Cervantes Rubí Brandon

Objetivo.

Obtener reglas de asociación a partir de datos obtenidos de una plataforma de películas, donde los clientes pueden rentar o comprar este tipo de contenidos.

Características.

- Por lo general, existe un patrón en lo que ven los clientes. Por ejemplo, superhéroes en la categoría para niños.
- En este sentido, se pueden generar más ganancias, si se puede identificar la relación entre las películas. Esto es, si las películas A y B se rentan juntas, este patrón se puede aprovechar para aumentar las ganancias.
- Las personas que rentan una de estas películas pueden ser empujadas a rentar o comprar la otra, a través de campañas o sugerencias dentro de la plataforma.
- En este sentido, cada vez es común familiarizarse con los motores de recomendación en Netflix, Amazon, por nombrar los más destacados.

Desarrollo.

La práctica fue dividida en 4 partes fundamentales, la primera de ellas es la importación e instalación de las bibliotecas necesarias dentro de las cuales se encuentra la biblioteca `apyori` siendo ésta una implementación para Python del algoritmo apriori.

```
!pip install apyori  # Se instala paquete Apyori

import pandas as pd                 # Para la manipulación y análisis de los datos
import numpy as np                  # Para crear vectores y matrices n dimensionales
import matplotlib.pyplot as plt     # Para la generación de gráficas a partir de los
datos
from apyori import apriori
```

También se utilizaron las bibliotecas `pandas` para el manejo de los datos, `numpy` para trabajar con vectores/matrices más cómodamente y por último el módulo `pyplot` de `matplotlib` para graficar los resultados del análisis. Además de que por buenas prácticas se utilizó un alias para cada una de estas bibliotecas con la palabra reservada `as`.

La segunda parte fundamental en la práctica es la importación de los datos, para esto se usó el módulo `colab` de la librería de `google` para trabajar con un archivo alojado en nuestra computadora y se leyó con la librería de pandas devolviéndonos un `dataframe` el cual es un tipo de dato similar a una tabla de SQL con renglones y columnas etiquetados.

	The Revenant	13 Hours	Allied	Zootopia	Jigsaw	Achorman	Grinch	Fast and Furious	Ghostbusters	Wolverine	Mad Max	John Wick	La La Land
0	Beirut	Martian	Get Out	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Deadpool	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	X-Men	Allied	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	Ninja Turtles	Moana	Ghost in the Shell	Ralph Breaks the Internet	John Wick	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	Mad Max	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...
7454	Big Sick	Looper	Hulk	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Al mostrar esta tabla podemos ver, primero que nada, el número de registros el cual supera los 7450. Segundo, los registros que contengan `NaN` significaran que el espacio en la tabla no tiene ningún valor asociado por lo cual es necesario que se remuevan. Tercero, podemos apreciar que los nombres de cada columna no hacen sentido pues el archivo `.csv` que se leyó no contenía en su estructura los nombres de las columnas, sino que eran puros registros por lo cual se tuvo que manipular este `dataframe` para contemplar todos los registros en el algoritmo.

```
DatosMovies = pd.read_csv('movies.csv', header=None)
#Para que el encabezado no se lea como los nombres de las columnas
DatosMovies.head(10)
#DatosMovies
```

Para corregir el error, simplemente al leer los datos del archivo eliminamos el `header` o encabezado (Con la opción `header=None`) que representa el nombre de las columnas para que nos muestre una tabla solamente con registros.

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	The Revenant	13 Hours	Allied	Zootopia	Jigsaw	Achorman	Grinch	Fast and Furious	Ghostbusters	Wolverine	Mad Max	John Wick	La La Land
1	Beirut	Martian	Get Out	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	Deadpool	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	X-Men	Allied	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Después llegamos a la etapa del procesamiento de los datos la cual va a tener como objetivo la exploración y preparación de los datos, es decir observar la distribución de frecuencia de los elementos para después moldear la forma de las matrices de datos que servirán como entrada para el algoritmo.

