Sistema de recomendación de moda. H & M

Introducción

H&M Group es una familia de marcas y negocios con 53 mercados en línea y aproximadamente 4850 tiendas. Su tienda en línea ofrece a los compradores una amplia selección de productos para navegar. Pero con demasiadas opciones, es posible que los clientes no encuentren rápidamente lo que les interesa o lo que buscan y, en última instancia, es posible que no realicen una compra. Para mejorar la experiencia de compra, las recomendaciones de productos son clave.

H&M Group ha decidico lanzar una competencia para desarrollar recomendaciones de productos basadas en datos de transacciones anteriores, así como en metadatos de clientes y productos. Los metadatos disponibles abarcan desde datos simples, como el tipo de prenda y la edad del cliente, hasta datos de texto de descripciones de productos y datos de imágenes de prendas.

Un sistema de recomendación también mejora la experiencia del cliente, pues es mejor que navenguen en la red viendo productos por gusto y no porque se vean obligados a hacerlo porque no encuentran lo que quieren comprar.

Análisis exploario de datos (EDA)

Los datos propocionados por la compañía son:

- Imagenes de los productos
- Data set de clientes
- Data set de transacciones

Describir las características de los datos Describir cataracterísticas del data set de transacciones Concluir que hay ciertos tipos de productos que no se van a tomar en cuenta porque en relidad la variedad de los mismos es poca Generar data set aptos para el modelo

Definición conceptual de retroalimentación implicita

En análisis exploratorio de los datos muestran que no tenemos una retroalimentación

explicita de parte del usuario respecto de su compra. Generalmente existen dos tipos de sistemas de recomendación:

- **Sistemas de recomendación basados en contenido**. Busca crear perfiles para careacterizar la naturaleza de usuario o del producto[1]. Genera recomendaciones a partir de dicho perfil, por ejemplo a partir de gustos simlares, es decir, si el usuario busco zapatos, la idea sería recomendarle zapatos, o si el usuario buscó peliculas de terror la idea sería recomendarle películas de terror. Estos sistemas son muy usados en la actualidad, por ejemplo en redes sociales como Facebook que usualmente presenta anuncios basados en la busqueda de los usuarios.

- **Sistemas de recomencación por filtrado colaborativo**. El enfoque en este sistema de recomendación es distinto, pues se basa en el comportamiento de los usuarios. Bajo este enfoque no es necesario crear un perfil. Filtrado colaborativo analiza las relaciones entre usuarios e interdependencias entre productos, para identificar nuevas asociaciones usuario-producto[1].

La información en este tipo de sistemas puede ser explicícita o implicita. La pimera hace referencia a que se tiene una retroalimentación explícita del usuario acerca del producto, ej. ratings. La segunda hace referencia a que no se tiene un feeback explictio del usuario y la recomendación se basa más bien en el perfil de compara o uso del producto en si. **Para el enfoque de nuestro probelma en particular se hará referencia al comportamiento de compras de x o y producto, es decir sin tener un feedback de manera explícita**.

Sea r_{ui} los valores que asocian al usuario u con el item i. En nuesto caso r_{ui} representa el número de comparas que hizo el usuario u del producto i. Definimos p_{ui} como la variable que indica la preferencia del usuario u por el item i tal que:

$$p_{ui} = \begin{cases} 1, & \text{si } r_{ui} > 0 \\ 0, & \text{si } r_{ui} = 0 \end{cases}$$

Mientras mayor sea el valor de r_{ui} mayor será nuestra creencia de que al usuario u tiene preferencias por el artículo i, sin embargo, el hecho de que p_{ui} sea 0 no significa que al usuario no le guste o no tenga preferencia por el item i necesariamente. Esto se debe a que en realidad puede ser que la persona nuca haya visto el producto y por ello nunca lo ha comprado. Dicho lo anteriror definimos:

$$c_{ui} = 1 + \alpha r_{ui}$$

como la medida de confidencia de la observación p_{ui}. Con ello tenemos una confianza mínima en p_{ui} para cada usuario y producto. De forma que a medida que vemos más observaciones nuestra confidencia en p_{ui} crece. El objetivo será encontrar un vector x_u para cada usuario y un vector y_i para cada prodcuto tal que

$$p_{ui} = x_u^T y_i$$

Esta aproximación en factores latentes es la que se busca optimizar y deberá tomar en cuenta los distintos niveles de confianza c_{ui} y además deberá considerar todos los posibles pares u, i y no sólo los observados. Luego entonces, se busca que:

$$min \sum_{u,i} c_{ui}(p_{ui} - x_u^T y_i)^2 + \lambda(\sum_u ||x_u||^2 + \sum_i ||y_i||^2)$$

Donde el segundo término de la suma es necesario para regularizar el modelo tal que no exista sobreajuste. Finalmente, el método de optimización para este tipo de modelos es el de mínimos cuadrados alternados (ALS), por sus siglas en inglés.

