Trabajo final:
Sistema de recomendación para un Marketplace

Pablo Saldarriaga-Aristizabal, *Nicolás Prieto-Escobar*,
Victoria Álvarez-Restrepo, *Karen Velásquez-Moná*, *Ana Urán-González*
Maestría en Ciencia de los Datos y Analítica

Universidad EAFIT
Medellín – Colombia

1. Pregunta de investigación y objetivos

1.1. Pregunta de investigación

¿Cómo emplear reseñas y características de los productos de un marketplace para generar recomendaciones de otros artículos a usuarios en presencia de grandes volúmenes de datos?

En este trabajo, se implementó un sistema de recomendación para artículos de Amazon. Los modelos creados le sugieren a cada usuario los productos que es probable que le interesen, considerando tanto las reseñas que le ha dado a otros artículos que ha adquirido como también las características propias de los productos. El modelo fue construido usando Spark y desplegado en Amazon EMR, de forma tal que utiliza técnicas adecuadas para ser usado con grandes cantidades de datos, para que posteriormente pueda ser escalado al gran catálogo de productos que tiene Amazon. Dentro del trabajo realizado se compararon varias técnicas para obtener el mejor rendimiento posible, es decir, para realizar sugerencias adecuadas de productos (tanto existentes como nuevos) para los usuarios.

1.2. Objetivos

General:
Crear sistemas de recomendación en un contexto de Big Data que permitan sugerir productos afines a los intereses de los usuarios en un marketplace.

Específicos:

- Implementar un flujo de trabajo básico usando Amazon EMR con Pyspark y Python para la creación de los sistemas de recomendación.
- Comparar versiones adaptadas para Big Data de sistemas de recomendación que usen información histórica de las reseñas, tales como técnicas baseline (rating promedio global, rating promedio por producto o por usuario, etc.) y modelos de factores latentes, teniendo en cuenta la inclusión de sesgos para estos últimos.

- Evaluar otro tipo de modelos para resolver la problemática de dar recomendaciones en presencia de usuarios o productos nuevos (inicio en frío o cold start).

2. Estado del arte

El problema de generar recomendaciones a usuarios sobre productos, música, lugares, noticias, inversiones, amistades, etc. ha ganado popularidad durante los últimos años debido al auge de las plataformas digitales. En la literatura se pueden identificar tres enfoques principales mediante los cuales se ha abordado esta problemática: el *filtrado basado en contenido*, el *filtrado colaborativo* y los *algoritmos híbridos*, siendo este último una combinación de los dos enfoques anteriores (Lü L, 2012).
El *filtrado basado en contenido* consiste en utilizar características propias de los ítems y/o usuarios para calcular las similitudes entre ellos y usarlas para generar las recomendaciones. Trabajos como el desarrollado por Jieun & Bum Kim (2017) muestran como este tipo de algoritmos pueden ser potencializados mediante el uso de redes de atributos múltiples y técnicas de centralización y agrupamiento que permitan determinar los patrones estructurales de las interacciones entre los elementos.
Por otro lado, el enfoque de *filtrado colaborativo* analiza datos históricos sobre la actividad del usuario y la emplea para predecir lo que podría gustarle, teniendo en cuenta lo que le ha gustado a usuarios cuya actividad haya sido similar. Smirnov A V. (2014) plantea un claro ejemplo del uso de este algoritmo para realizar recomendaciones de sitios turísticos. Entre las técnicas de filtrado colaborativo pueden distinguirse dos familias: los algoritmos *basados en memoria* y los *basados en modelos*. Los algoritmos basados en memoria generan recomendaciones utilizando directamente la retroalimentación del usuario sobre el elemento como lo explica a detalle Ning & Desrosiers (2015), mientras que los algoritmos basados en modelos aprenden un modelo predictivo a partir de los comentarios o calificaciones de usuario y el elemento objetivo. Entre los sistemas de recomendación de este tipo, los más exitosos son aquellos que emplean técnicas de factorización de matrices, entre los cuales es común encontrar el modelo de factores latentes. Estos métodos permiten descomponer la matriz inicial (usuario/item) en una matriz de factores del usuario y otra de factores de los ítems, siendo éstas de menor dimensión, y por consiguiente se termina reduciendo la dimensionalidad del problema a abordar, además de que permite mejorar la escalabilidad del modelo debido a que este método es fácilmente paralelizable.
Ahora bien, debido a que existe la tendencia de que algunos usuarios califiquen mejor que otros y que algunos productos son mejor calificados que otros, los datos suelen evidenciar sesgos (a nivel de usuario e ítem), por esta razón, se hace necesario tener en cuenta estos sesgos para incorporarlos dentro de los modelos de factores latentes. Una de las alternativas es incluir las calificaciones promedio como sesgo al modelo original tal y como lo presenta Koren (2010).
El trabajo de Aung & Jiamthapthaksin (2015) es una de las ultimas mejoras que se ha realizado a los algoritmos de factorización, este plantea el método *Alternating Least Squares (ALS) with Incremental Learning Bias*, en este algoritmo los sesgos son tratados como dimensiones adicionales y se busca que éste se actualice gradualmente mediante el

uso de la calificación predicha y la calificación real emitida por el usuario.
Dentro de la literatura sobre sistemas de recomendación, se resaltan problemáticas como la escalabilidad, el procesamiento de datos en streaming y el problema de cold start. Este último consiste en generar recomendaciones para nuevos usuarios o nuevos ítem. Entre las soluciones propuestas para éste se encuentran la implementación de familias de algoritmos de LSH (Localitive Sesitive Hashing) que permitan buscar de manera rápida los vecinos del nuevo item o usuario y así generar recomendaciones. Sin embargo se menciona que, tal y como lo plantean Chi *et al.* (2020), estas metodologías por si solas pueden incurrir en Falsos Positivos (FP) y Falsos Negativos (FN) ya que son algoritmos probabilísticos.

3. Metodología de investigación.

Durante este proyecto, se sigue la metodología de investigación CRISP-DM. Se aclara que en el proyecto se abordan dos problemáticas, la primera corresponde a la creación de sistemas de recomendación basados en información histórica, y la segunda a la implementación de un sistema de recomendación para los productos nuevos que ingresan al marketplace. Teniendo esto claro, dentro de la aplicación de la metodología primero se realiza el entendimiento del problema y la revisión de los datos con PySpark usando Amazon EMR. Luego, se preparan los datos de los productos y las reseñas para crear distintas versiones de modelos de recomendación (esta preparación corresponde a una etapa de filtrado, en la cual se tendrán subconjuntos de datos para realizar el proceso de modelamiento, además de la eliminación de registros nulos) y finalmente se realiza el proceso de modelamiento y evaluación de estos modelos y se considera el escalamiento a un conjunto mayor de información.
Dentro de las primeras versiones de modelos, para la primera problemática abordada se tiene inicialmente la aplicación de técnicas baseline (acá se consideran 4 modelos baseline diferentes, (1) asignar el rating como el promedio global de los ratings, (2) asignar el rating como la mediana global de los ratings, (3) asignar el promedio del rating por producto y (4) asignar el promedio del rating por usuario. Estás técnicas no sólo definen un punto de partida para comparar resultados, sino que podrían ser un paso necesario en el manejo de nuevos usuarios. Luego se utilizan técnicas clásicas de la literatura relacionadas a modelos de factores latentes y su versión con inclusión de sesgos (acá se utiliza la implementación del modelo ALS de PySpark.ml, además de la adición de los sesgos al mismo).
Con el fin de mantener la participación de cada producto en los grupos de entrenamiento, validación y testeo, se realiza un muestreo estratificado por producto empleando la proporción de cada ítem dentro de la muestra total. La muestra fue dividida en un 60 % - 20 % - 20 % para datos de entrenamiento, validación y testeo, respectivamente.
En la etapa de modelamiento al momento de utilizar los modelos de factores latentes (ALS en Pyspark.ml), se evalúan diferentes combinaciones que permitan encontrar bajo la métrica $RMSE$ la mejor elección de hiperparámetros que minimicen dicho objetivo. Así, las variaciones de los hiperparámetros a considerar son:

- *Rank*: Hace referencia a la cantidad de factores latentes que utilizará el modelo. En este trabajo se evalúan 5, 10 y 15 factores latentes.

- $regParam$: Parámetro de regularización λ. Se considera 0.01, 0.1 y 0.5.
- $maxIter$: Número máximo de iteraciones. En este caso, empleamos las iteraciones por defecto que equivalen a 10.

Al momento de considerar la incorporación de sesgos, se realizan las mismas combinaciones de hiperparámetros mencionadas anteriormente, con la diferencia que la variable respuesta del sistema de recomendación está normalizada y libre de sesgos, y al realizar la predicción, se realiza la incorporación de los respectivos sesgos asociados a los usuarios y productos (al igual que el sesgo global). Al realizar esta modificación, el modelo permite detectar, por ejemplo, que hay usuarios que suelen dar ratings mucho más altos que el usuario promedio, y tener esto en cuenta dentro de la predicción del rating.
Inicialmente, éstos modelos fueron ejecutados con solo una muestra de la base de datos, de forma que fuera posible realizar diferentes pruebas y ajustes de los modelos en un tiempo razonable. Posteriormente, el mejor modelo seleccionado con esta muestra es escalado a un conjunto de datos mucho más grande (la cantidad de datos considerados dentro del escalamiento depende de la capacidad del cluster utilizado).
Por otro lado, al realizar la exploración de los modelos para la problemática del inicio en frío ("cold start"), no se contaba con ningún tipo de información propia de los usuarios, por lo que un modelo para un usuario nuevo se limitaría probablemente a solamente recomendar los productos más populares del marketplace, siendo así un modelo no tan interesante para analizar. Sin embargo, en este estudio sí se cuenta con información propia de los productos, por lo que para abordar el problema del cold start se utilizan modelos basados en content-based filtering tal como lo presenta Bianchi *et al.* (2017), los cuales usan características propias de los productos diferentes al rating, como por ejemplo, su precio o su categoría.
En este estudio se diseñó una versión propia de un procedimiento que se sugiere seguir para los casos donde se deba resolver la problemática del cold start. El proceso sugerido consta de dos fases, las cuáles se deben seguir para generar recomendaciones.
La primera fase consiste en usar la metodología de *Locality Sensitive Hashing* vista en clase para hallar productos similares para el producto de interés. En el caso de enfrentar la problemática de cold start, el producto de interés sería el producto nuevo que ingresa a la base de datos. La variable que usaríamos como entrada para esta parte sería el título del producto nuevo. La primera parte de esta metodología consiste en llevar todos los títulos de los productos a shingles, aquí usamos 1-gramas (palabras) para realizar esta conversión. Posteriormente, se sigue con las partes de *MinHashing* y luego de *Locality Sensitive Hashing*, las cuales son ejecutadas a través del modelo de *MinHashLSH* de la librería pyspark.ml, usando un total de 5 tablas de hash (5 bandas) como uno de los hiperparámetros del proceso. Con este *MinHashLSH*, lo que se obtendrá serán los 15 productos más similares al producto nuevo según el campo de título (este número de productos similares también es un hiperparámetro del proceso).
La entrada para la segunda fase del proceso corresponde a la salida de la primera fase. Aquí lo que se hace es calcular un score (el cual llamamos 'score de afinidad') entre el producto de interés (el producto nuevo) y cada uno de los 15 artículos más similares según

la fase 1 (en total se tienen 15 scores). El score de afinidad que calculamos tiene en cuenta variables propias de los productos para ver qué tan afines son (precio, marca y categorías relacionadas). Para calcular el score, se hace lo siguiente para cada variable:

- Se usa una transformación logarítmica en base 10 para llevar los precios a una escala donde se penalicen más las diferencias en precios en las magnitudes pequeñas (un producto que valga 1000 dólares y otro que valga 1010 dólares son muy parecidos en precio, mientras que un producto de 1 dólar y otro de 11 dólares son muy distintos en precio). Luego, se calculan las distancias entre el producto de interés y cada uno de los 15 productos y se normalizan usando Min-Max. Por último, se aplica la operación de 1 menos las distancias normalizadas. Así, este valor serviría como una medida de similitud entre los precios de los productos (un valor de 1 indica que los precios son idénticos).

- Para la variable de marca, este factor simplemente se resume en asignar un valor de 1 si ambos productos son de la misma marca y un 0 en otro caso.

- La variable de categorías relacionadas incluye una lista de varias categorías donde pueden catalogarse los productos (por ejemplo, un producto puede estar en la categoría de ropa y en la de deportes). Lo que se hace aquí es calcular la similaridad de Jaccard entre las categorías relacionadas de ambos productos.

Teniendo ya el valor del factor de cada variable en el score, lo que se hace es sumarlos para hallar el score de afinidad entre el producto de interés y cada uno de los 15 artículos más similares (el score máximo sería de 3). Así, lo que se hace ahora es simplemente elegir los 5 productos con mayor score de afinidad, los cuales serían la salida de este proceso.
De esta forma, se puede, por ejemplo, identificar a las personas que hayan comprado o reseñado alguno de los 5 productos más afines al producto nuevo, para así recomendarles que adquieran este nuevo producto.

4. Análisis de datos

4.1. Descripción de los datos

El dataset seleccionado es Amazon Customer Reviews, el cual contiene más de 150 millones de reseñas registradas entre 1995 y 2015 por diferentes usuarios frente a sus compras en Amazon y se encuentran clasificadas en 46 categorías[1]. El conjunto de datos se encuentra almacenado en un bucket público de *S3* de Amazon y la información a utilizar está disponible en formatos TSV (la totalidad de los archivos comprimidos ocupa cerca de 32 GB).
En cuanto a la información relacionada a los datos de las reseñas de Amazon, la tabla 1 presenta una descripción de los campos que ésta contiene.

[1] `https://s3.amazonaws.com/amazon-reviews-pds/readme.html`

Tabla 1: Descripción del conjunto de datos

Variable	Tipo de dato	Descripción
marketplace	string	Código del país del usuario que escribe la reseña
customer_id	string	Identificador de usuario que escribe la reseña
review_id	string	Identificador único de la reseña
product_id	string	Identificador único del producto que recibe la reseña
product_parent	string	Otro identificador del producto
product_title	string	Nombre del producto
product_category	string	Amplia categoría de productos
star_rating	int	Calificación del producto de 1 a 5
helpful_votes	int	Número de votos de Reseña útil"que recibe la reseña
total_votes	int	Número total de votos
vine	string	Reseña escrita por un usuario del Vine program
verified_purchase	string	Compra realizada en Amazon
review_headline	string	Título de reseña
review_body	string	Reseña
review_date	bigint	Fecha de la reseña
year	int	Año de la reseña

El total de las reseñas corresponden a 21.3 millones de artículos únicos adquiridos por 33.5 millones de clientes. Al analizar la información de las reseñas disponibles en las diferentes categorías de productos, se evidencia un desbalance importante: prendas de vestir, zapatos, libros (físicos y digitales), artículos para el hogar y productos deportivos concentran casi la mitad de los artículos reseñados.

Además, la mínima calificación promedio por categoría es de 3.5 y la máxima calificación promedio por categoría es de 4.7, mientras que el promedio de los promedios de los ratings por categoría corresponde a 4.2, por lo que, en general, los productos tienen reseñas positivas. También se menciona que las categorías con mayor cantidad de reseñas por producto son las de videojuegos, aplicaciones móviles y software.

Adicional a la información descrita, en un repositorio público[2] se encuentra información relacionada a los reviews de Amazon entre 1996 y 2018. En particular, hay un archivo que contiene metadatos de los productos de Amazon y contiene el nombre del producto, código (id del producto), precio, entre otras variables, las cuales permiten enriquecer la información ya mencionada. Esta nueva información es utilizada para abordar el problema de cold start para nuevos productos. El archivo comprimido de metadata tiene un tamaño de 11 GB y contiene información de alrededor de 15 millones de productos.

Para adaptarnos a los recursos computacionales del clúster empleado, para la creación de los modelos de factores latentes (con y sin sesgo) no es adecuado utilizar la totalidad de la base de datos, por lo que se realiza un filtro de la información para obtener una muestra que permita realizar un mejor proceso de modelamiento. Para esto se seleccionaron solo las reseñas de productos con más de 150 calificaciones (aunque es un filtro grande, esto nos garantiza que, al momento de realizar la partición de los conjuntos de datos en entrenamiento, validación y prueba, tengamos en los 3 conjuntos reviews de todos los productos filtrados debido a que se realiza la partición estratificada por código de producto) y de clientes que hayan efectuado al menos 10 reseñas (esto para intentar procurar que

[2] `http://jmcauley.ucsd.edu/data/amazon/`

se pueda modelar bien el comportamiento y los sesgos de cada usuario, además de que ayuda a reducir mucho la dimensionalidad de la matriz de usuarios), dejando un total de 14.9 millones de reseñas para evaluar los modelos con información histórica. Esta muestra contiene 115.434 productos diferentes y 765.213 usuarios únicos.
Por otro lado, en cuanto al modelo de cold start con la metadata de los productos, luego de filtrar para quitar los registros con información nula o errónea y de quitar los productos de la metadata que no tengan ningún rating para que pueda hacerse una fase de validación, se conservan 3.7 millones de productos que se usan en el proceso propuesto para este caso.

4.2. Modelos con información histórica

Los modelos con información histórica inician con los baseline, para así tener puntos de comparación para los modelos posteriores. Éstos consisten en extraer el promedio o la mediana de las calificaciones (global, por usuario o por producto) para predecir la reseña que van a dar los clientes a los productos que aún no compran. En este caso, se evalúan cuatro modelos baseline empleando la mediana, la media global, la media por producto y la media por usuario. Los resultados presentados en la tabla 2 muestran que el modelo baseline con mejor comportamiento es el de media por producto con un RMSE de 1.058.

Tabla 2: Desempeño de modelos Baseline evaluados en conjunto de validación

	Mediana	**Media**	**Media Producto**	**Media Usuario**
RMSE (Validación)	1.309	1.115	**1.058**	1.068
Tiempo ejecución (segs)	172.58	28.57	**29.98**	32.09

Continuando con la aplicación de modelos con información histórica, se emplean modelos de factores latentes utilizando el método ALS de PySpark.ml. Los modelos fueron entrenados con el conjunto de training previamente mencionado y validados con el conjunto de validación. Los inputs recibidos por el modelo son el *user id* que corresponde a la identificación del usuario que calificó el producto, el *star rating* que refleja la calificación dada por el usuario y el *item* que corresponde al producto calificado por el usuario. Como se mencionó previamente, se realizó la aplicación de diferentes combinaciones de hiperparámetros generando un total de 9 modelos. En la tabla 3 se encuentran los resultados obtenidos en el conjunto de validación para cada uno de los modelos correspondientes a las combinaciones de hiperparámetros.

Tabla 3: Desempeño de modelos de factores latentes en el conjunto de validación

Iteraciones	**Factores latentes**	**Regularización**	**RMSE**	**Tiempo de ejecución(s)**
10	10	0.5	1.121373	131.4478
10	15	0.5	1.121463	142.8668
10	5	0.05	1.12544	128.7130
10	5	0.1	1.145486	120.6946
10	15	0.1	1.159793	140.3277
10	10	0.1	1.166197	129.0120
10	5	0.01	1.284851	120.7189
10	10	0.01	1.660710	128.8289
10	15	0.01	2.146603	137.0819

Como se observa en la tabla 3, el mejor modelo basado en la métrica RMSE es el que tiene como hiperparámetros 10 iteraciones, 10 factores latentes y 0.5 como parámetro de regularización, permitiéndonos obtener el menor RMSE con un valor 1.121373; de acuerdo a estos resultados, se elige como modelo para ser comparado con el resto de modelos incluidos dentro de la sección, buscando elegir el modelo con mejor desempeño que permita realizar las recomendaciones finales.
Finalmente, se incluyen sesgos a la estructura de modelo de factores latentes. Se consideraron sesgos globales (media global), a nivel de usuario (media por usuario menos media global) y a nivel de producto (media por producto menos media global). Estos modelos de factores latentes con sesgos fueron evaluados para las mismas combinaciones de hiperparámetros mencionadas anteriormente, con la diferencia que ahora la variable respuesta corresponde al rating normalizado (rating sin sesgo) para que, una vez obtenida la predicción, los respectivos sesgos sean incorporados. En la tabla 4 se pueden encontrar los resultados de los diferentes modelos que incluyen sesgos en el conjunto de validación.