Lo primero que se hará en esta etapa será transformar la matriz de datos, originalmente se tiene una de dimensión 7460 x 20 la cual se volverá una matriz de dimensión 149200 x 1 (Donde 7460 x 20 = 149 200) a la que fácilmente podemos manejar como una lista. Esta lista de nuevo se volverá una Dataframe para poder aprovechar sus beneficios. Se creará una nueva columna llamada frecuencia donde cada una de las películas tendrá asociada su nivel de visualizaciones en base al número total de registros del conjunto de datos.

```
#Se agrupa los elementos
ListaM = ListaM.groupby(by=[0], as_index=False).count().sort_values(by=['Frecuencia'], ascending=True)
ListaM['Porcentaje'] = (ListaM['Frecuencia'] / ListaM['Frecuencia'].sum()) #Porcentaje
ListaM = ListaM.rename(columns={0 : 'Item'}) #Renombrar la columna
```

`ListaM` en este momento es el `dataframe` que contiene a todos los registros verticalmente donde estos se repiten, se eliminara esa redundancia en los datos agrupándolos con el método group by del `dataframe`, con `by = 0` le decimos que agrupe conforme a la columna 0, con `as_index = false` le decimos que no queremos que nos regrese los datos etiquetados como normalmente lo haría ya que nosotros estamos manejando esas etiquetas como columnas, contándolos al final. Además, se indica con `sort_values` que se ordene con respecto los valores de la columna de frecuencia de forma ascendente.

Al final se agregará una columna nueva llamada `Porcentaje`, en ella a cada registro se le asocia su porcentaje de distribución de frecuencia o frecuencia relativa dividiendo su respectiva frecuencia entre el número total de datos y se renombrará la columna 0 por `Items` ya que corresponde a cada película en el conjunto de datos. Obteniendo al final la siguiente tabla.

	Item	Frecuencia	Porcentaje
106	Vampire in Brooklyn	3	0.000102
63	Lady Bird	5	0.000171
34	Finding Dory	7	0.000239
11	Bad Moms	14	0.000477
118	water spray	29	0.000989
...	...	...	...
25	Coco	1229	0.041915
44	Hotel Transylvania	1280	0.043655
103	Tomb Rider	1305	0.044507
37	Get Out	1346	0.045906
75	Ninja Turtles	1786	0.060912

119 rows × 3 columns

Observando la tabla podemos decir que en total existe 119 películas diferentes en el conjunto de datos donde la más vista es Ninja Turtles con 1786 visualizaciones y un 6% de frecuencia relativa en comparación con Vampire in Brooklyn con 3 visualizaciones que indican un 0.01% del total de la frecuencia relativa en los datos.