Modelo propuesto

Con base en lo anterior nuestro probelma se traduce de la siguiente manera:

- u serán el _client*id* dentro de la tabla de transacciones
- i será la variable _article*id*
- r_{ui} será el número de compras. En este caso la variable _purchase*count* que se definio en la sección del EDA.

La implementación manual de este algoritmo resulta ser poco factible pues se tendría que construir una matriz de preferencias del orden de 145 000 millones de entradas producto de 1,371,980 usuarios y 105,534 items diferentes. Debido a la dificultad en el manejo de los datos y contando con en el hecho de que en spark existe una impplementación para el modelo de recomendación implicita hemos decidido usar dicha herramienta.

El primer modelo considera el conjunto de datos completo de la tabla de transacciones. Se considera un set de entrenamiento del 80% de lo datos y el restante se utiliza para evaluar la presición del modelo[2]. En el apendice de este documento, en la sección modelo 1 se puede observar el código utilizado en spark para configurar este primer modelo. Los valores que se tomaron en cuenta son:

- **rank**: hace referencia el número de factores latentes derivados de la descompsición de la matriz p_{ui}.
- **maxIter**: dado que el métod de optimización se hace a través de ALS, se define dentro de los parmétos del modelo el número máximo de iteraciones a considerar para obtener los valores que aproximen a la matriz p.
- **regParm**: hace referencia a λ que es el parametro de regularización dentro de la función de optimización.
- **alpha**: hac referencia al valor α que nos ayuda a definir nuestro nivel de creencias dentro del modelo.

La siguiente tabla muestra los posibles valores considerados para cada uno de los

parámetos del modelo:

Parameter	Val 1	Val 2	Val 3	Val 4
rank	10	20	30	40
maxIter	5	10	-	-
regParams	0.001	0.01	0.1	-
alphas	1	10	100	-

Se considera cada posible combinación de valores de la tabla anteiror, resultando en la iteración de 72 posibles modelos a evaluar. Como se verá en la sección de resultados, bajo este enfoque esta primer propuesta de midelo resulta ser deficiente en resultados e incluso en el performance al momento de calibrar cada posible combinación.

El resultado anterior plantea una solución diferente al problema. Actualmente, en muchas de las tiendas de ropa en línea cuando una persona busca un producto de ropa, zapatos por ejemplo, se tiende a mostrar items del mismo tipo, es decir, se tiende sugerir una gama distinta de zapatos. Lo mismo pasa con cualquier otro producto, pantalones por ejemplo y bajo ese contexto una segunda propuesta es calibrar un modelo por cada tipo de porducto.

Derivado de la tabla "articules" vista en la sección del EDA como se puede observar se tienen 18 posibles tipos de productos la idea es entonces generar un sistema de recomendación para cada uno de ellos. Para fines prácticos y por temas de recurso y timepo, en este estudio sólo plantea considerar generar modelos para los siguientes artículos:

- Germent Lower body (GLB)
- Accesories
- Underwear
- Shoes

Los parámetos considerados para calibrar cada modelo son los mismos que se mencionan en la tabla 1, esto quire decir que ahora por cada tipo de producto se probabrán 72 modelos diferentes dando un total de 288 iteraciones diferentes.

