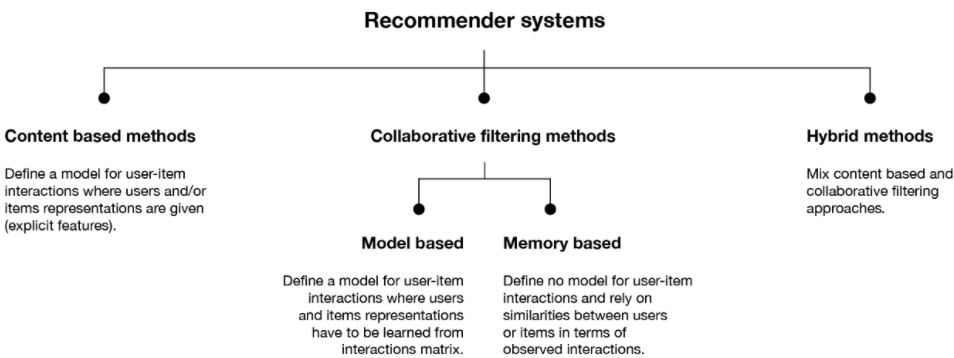
**Sistema de Recomendación**

**- Explicit Feedback:** Datos cuantitativos recogidos de los usuarios, que nos dan ellos voluntariamente.

**- Implicit Feedback:** Datos recopilados indirectamente de las interacciones del usuario.



Collaborative filtering methods: USER-ITEM (para ello los usuarios serán las filas, y los items las columnas)

* Basado en **iteraciones pasadas** de elementos de usuario.
* Detectar **usuarios similares** o **elementos similares**.
* MEMORY BASED (vecinos más cercanos) y MODEL BASED (modelo generativo subyacente).
* No requieren información sobre los usuarios o artículos, sino sobre su **comportamiento**.
* A más interacciones => más precisión.

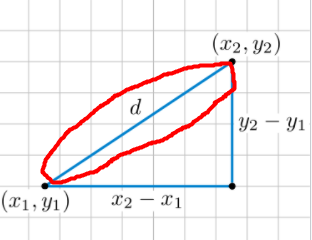
Vamos a utilizar la librería [SciPy](https://scipy.github.io/devdocs/reference/index.html) → que contiene módulos para optimización, álgebra lineal, integración, interpolación, funciones especiales, FFT, procesamiento de señales e imágenes, solucionadores de ODE y otras tareas comunes en ciencia e ingeniería.

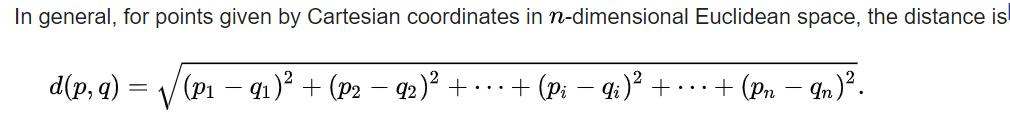
* *(conda install -c anaconda scipy)*
* **from** scipy.spatial.distance **import** pdist, squareform

MEMORY BASED MODEL (similitudes entre usuarios basadas en las interacciones de los usuarios):

**ELEMENTOS:**

* USUARIOS
* ITEMS (en este caso películas y sus puntuaciones)
* DISTANCIA entre usuarios (Distancias por pares entre observaciones en el espacio n-dimensional)
  + En matemáticas, la **distancia euclidiana** o euclídea, es la distancia "ordinaria" entre dos puntos de un espacio euclídeo, la cual se deduce a partir del teorema de Pitágoras.



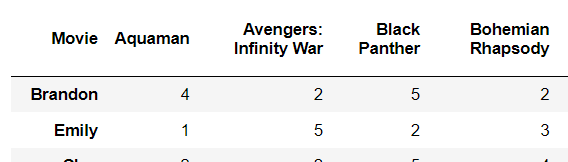


**PASOS:**

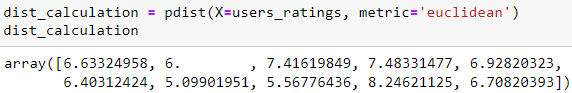
1. Importamos el dataset y creamos un dataframe:
   1. USUARIOS como FILAS
   2. ITEMS como COLUMNAS



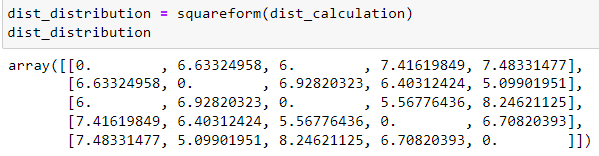


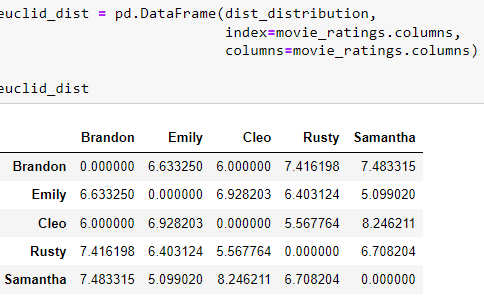


1. calculamos las distancias “eucledianas”: **pdist(**X=users\_ratings,metric='euclidean'**)**