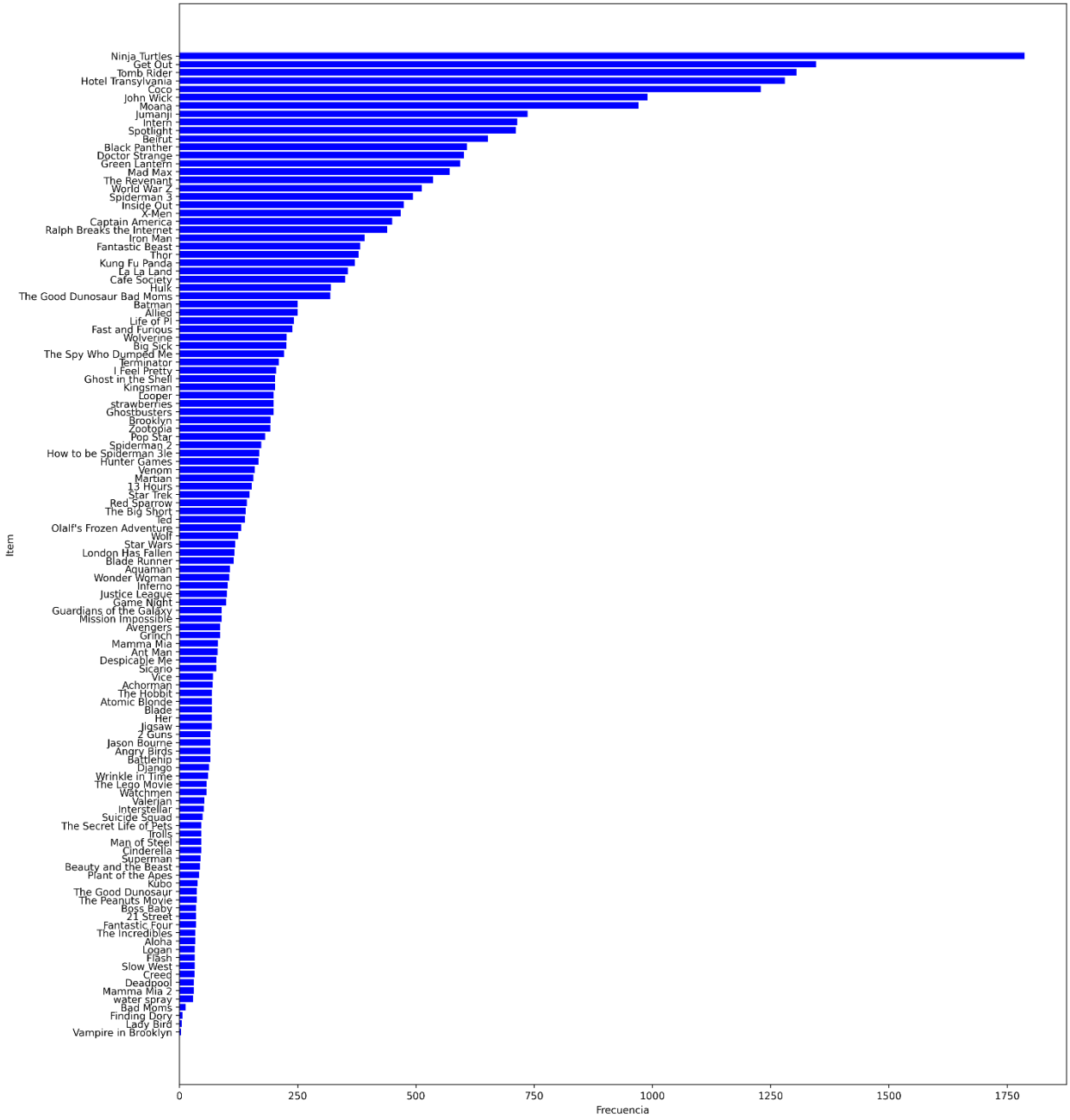
Con esto en mente se creó una gráfica para poder visualizar un panorama general de las visualizaciones para todas las películas en esta tabla. La grafica se realizó con la librería `matplotlib`, definimos la resolución con el método `figure`, con `ylabel` y `xlabel` le damos un nombre a cada uno de los ejes y la forma se le da con `barh` que hace una gráfica de barras horizontal pasándole la columna del `dataframe` que nos interesa (`Item`) como primer argumento para el eje Y, la frecuencia como segundo argumento para el valor en el eje X y el color con el que queremos que nos grafique todos los elementos. Por último, con `show` mostramos la gráfica.

```
# Se genera un gráfico de barras
plt.figure(figsize=(16,20), dpi=300) #Tamaño y resolucion
plt.ylabel('Item')
plt.xlabel('Frecuencia')
plt.barh(ListaM['Item'], width=ListaM['Frecuencia'], color='blue')
plt.show()
```

Los resultados se muestran en la siguiente página.

Cervantes Rubí Brandon

Inteligencia Artificial 2022-2, G.3, 02/03/2022
Practica No.1 Reglas de Asociación (Algoritmo Apriori)



En la última etapa reformamos nuestro conjunto de datos de tal forma que el `dataframe` se convierta en una lista de listas la cual requiere el algoritmo `apriori` para poder funcionar correctamente.

```
#Se crea una lista de listas a partir del dataframe y se remueven los 'NaN' con stack
MoviesLista = DatosMovies.stack().groupby(level=0).apply(list).tolist()
MoviesLista
```