Evaluación del modelo y resultados

Cada modelo ajustado genera una predicción para los datos de prueba de la siguiente forma:

client_id	article_id	purchase_count	prediction
1	673396002	2	1.3635218
1	484398001	6	1.3175579

1	759871002	1	0.9576558
2	372860001	1	0.2375944

El valor de la variable *prediction* no busca aproximar la variable _purchase*count* sino que en realidad es una medida de la preferencia del artículo en cuestión y es por ello que [1] sugiere considerar la siguiente métrica a la cual denominamos ROEM (Rank ordering error metric) [3]:

$$ROEM = \frac{\sum_{u,i} r_{ui}^t rank_{ui}}{\sum_{u,i} r_{ui}^t}$$

Donde $rank_{ui}$ será el percentil de la i-ésima predicción odenada en forma descendente. Tomando como base la tabla anteriror, el ejmplo del cálculo del rank en este caso sería

client_id	article_id	purchase_count	prediction	rank	r*rank
1	673396002	2	1.3635218	0	0
1	484398001	6	1.3175579	0.5	3
1	759871002	1	0.9576558	1	1
2	372860001	1	0.2375944	1	1

Note que para el caso del cliente 1 el percentil más pequeño en el valor del *rank* se le asigna a la predicción más grande mientras que el percentil más grande se lo asigna a la predicción más pequueña. Finalmente el ROEM será el resultado de sumar la columna *r*rank* dividio por la suma total de la columna _purchase*count* que en este ejemplo de 40%. En escencia, lo que se esta haciendo es que para cada usuario estamos rankeando primero los items más importantes con base en la predicción, es por ello que mientras más cercano a cero se encuentre el ROEM mejor.

El cálculo del ROEM no esta implementado en spark, de modo que se hizo un calculo manual el tiene ciertas consideraciones:

- En el ejemplo anteior, cuando la combinación client_id - articule_id es única, el percentil asignado es 1 (valor rank =1).
- Es muy probable que en el conjunto de datos de validación exista una combinación de client_id - articule_id que no aparece en el set de entrenamiento, puesto que se definieron de forma aleatoria. Para esos casos, y para objeto de nuestro estudio, lo mejor será generar una propuesta de predicción aleatroria entre 0 y 1.

En este último punto, es importante hacer enfásis en que esta no es la mejor solución, pero el enfoque para resolver el problema de recomendación para esos usuarios y productos probablemente ya no sería un algoritmo de retroalimentación implicata. La aleatoriedad considera en estos usuarios no sesga nuestro problema, de hecho en el peor de los casos

nos diría que nuestro modelo no es bueno pues nos daría un ROEM cercano a 50% (si es que hay muchos valores nulos, por ejemplo).

La tabla 1 del apendice muestra algunos de los valores obtenidos para el ROEM en las distintas iteraciones.Cabe mencionar que por cuestiones técnicas y de tiempo se opto por buscar el modelo con el mejor resultado en la metrica ROEM para los datos de validación y posteriormente se hizo validación cruzada con el objetivo de comprobar que realmente no se tuviera overfitting. Se consideraron 5 folds, la siguiente tabla muestra el resultado del ROEM calculado en cada uno así como el ROEM de los mejores modelos ajustados:

Modelo	ROEM_f1	ROEM_f2	ROEM_f3	ROEM_f4	ROEM_f5	ROEM_final
GLB	25.49%	26.34%	25.62%	26.30%	26.13%	20.14%
Accesories	19.74%	20.13%	19.96%	20.18%	20.40%	22.14%
Underwear	24.73%	24.47%	25.11%	24.59%	24.65%	17.96%
Shoes	17.71%	17.39%	17.49%	17.45%	18.05%	18.10%

Finalmente, los parámetros en cada caso son:

Modelo	rank	maxIter	regParam	alpha
GLB	20	10	0.001	100
Accesories	10	5	0.001	100
Underwear	20	10	0.001	100
Shoes	10	10	0.001	100

Análisis de las recomendaciones (predicciones)

En la tabla 5 se observan los parametros definidos para cada modelo por tipo de producto. Para cada uno de ellos se ha cerado un modelo que nos permita generar predicciones de recomendación por tipo de producto. El nombre de los modelos que hemos desarrollado son:

- als_shoes
- als_underwear
- als_accesories
- als_glb

Dentro de pyspark se ha desarrollado un método para generar recomendaciones de productos basada en la proyección de los factores latentes que se han obtenido para cada modelo. El orden en que se muestran los rating proyectados es de forma decendente. Si

bien, no es posible mostrar en detalle

Imagen comparativo compro vs recomendacion para todos los modelos

Conclusiones

Mencionar que sería bueno ajustar el modelo considerando mayor recurso computacional, seguro que se obtendrían muy buenos resultados.

Apendice

Agregar el código utilizado (opcional)

In [ ]: