**GUIA ANALISIS DE CANASTA / ANALISIS DE ASOCIACION**

**OBJETIVO**

Realizar un proceso de análisis de reglas de asociación, con base en el conjunto de datos de valoración de películas **movielens** ,a través de python notebook.

# Instalación y Configuración

## Prerrequisitos

Jupyter Notebook

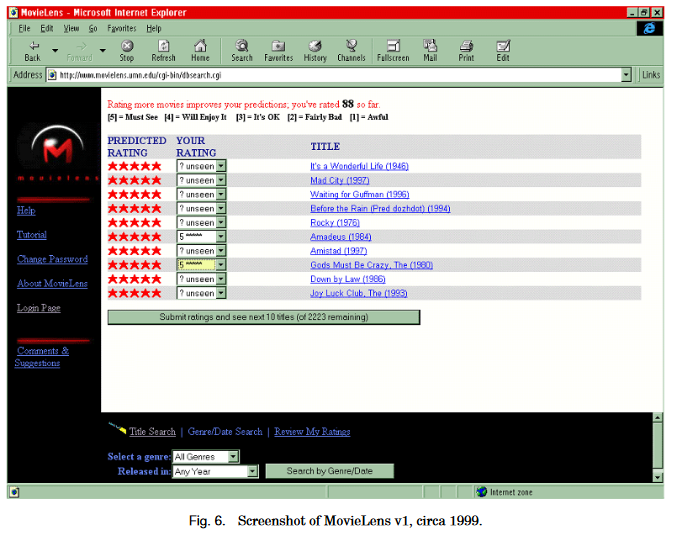
Conexión a Internet.

## Conjunto de Datos

Descripcion:

El conjunto de datos movielens es el resultado de la interaccion de los usuarios con el sitio online movielens.org, lugar donde los usuarios registran sus valoraciones(ratings) únicamente a las películas que aparecen en su pagina. Estos datos son utilzzados con fines de investigación en minería de datos, en especial, análisis de market basket y recomendación de ítems.

Al no ser un sitio comercial, las valoraciones tienen un alto grado de veracidad



El conjunto de datos se compone de varios archivos:

movies.csv :: Metadata de las peliculas: codificacion, nombre generos a los cuales pertenece.

ratings.csv :: Valoraciones de cada usuario a las distintas películas. Cada registro se corresponde con una valoración.

tags.csv : etiquetas colocadas alas peliculas por los usuarios.

links.csv :: Enlace a otros sitios de valoracion como <http://www.imdb.com> y <https://www.themoviedb.org>

Sin embargo, para este laboratorio, solo se utilizarán los dos primeros archivos de datos.

Mas información en <https://old.datahub.io/es/dataset/movielens>

# Desarrollo

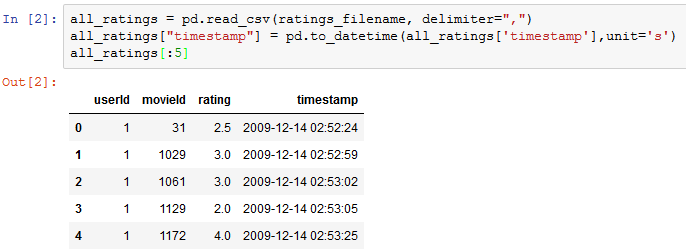
Leeremos el archivo de datos ratings.csv



El módulo os nos permite acceder a funcionalidades dependientes del Sistema Operativo. Sobre todo, aquellas que nos refieren información sobre el entorno del mismo y nos permiten manipular la estructura de directorios (para leer y escribir archivos)

Mayor detalle en <http://docs.python.org/library/os.html>

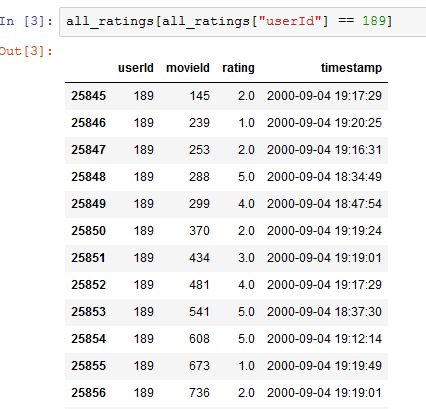
La función to\_datetime de pandas, permite convertir un valor numérico a su equivalente en formato fecha:



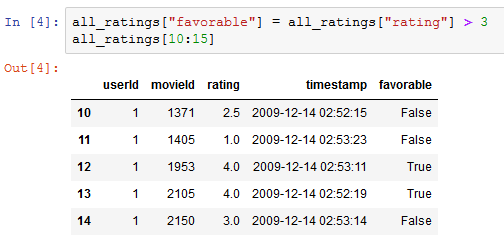
Algunas funciones de conversión en:

