Universidad Nacional
Autónoma de México

Inteligencia Artificial

Reglas de asociación

Grupo 3

Nombre:
Barreiro Valdez Alejandro

Práctica 1

Profesor: Dr. Guillermo Gilberto Molero Castillo

2 de marzo de 2022

Introducción

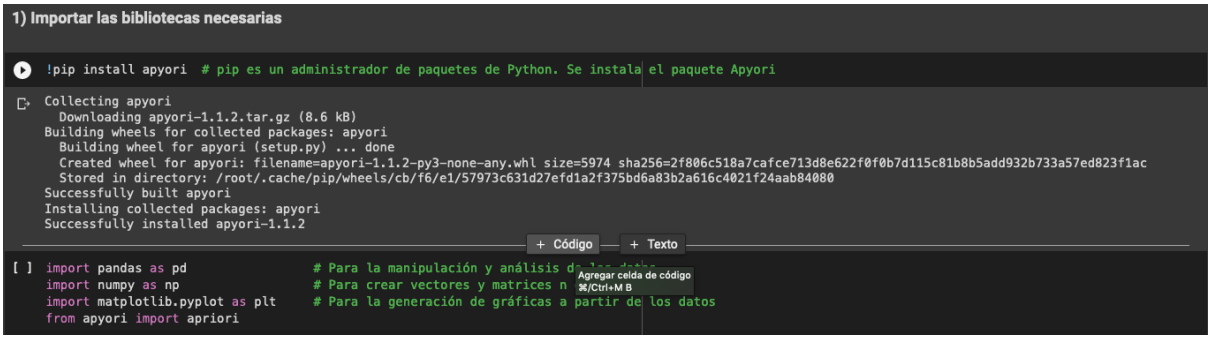
Para esta práctica se generará un sistema de recomendación de películas utilizando reglas de asociación para encontrar patrones en la manera que se ven películas y encontrar qué películas son más afines a otras. Se utilizará el algoritmo Apriori para generar reglas significativas con soporte, confianza y elevación. A partir de un conjunto de datos con las películas que renta un grupo de personas se generarán reglas utilizadas para un sistema de recomendación utilizando el algoritmo Apriori. Se generarán tres configuraciones diferentes de reglas de recomendación y explicarán las características de cada una.

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.

Desarrollo

El primer aspecto que se desarrolló fue importar las bibliotecas necesarias para la realización de esta práctica. Se importa el módulo para el algoritmo a utilizar, pandas, numpy y matplotlib.



Lo siguiente que se desarrolló fue la lectura de los datos a utilizar. Se subió el archivo a Google Colab y utilizando pandas se realizó la lectura del csv. Además, se obtuvo una vista de la tabla leída donde se pudo observar que se tenían muchos NaN.

2) Importar los datos

Fuente de datos: movie_dataset.csv

```
[ ] from google.colab import files
    files.upload()
```

```
Examinar...  Ningún archivo seleccionado. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving movies.csv to movies.csv
{'movies.csv': b"The Revenant,13 Hours,Allied,Zootopia,Jigsaw,Achorman,Grinch,Fast and Furious,Ghostbusters,Wolverine,Mad Max,John Wick,La La Land,The Good
```

```
[ ] DatosMovies = pd.read_csv('movies.csv')
```

```
DatosMovies
```

	The Revenant	13 Hours	Allied	Zootopia	Jigsaw	Achorman	Grinch	Fast and Furious	Ghostbusters	Wolverine	Mad Max	John Wick	La La Land	The Good Dunosaur	Ninja Turtles	The Good Dunosaur Bad Moms	2 Guns	Inside Out
0	Beirut	Martian	Get Out	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	Deadpool	NaN	NaN	NaN	NaN	NaN	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	NaN	NaN	NaN	NaN	NaN

Para procesar dichos datos se incluyeron todos los datos en una sola lista y a partir de dicha lista se creó una matriz con una columna llamada frecuencia.

```
[ ] #Se incluyen todas las transacciones en una sola lista
    Transacciones = DatosMovies.values.reshape(149200).tolist() #-1 significa 'dimensión no conocida'
```

```
[ ] #Se crea una matriz (dataframe) usando la lista y se incluye una columna 'Frecuencia'
    ListaM = pd.DataFrame(Transacciones)
    ListaM
```

	0
0	The Revenant
1	13 Hours
2	Allied
3	Zootopia
4	Jigsaw
...	...
149195	NaN
149196	NaN
149197	NaN
149198	NaN
149199	NaN

149200 rows × 1 columns

```
[ ] ListaM['Frecuencia'] = 0
    ListaM
```

Posteriormente, se agruparon los elementos para generar un conteo de las veces que aparece cada elemento y el porcentaje en el que cada elemento aparece. De esta manera, se tiene una tabla donde se muestran todas las películas vistas con un conteo de cuantas veces se contaron.

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

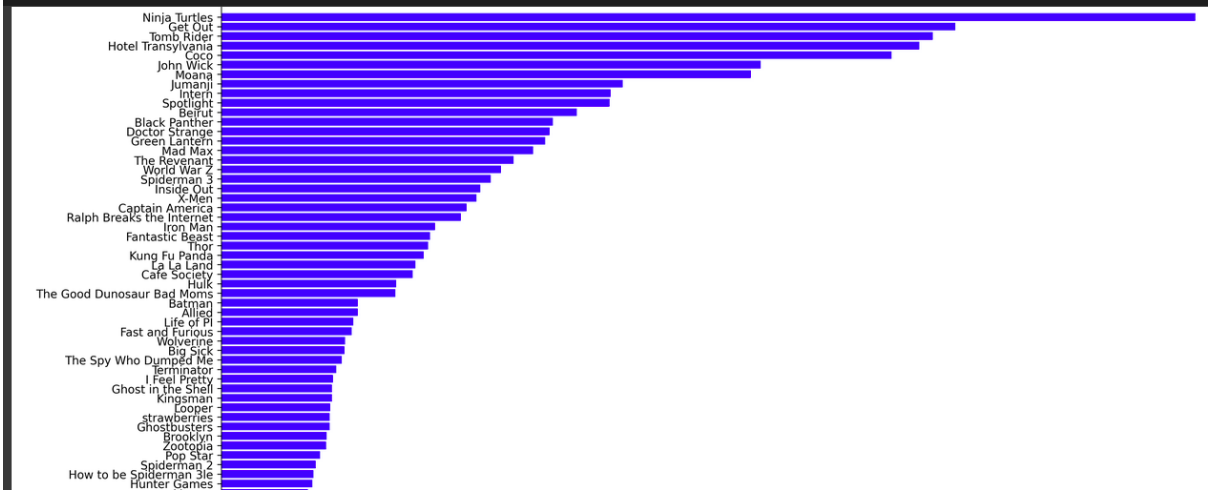
```
ListaM
```

	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

A partir de este conteo y esta matriz se generó una gráfica que utiliza las frecuencias para mostrar de manera gráfica el número de veces que se vio cada una de las películas.

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



Para la preparación de la utilización del algoritmo Apriori se requiere una forma de lista de listas. Se crea una lista de listas a partir de los datos anteriores y se quitan

todos los datos que representen un NaN. A partir de estos datos ya se puede aplicar el algoritmo para obtener un sistema de recomendaciones.

```
[ ]  #Se crea una lista de listas a partir del dataframe y se remueven los 'NaN'
     #level=0 especifica desde el primer índice
     MoviesLista = DatosMovies.stack().groupby(level=0).apply(list).tolist()
     MoviesLista
```

```
[['The Revenant',
  '13 Hours',
  'Allied',
  'Zootopia',
  'Jigsaw',
  'Achorman',
  'Grinch',
  'Fast and Furious',
  'Ghostbusters',
  'Wolverine',
```

Para la aplicación del algoritmo se crearon dos configuraciones para obtener diferentes métodos de recomendaciones. Para la primera configuración se desean películas que se hayan visto por lo menos 70 veces a la semana. Esto va de la mano con que la frecuencia con la que se haya visto es del 1% con una confianza mínima del 30% y elevación de 2. Estos valores se alimentan al algoritmo y las reglas generadas por el algoritmo se convierten en una lista. Se imprime el número de reglas encontradas, en este caso nueve, y la lista que se generó.

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

onvierte las reglas encontradas por la clase apriori en una lista, puesto que es más fácil ver los resultados.

```
ResultadosC1 = list(ReglasC1)
print(len(ResultadosC1)) #Total de reglas encontradas
```

```
9
```

```
ResultadosC1
```

```
[RelationRecord(items=frozenset({'Kung Fu Panda', 'Jumanji'}), support=0.0160857908847185, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Kung
 RelationRecord(items=frozenset({'Tomb Rider', 'Jumanji'}), support=0.03941018766756032, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Jumanj
 RelationRecord(items=frozenset({'Moana', 'Thor'}), support=0.015281501340482574, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Thor'}), items
 RelationRecord(items=frozenset({'Tomb Rider', 'Terminator'}), support=0.01032171581769437, ordered_statistics=[OrderedStatistic(items_base=frozenset({'Term
 RelationRecord(items=frozenset({'Ninja Turtles', 'Jumanji', 'Get Out'}), support=0.010187667560321715, ordered_statistics=[OrderedStatistic(items_base=froz
 RelationRecord(items=frozenset({'Ninja Turtles', 'Moana', 'Intern'}), support=0.011126005361930294, ordered_statistics=[OrderedStatistic(items_base=frozens
 RelationRecord(items=frozenset({'Ninja Turtles', 'Jumanji', 'Moana'}), support=0.011126005361930294, ordered_statistics=[OrderedStatistic(items_base=frozen
 RelationRecord(items=frozenset({'Ninja Turtles', 'Jumanji', 'Tomb Rider'}), support=0.017158176943699734, ordered_statistics=[OrderedStatistic(items_base=
 RelationRecord(items=frozenset({'Ninja Turtles', 'Spiderman 3', 'Tomb Rider'}), support=0.01032171581769437, ordered_statistics=[OrderedStatistic(items_bas
```

Para entender la lista generada se imprimió una regla donde se relaciona *Kung Fu Panda* y *Jumanji*. Esta regla tiene sentido porque ambas son películas familiares y fueron muy vistas según los datos presentados. En la lista también se presentan los datos sobre el soporte, la confianza y la elevación de cada regla. A partir de estos datos y este formato se imprime cada una de las reglas que se generaron utilizando

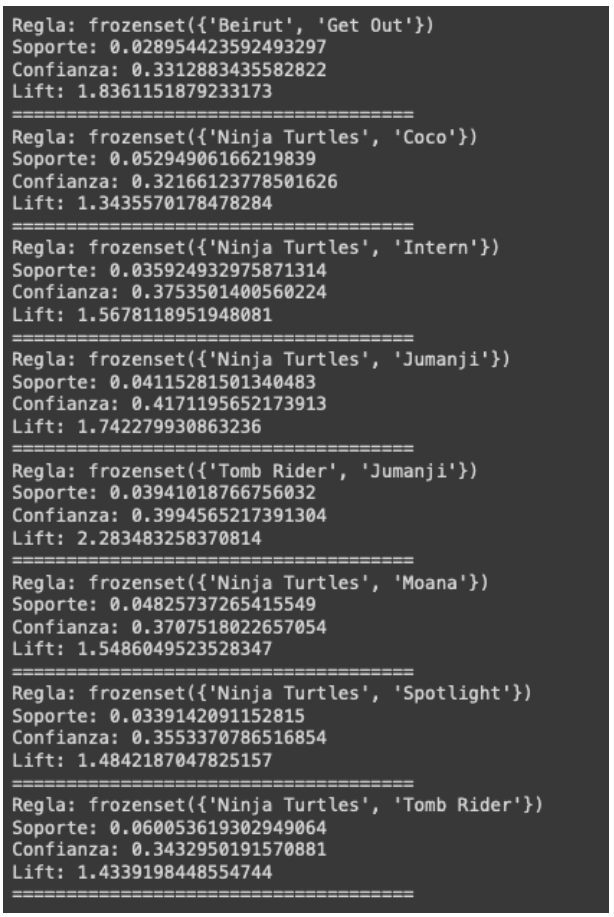
un ciclo que imprime cada uno de los datos con un formato. Se muestra el código de la función y las reglas generadas en ese formato.

```
for item in ResultadosC1:
  #El primer índice de la lista
  Emparejar = item[0]
  items = [x for x in Emparejar]
  print("Regla: " + str(item[0]))

  #El segundo índice de la lista
  print("Soporte: " + str(item[1]))

  #El tercer índice de la lista
  print("Confianza: " + str(item[2][0][2]))
  print("Lift: " + str(item[2][0][3]))
  print("====================================")
```

```
Regla: frozenset({'Kung Fu Panda', 'Jumanji'})
Soporte: 0.0160857908847185
Confianza: 0.3234501347708895
Lift: 3.2784483768897226
====================================
Regla: frozenset({'Tomb Rider', 'Jumanji'})
Soporte: 0.03941018766756032
Confianza: 0.3994565217391304
Lift: 2.283483258370814
====================================
Regla: frozenset({'Moana', 'Thor'})
Soporte: 0.015281501340482574
Confianza: 0.3007915567282322
Lift: 2.3109217437617016
====================================
Regla: frozenset({'Tomb Rider', 'Terminator'})
Soporte: 0.01032171581769437
Confianza: 0.36492890995260663
Lift: 2.0861070254762035
====================================
Regla: frozenset({'Ninja Turtles', 'Jumanji', 'Get Out'})
Soporte: 0.010187667560321715
Confianza: 0.5066666666666666
Lift: 2.1163120567375886
====================================
Regla: frozenset({'Ninja Turtles', 'Moana', 'Intern'})
Soporte: 0.011126005361930294
Confianza: 0.30970149253731344
Lift: 2.37937500960696
====================================
Regla: frozenset({'Ninja Turtles', 'Jumanji', 'Moana'})
Soporte: 0.011126005361930294
Confianza: 0.5030303030303029
Lift: 2.1011232142251175
====================================
Regla: frozenset({'Ninja Turtles', 'Jumanji', 'Tomb Rider'})
Soporte: 0.017158176943699734
Confianza: 0.4169381107491857
Lift: 2.383416326581552
====================================
Regla: frozenset({'Ninja Turtles', 'Spiderman 3', 'Tomb Rider'})
Soporte: 0.01032171581769437
Confianza: 0.3719806763285024
Lift: 2.1264182723453087
====================================
```

Se creó una segunda configuración donde las reglas que se ponen son películas que se hayan visto al menos 210 veces a la semana, con una confianza mínima de 30% y una elevación mayor a uno. Para este algoritmo se realizaron los mismos pasos que para la anterior configuración y se obtuvo una lista de las reglas que propone el algoritmo. En este caso la primera regla relaciona *Beirut* y *Get Out*. Una es una película de terror y la otra de acción de años parecidos, por esto mismo esta

regla tiene sentido. De la misma manera que en la configuración pasada se muestran las reglas generadas por este algoritmo y las características de cada una.



Se dejó como ejercicio individual crear una tercera configuración. Para dicha configuración se elevó el número de veces visto en un día a 42. También se modificó la elevación a mayor de 1.5. Se crearon las reglas y se obtuvo el número de reglas generadas. Se obtuvo la vista de la lista generada con las reglas y se convirtieron al tipo de dato Data Frame.

```
[28] ReglasC3 = apriori(MoviesLista,
                        min_support=0.039,
                        min_confidence=0.3,
                        min_lift = 1.51)

[29] ResultadosC3 = list(ReglasC3)
     print(len(ResultadosC3))

     3

[30] ResultadosC3

     [RelationRecord(items=frozenset({'Ninja Turtles', 'Jumanji'}), support=0.04115281501340483, ordered_statistics=[OrderedStatistic(items_base=fr
      RelationRecord(items=frozenset({'Tomb Rider', 'Jumanji'}), support=0.03941018766756032, ordered_statistics=[OrderedStatistic(items_base=froze
      RelationRecord(items=frozenset({'Moana', 'Ninja Turtles'}), support=0.04825737265415549, ordered_statistics=[OrderedStatistic(items_base=froz
```

```
pd.DataFrame(ResultadosC3)
```

	items	support	ordered_statistics
0	(Ninja Turtles, Jumanji)	0.041153	[((Jumanji), (Ninja Turtles), 0.41711956521739...
1	(Tomb Rider, Jumanji)	0.039410	[((Jumanji), (Tomb Rider), 0.3994565217391304,...
2	(Moana, Ninja Turtles)	0.048257	[((Moana), (Ninja Turtles), 0.3707518022657054...

La primera regla relaciona *Ninja Turtles* y *Jumanji*. Lo realiza con un soporte del 4.1%, una confianza del 41% y una elevación de 1.74. Esto corresponde a los parámetros que se solicitaron y tiene sentido ya que ambas películas son infantiles y fueron muy vistas. La razón por la que no se obtuvieron tantas reglas fue porque no se tienen tantas opciones con los parámetros mencionados. Esto también tiene sentido ya que se puso un número alto de veces vistas a la semana por lo que las películas relacionadas serán solo unas cuantas. Por último, se imprimió cada una de las reglas generadas con las características dentro del algoritmo que tiene cada una. Las tres reglas generadas tienen sentido.

```
Regla: frozenset({'Ninja Turtles', 'Jumanji'})
Soporte: 0.04115281501340483
Confianza: 0.4171195652173913
Lift: 1.742279930863236
=====================================
Regla: frozenset({'Tomb Rider', 'Jumanji'})
Soporte: 0.03941018766756032
Confianza: 0.3994565217391304
Lift: 2.283483258370814
=====================================
Regla: frozenset({'Moana', 'Ninja Turtles'})
Soporte: 0.04825737265415549
Confianza: 0.3707518022657054
Lift: 1.5486049523528347
=====================================
```

Conclusión

Para esta práctica se lograron generar varias reglas de asociación en tres configuraciones distintas a partir de datos que se obtuvieron de una plataforma de películas. En cada una de las configuraciones se logró obtener reglas y se pudieron observar las características de cada una de estas reglas. Se pudo observar el

soporte, la confianza y la elevación de cada una de estas reglas. A partir de cada una de las reglas generadas se puede generar un sistema de recomendaciones. Además de utilizar el algoritmo Apriori y de generar reglas de recomendación se pudo utilizar Google Colab y se procesaron los diferentes datos utilizados para esta práctica. Se tuvo una introducción a la manera en que se trabaja en las Notebooks de Python.