras columnas de nuestro conjunto de datos:

data = customer\_data.iloc[:, 3:5].values

A continuación, necesitamos saber los grupos en los que queremos que se dividan nuestros datos. Usaremos nuevamente la scipy biblioteca para crear los dendrogramas para nuestro conjunto de datos. Ejecute el siguiente script para hacerlo:

import scipy.cluster.hierarchy as shc

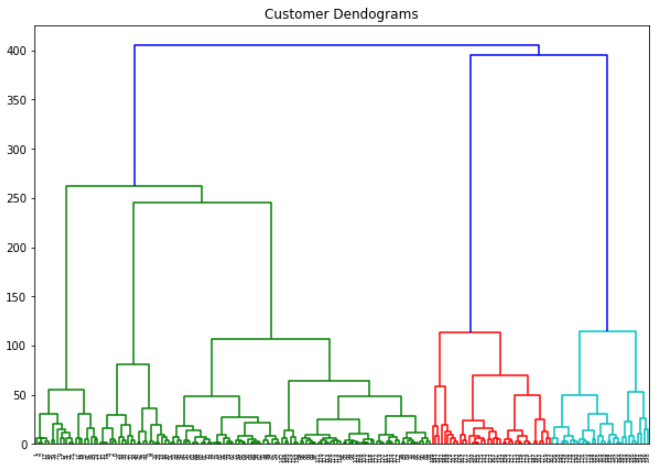
plt.figure(figsize=(10, 7))

plt.title("Customer Dendograms")

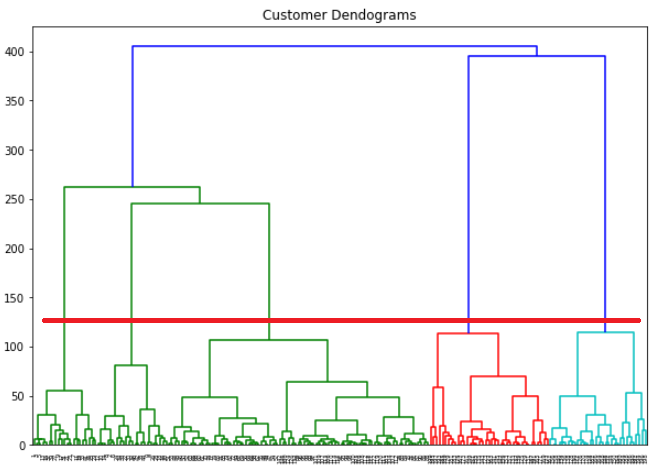
dend = shc.dendrogram(shc.linkage(data, method='ward'))

En el script anterior, importamos la clase de jerarquía de la scipy.clusterbiblioteca como shc. La clase de jerarquía tiene un dendrogrammétodo que toma el valor devuelto por el linkagemétodo de la misma clase. El linkagemétodo toma el conjunto de datos y el método para minimizar distancias como parámetros. Usamos 'ward' como el methodya que minimiza las variantes de distancias entre los grupos.

El resultado del script anterior se ve así:



Si dibujamos una línea horizontal que pasa por la distancia más larga sin una línea horizontal, obtenemos 5 grupos como se muestra en la siguiente figura:



Ahora que conocemos la cantidad de grupos para nuestro conjunto de datos, el siguiente paso es agrupar los puntos de datos en estos cinco grupos. Para ello utilizaremos de nuevo la AgglomerativeClusteringclase de la sklearn.clusterbiblioteca. Echa un vistazo al siguiente script:

from sklearn.cluster import AgglomerativeClustering

cluster = AgglomerativeClustering(n\_clusters=5, affinity='euclidean', linkage='ward')

cluster.fit\_predict(data)

El resultado del script anterior se ve así:

array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4,

3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1, 4, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,

1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2,

0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 1,

2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,

0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2], dtype=int64)

Puede ver las etiquetas de clúster de todos sus puntos de datos. Como teníamos cinco grupos, tenemos cinco etiquetas en la salida, es decir, 0 a 4.