<https://pandas.pydata.org/pandas-docs/stable/api.html#general-functions>

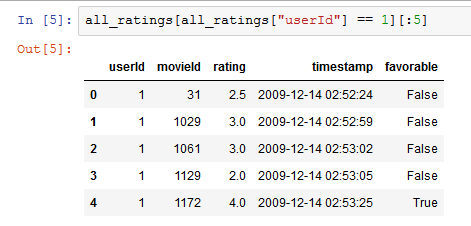
Filtremos las valoraciones hechas por el usuario 189



Realicemos una transformación de las valoraciones superiores a 3, como positivas y las menores a 3 como negativas en una nueva variable llamada **favorable**



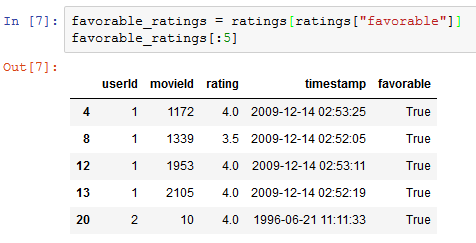
Consultemos como fueron las valoraciones del usuario 1



El tipo range es una lista de números enteros en forma de sucesión, el cual es inmutable(no se puede modificar)



Explique que hace el siguiente fragmento??



## Sets y Frozensets

Un set en Python es una colección de elementos únicos , el cual no presenta un ordenamiento .

Podemos crear un set y agregarle elementos de la siguiente manera:

x = set(["Perl", "Python", "Java"])

x.add("SQL")

Por otra parte un frozenset por algo así como una coleccion congelada, término que muestra claramente su carácter **estático**. Podemos pensar en un frozenset como en un set en el que no podemos modificar su composición una vez creado.

x = set(["Perl", "Python", "Java"])

~~x.add("SQL")~~



## Diccionarios

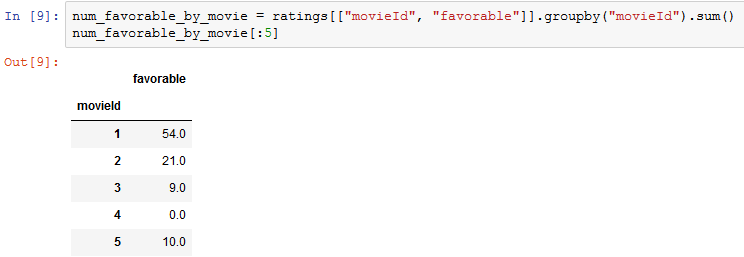
Otro tipo de dato útil incluído en Python es el *diccionario*. Los diccionarios se encuentran a veces en otros lenguajes como “memorias asociativas” o “arreglos asociativos”. Se puede pensar en un diccionario como un conjunto no ordenado de pares *clave: valor*, con el requerimiento de que las claves sean únicas (dentro de un diccionario en particular). Un par de llaves crean un diccionario vacío: {}. Colocar una lista de pares clave:valor separados por comas entre las llaves añade pares clave:valor iniciales al diccionario; esta también es la forma en que los diccionarios se presentan en la salida.

Las operaciones principales sobre un diccionario son guardar un valor con una clave y extraer ese valor dada la clave. También es posible borrar un par clave:valor con del. Si usa una clave que ya está en uso para guardar un valor, el valor que estaba asociado con esa clave se pierde. Es un error extraer un valor usando una clave no existente.

El método keys() de un diccionario devuelve una lista de todas las claves en uso de ese diccionario, en un orden arbitrario . Para verificar si una clave está en el diccionario, utilizar la palabra clave in.

Pruebe:

favorable\_reviews\_by\_users.keys()



En una nueva celda copie y pegue el siguiente fragmento:

#####################################

from collections import defaultdict

def find\_frequent\_itemsets(favorable\_reviews\_by\_users, k\_1\_itemsets, min\_support):

counts = defaultdict(int)

for user, reviews in favorable\_reviews\_by\_users.items():

for itemset in k\_1\_itemsets:

if itemset.issubset(reviews):

for other\_reviewed\_movie in reviews - itemset:

current\_superset = itemset | frozenset((other\_reviewed\_movie,))

counts[current\_superset] += 1

return dict([(itemset, frequency) for itemset, frequency in counts.items() if frequency >= min\_support])

#######################################

## Fragmentos def

En Python, la definición de funciones se realiza mediante la instrucción def más un nombre de función descriptivo -para el cuál, aplican las mismas reglas que para el nombre de las variables- seguido de paréntesis de apertura y cierre. Como toda estructura de control en Python, la definición de la función finaliza con dos puntos (:) y el algoritmo que la compone, irá identado con 4 espacios:

def mi\_funcion():

# aquí el algoritmo

Una función, no es ejecutada hasta tanto no sea invocada. Para invocar una función, simplemente se la llama por su nombre:

def mi\_funcion():

print "Hola Mundo"

mi\_funcion()

Ahora agregue una nueva celda y copie el siguiente fragment y ejecútelo..