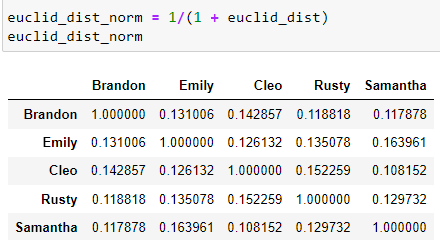


1. calculamos el “cuadrado” y su dataFrame: **squareform(**dist\_calculation**)**





1. “normalizamos” el anterior dataFrame: **1/(1+** dataFrame**)**

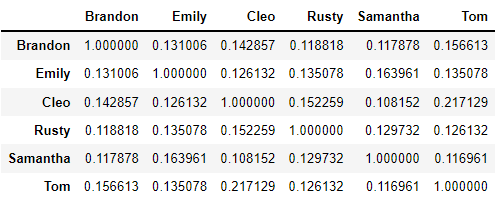


**\* pasos 2, 3 y 4 JUNTOS (**continuamos con otro ejemplo diferente en el que hemos añadido a TOM como usuario**):**

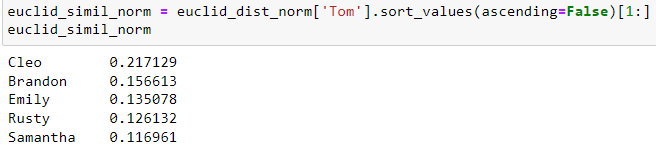
pd.DataFrame**(1/(1 + squareform(pdist(**movie\_ratings.T**,** 'euclidean'**))),**

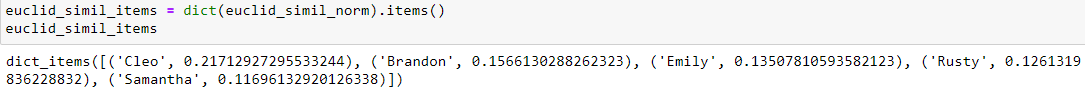
index=movie\_ratings.columns,

columns=movie\_ratings.columns)

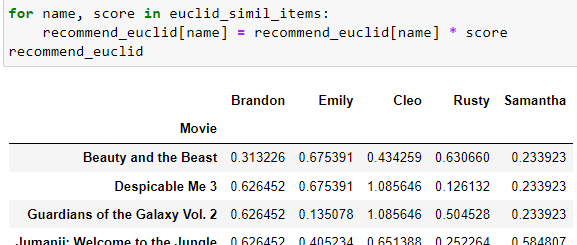


1. De un usuario obtenemos la SIMILITUD con el resto, lo almacenamos en un DICT e importamos un nuevo dataset (en el que este último usuario no ha visto las películas)



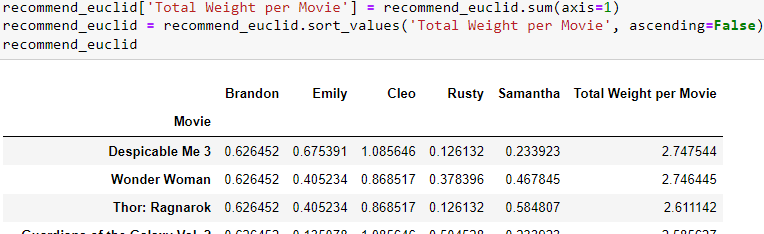


1. multiplicamos dicho dataset por la similitud para PONDERAR las valoraciones:

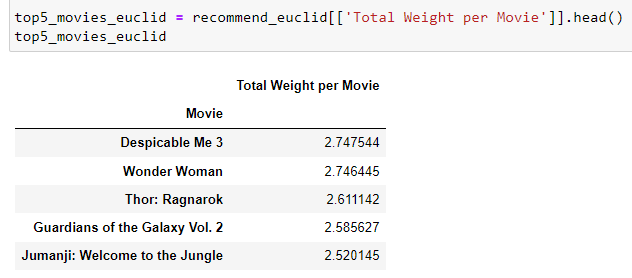


1. Asignamos un PESO de similitud con Tom a cada película:

* añadimos la columna de la suma
* ordenamos los valores del DF en base a esa columna



1. Mostramos solo las recomendaciones finales para Tom



**\* pasos 5 - 8 JUNTOS:**

**euclid\_simil\_norm** = euclid\_dist\_norm['Tom'].sort\_values(ascending=False)[1:]

**movie\_ratings\_test** = pd.read\_csv('./datasets/movie\_ratings\_test.csv').set\_index('Movie')

**recommend\_euclid** = movie\_ratings\_test.copy()

**euclid\_simil\_items** = dict(euclid\_simil\_norm).items()

for name, score in euclid\_simil\_items:

recommend\_euclid[name] = recommend\_euclid[name] \* score

recommend\_euclid['Total Weight per Movie'] = recommend\_euclid.sum(axis=1)

**recommend\_euclid** = recommend\_euclid.sort\_values('Total Weight per Movie', ascending=False)

**top5\_movies\_euclid** = recommend\_euclid[['Total Weight per Movie']]

**top5\_movies\_euclid**