Por último, viene la etapa de la aplicación del algoritmo con la función `apriori` que importamos previamente la cual recibe como parámetros la lista de listas que preparamos previamente, el soporte mínimo (0.01 o 1%), el nivel de confianza mínimo (0.3) y la elevación mínima (2) propias del algoritmo para que todas las reglas que cumplan con estos niveles sean consideradas como significativas.

```
ReglasC1 = apriori(MoviesLista,
                   min_support=0.01,
                   min_confidence=0.3,
                   min_lift=2)
```

La función regresa directamente los resultados del algoritmo con el tipo de dato `generator`, por lo cual se tiene que transformar a una lista de listas para poder apreciarlo mejor o bien visualizarlo como si fuera un `dataframe` para que se muestre una tabla con cada una de las reglas, así como sus niveles de confianza, soporte y elevación.

Como podemos ver en la siguiente tabla se obtienen 9 reglas con la configuración inicial.

```
pd.DataFrame(ResultadosC1) #Visualizacion con un data frame
```

	items	support	ordered_statistics
0	(Kung Fu Panda, Jumanji)	0.016086	[((Kung Fu Panda), (Jumanji), 0.32345013477088...
1	(Tomb Rider, Jumanji)	0.039410	[((Jumanji), (Tomb Rider), 0.3994565217391304,...
2	(Thor, Moana)	0.015282	[((Thor), (Moana), 0.3007915567282322, 2.31092...
3	(Terminator, Tomb Rider)	0.010322	[((Terminator), (Tomb Rider), 0.36492890995260...
4	(Get Out, Ninja Turtles, Jumanji)	0.010188	[((Get Out, Jumanji), (Ninja Turtles), 0.50666...
5	(Moana, Intern, Ninja Turtles)	0.011126	[((Intern, Ninja Turtles), (Moana), 0.30970149...
6	(Ninja Turtles, Moana, Jumanji)	0.011126	[((Moana, Jumanji), (Ninja Turtles), 0.5030303...
7	(Ninja Turtles, Tomb Rider, Jumanji)	0.017158	[((Ninja Turtles, Jumanji), (Tomb Rider), 0.41...
8	(Tomb Rider, Ninja Turtles, Spiderman 3)	0.010322	[((Ninja Turtles, Spiderman 3), (Tomb Rider), ...

La primera de ellas tiene que ver con las películas Kung Fu Panda y Jumanji, donde podemos catalogar ambas películas como películas del género familiar saliendo además en años similares 2016 y 2017.

En cuanto a las métricas podemos decir que la regla tiene un soporte de 1.6% lo cual indica que se trata de una regla importante dentro del conjunto de datos, una confianza del 32.3% que indica que dentro del conjunto de datos se trata de una regla fiable y por último se obtuvo un valor de elevación de 3.28 indicando el nivel de relación entre las películas evaluadas, es decir que existirán 3 veces más posibilidades de que los que vean Kung Fu Panda también miren Jumanji y viceversa.

	items	support	ordered_statistics
0	(Get Out, Beirut)	0.028954	[((Beirut), (Get Out), 0.3312883435582822, 1.8...
1	(Coco, Ninja Turtles)	0.052949	[((Coco), (Ninja Turtles), 0.32166123778501626...
2	(Intern, Ninja Turtles)	0.035925	[((Intern), (Ninja Turtles), 0.375350140056022...
3	(Ninja Turtles, Jumanji)	0.041153	[((Jumanji), (Ninja Turtles), 0.41711956521739...
4	(Tomb Rider, Jumanji)	0.039410	[((Jumanji), (Tomb Rider), 0.3994565217391304,...
5	(Moana, Ninja Turtles)	0.048257	[((Moana), (Ninja Turtles), 0.3707518022657054...
6	(Spotlight, Ninja Turtles)	0.033914	[((Spotlight), (Ninja Turtles), 0.355337078651...
7	(Tomb Rider, Ninja Turtles)	0.060054	[((Tomb Rider), (Ninja Turtles), 0.34329501915...

Para la segunda configuración se definió un soporte mínimo más estricto de 2.8%, un nivel mínimo de confianza igual a 0.3 cono en la configuración anterior y una elevación mínima de 1.1 siendo un poco más flexible en esta métrica en comparación a la anterior, los resultados se muestran a continuación.

Se observa que se generaron 8 reglas después de la poda las cuales cumplen con estos parámetros, además de que podemos apreciar que los títulos se repiten, por ejemplo, Nija Turtles, Jumanji, Get Out, Tomb Raider, etc. esto debido a que todas estas películas pertenecen al grupo que genera más visualizaciones dentro del conjunto de datos.

Analizando la primera de esas reglas, tenemos la relación entre Get Out y Beirut. La cual tiene como resultados, un soporte del 2.9%, una confianza del 33% y una elevación de 1.83 lo que nos representa que aumenta en 2 veces más la posibilidad de que los usuarios que vean Get Out también vean la película de Beirut y viceversa.

Cervantes Rubí Brandon

Configuración propuesta.

Para el soporte mínimo se consideraron películas que hayan sido visualizadas por lo menos 175 veces a la semana o 25 veces por día resultando en un soporte mínimo de 2.3% (Es decir 175/7460 = 0.023), además de una confianza mínima de 25% y una elevación mínima de 1.7 para considerarse una regla significativa.

Configuración que dio como resultado 6 reglas de asociación, las cuales se muestran a continuación.

	items	support	ordered_statistics
0	(Get Out, Beirut)	0.028954	[((Beirut), (Get Out), 0.3312883435582822, 1.8...
1	(Ninja Turtles, Jumanji)	0.041153	[((Jumanji), (Ninja Turtles), 0.41711956521739...
2	(Tomb Rider, Jumanji)	0.039410	[((Jumanji), (Tomb Rider), 0.3994565217391304,...
3	(Ninja Turtles, Spiderman 3)	0.027748	[((Spiderman 3), (Ninja Turtles), 0.4190283400...
4	(Thor, Ninja Turtles)	0.023190	[((Thor), (Ninja Turtles), 0.45646437994722955...
5	(Tomb Rider, Spiderman 3)	0.023056	[((Spiderman 3), (Tomb Rider), 0.3481781376518...

Tomare la sexta regla (Con índice 5), la cual nos arroja la relación que existe entre Tomb Raider y Spiderman 3 cosa que tiene sentido debido a que por lo general a las personas que les gustan películas del género de acción o aventura como Tomb Raider por lo general también les atraen las películas del género de ciencia ficción como Spiderman 3.

Podemos observar que el soporte es de 2.3 lo cual indica que se trata de una regla importante dentro del conjunto de datos, una confianza del 34% porcentaje que respalda la fiabilidad de la regla y una elevación de 1.99 representando que se duplican las posibilidades de que los usuarios que vean Tomb Raider vean también Spidermam 3 o viceversa.

Además, en esta prueba se observó que si se baja soporte en el algoritmo dará paso a que nuevas películas entren en el conjunto de reglas del sistema de recomendación ya que esto significa que también se baja la métrica de visualizaciones que las películas debieron de haber generado, dando lugar a que el sistema de recomendaciones no solamente recomiende las películas más vistas de la plataforma y haciendo que los usuarios puedan descubrir nuevas franquicias.

Conclusiones

Las reglas de asociación nos sirven para poder modelar relaciones entre un conjunto de transacciones del mismo tipo, por ejemplo, la lista de películas que un usuario visualiza en alguna plataforma de streaming. Con base en esas relaciones podremos hacer recomendaciones para generar interés.

Para poder lograr esto es necesario contemplar todas las posibles reglas, las cuales incrementan en número de forma exponencial conforme incrementa el número de transacciones y es por eso por lo que el algoritmo apriori nos puede ayudar a eliminar reglas que no sean significativas o importantes definiendo métricas que se deben de cumplir para cada relación. La primera de ellas es el soporte que indica lo importante que es la regla en el total de transacciones, la segunda es la confianza que indica que tan fiable es la regla a considerar para estas dos consideraciones se emplea un porcentaje para medirlas.

Por último, está la elevación el cual es un margen que se debe de superar para indicar el incremento de posibilidades de que si se cumple la primera condición de la regla también se cumpla la segunda, en este caso, si el usuario visualiza la primera película, también pueda visualizar las recomendadas por el sistema.