########################################

import sys

frequent\_itemsets = {} # itemsets are sorted by length

min\_support = 50

# candidatos k=1 convaloraciones favorablea mayors a min\_support

frequent\_itemsets[1] = dict((frozenset((movie\_id,)), row["favorable"])

for movie\_id, row in num\_favorable\_by\_movie.iterrows()

if row["favorable"] > min\_support)

print("Existen {} peliculas con mas de {} favorable valoraciones".format(len(frequent\_itemsets[1]), min\_support))

sys.stdout.flush()

for k in range(2, 5):

cur\_frequent\_itemsets = find\_frequent\_itemsets(favorable\_reviews\_by\_users, frequent\_itemsets[k-1],

min\_support)

if len(cur\_frequent\_itemsets) == 0:

print("No se encuentran conjuntos de elementos frecuentes de tamanio {}".format(k))

sys.stdout.flush()

break

else:

print("Se encontraron {} conjuntos de elementos frecuentes de tamanio {}".format(len(cur\_frequent\_itemsets), k))

#print(cur\_frequent\_itemsets)

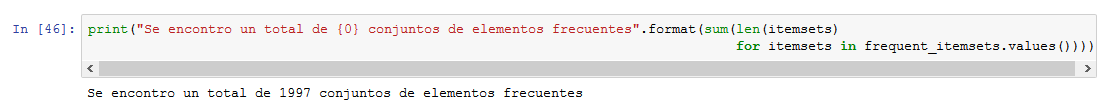
sys.stdout.flush()

frequent\_itemsets[k] = cur\_frequent\_itemsets

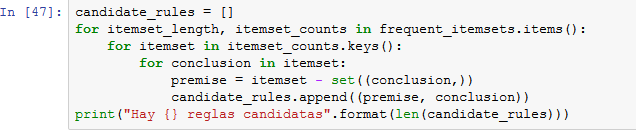
# Como no estamos interesados en conjuntos de tamanio 1, los removemos

del frequent\_itemsets[1]

########################################



Ahora creamos las reglas de asociacion. Todas son reglas candidatas, hasta que se verifica su confiabilidad



Cual es la salida para la siguiente instrucción??



Que significa?

Ahora copie el siguiente fragmento en la siguiente celda del cuaderno

###################################################

correct\_counts = defaultdict(int)

incorrect\_counts = defaultdict(int)

for user, reviews in favorable\_reviews\_by\_users.items():

for candidate\_rule in candidate\_rules:

premise, conclusion = candidate\_rule

if premise.issubset(reviews):

if conclusion in reviews:

correct\_counts[candidate\_rule] += 1

else:

incorrect\_counts[candidate\_rule] += 1

rule\_confidence = {candidate\_rule: correct\_counts[candidate\_rule] / float(correct\_counts[candidate\_rule] + incorrect\_counts[candidate\_rule])

for candidate\_rule in candidate\_rules}

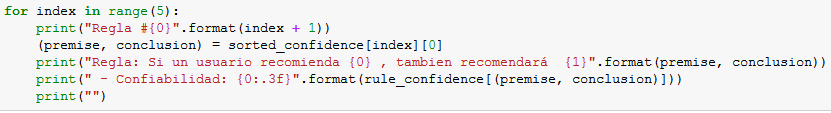
###################################################

Definimos el umbral minimo para la confiabilidad:









## Opcional::

Hasta aquí se tiene la implementación del algoritmo para la extracción de reglas de asociación apartir de los elementos frecuentes. Sin embargo, Si se quiere tener información de las películas asociadas a su identificador, para que las reglas sean un poco mas legibles, tendremos que leer el archivo donde se encuentran los detalles de las películas:

movies\_filename = "movies.csv"

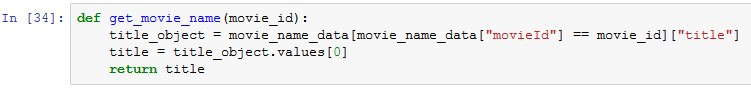
movie\_name\_data = pd.read\_csv(movies\_filename, delimiter=",")

movie\_name\_data.head(5)

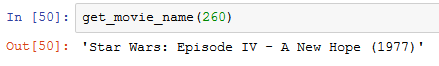
Podemos hacer una transformacion a variables dummy(binarias) apartir de los distintos generos de la pelicula.



Ahora incluimos una function de busqueda, dado el id de la película (movieId).



Probamos una busqueda..



Y finalmente reescribimos la impression de las reglas de asociacion indicando el nombre de la película, y no el id.

##########################################################

for index in range(5):

print("Regla #{0}".format(index + 1))

(premise, conclusion) = sorted\_confidence[index][0]

premise\_names = ", ".join(get\_movie\_name(idx) for idx in premise)

conclusion\_name = get\_movie\_name(conclusion)

print("Regla: Si un usuario recomienda {0} , tambien recomendará {1}".format(premise\_names, conclusion\_name))

print(" - Confiabilidad: {0:.3f}".format(rule\_confidence[(premise, conclusion)]))

print("")

#############################################################