Tabla 4: Desempeño de modelos de factores latentes con sesgos en el conjunto de validación

Iteraciones	**Factores latentes**	**Regularización**	**RMSE**	**Tiempo de ejecución(seg)**
10	15	0.5	1.030120	125.8144
10	10	0.5	1.030124	119.2589
10	5	0.5	1.030137	113.5616
10	15	0.1	1.101444	127.2024
10	10	0.1	1.123069	118.5687
10	5	0.1	1.136435	110.0443
10	15	0.01	1.352336	125.5230
10	10	0.01	1.464103	120.3595
10	5	0.01	1.493786	109.8102

Teniendo en cuenta los resultados de los diferentes modelos explicados previamente, se observa que el mejor modelo en general, bajo la métrica RMSE, es el modelo de factores latentes con sesgos teniendo como resultado un RMSE de 1.030 y un tiempo de ejecución de 125.81 segundos y considerando como hiperparámetros 10 iteraciones, 15 factores latentes y 0.5 como parámetro de regularización.
Finalmente, como conocemos el desempeño en el conjunto de validación del mejor modelo baseline, del mejor modelo de factores latentes y del mejor modelo de factores latentes con sesgo, es posible concluir que la mejor arquitectura de sistema de recomendación corresponde a la obtenida por el modelo de factores latentes con sesgo. Por lo tanto, pasando al conjunto de prueba, con el modelo ganador se obtiene un RMSE en test de 1.01988, indicando que estos resultados son consistentes e incluso mejores con respecto a los obtenidos en validación. La arquitectura ganadora será evaluada al momento de escalar el modelo en la sección de arquitectura de big data.

4.3. Modelos cold start

4.3.1. Validación del modelo

Para validar el rendimiento del proceso propuesto para abordar el problema de cold start, se considera la exclusión del conjunto de datos de un total de 44 artículos elegidos de forma

aleatoria. Éstos harían el papel de productos nuevos, por lo que a éstos se les hallará los productos más afines a cada uno para realizar la recomendación.

Al ejecutar el proceso definido para encontrar los artículos más afines a cada uno de estos productos, notamos que el tiempo total que tarda por cada uno de los 44 productos es de aproximadamente 40 segundos. Esto se justifica debido a que el modelo de MinHashLSH definido en Pyspark está diseñado para que busque los datos más similares a solamente un producto a la vez, por lo que es necesario ejecutar la búsqueda de vecinos más similares para cada uno de los 44 productos. Esta es la razón por la que se usa un conjunto de validación pequeño, de solo 44 datos. Sin embargo, se menciona que, si el modelo estuviera en producción, este proceso se ejecutaría solamente una vez por cada producto nuevo que un usuario quiera ingresar al marketplace, lo que conllevaría a que el usuario tenga que esperar solo aproximadamente 40 segundos para que su producto empiece a ser recomendado para otros usuarios (o, si muchos usuarios suben productos al mismo tiempo, puede que incremente, pero no debería ser mayor a unos minutos), un tiempo que se considera bastante prudente en este caso de uso.

Ahora, para evaluar con una métrica el rendimiento del modelo, se calcula el rating promedio entre los 5 productos más afines a cada producto del conjunto de validación y se considera que éste sería el rating predicho para este producto. Se procede a calcular entonces el RMSE entre el rating real y el rating predicho para cada producto y se obtiene un RMSE de 0.714 (notar que en este caso la métrica se calcula sobre los ratings promedios de cada producto en vez de sobre cada una de las reseñas). Si se hubiera usado un baseline del rating promedio de cada productos, se hubiera obtenido un RMSE de 0.719, por lo que es claro que no hay una diferencia significativa entre el RMSE del proceso propuesto y del baseline. Sin embargo, se considera que esto no representa un problema por dos razones: primero, porque el proceso propuesto usa técnicas que se basan simplemente en buscar elementos similares al producto nuevo (como LSH), pero nunca intenta optimizar un error como sí lo hacen otro tipo de modelos (como los de factores latentes); y segundo, porque este proceso se usaría solamente en presencia del problema de cold start, siendo ésta precisamente la ventaja que tiene y la razón por la que se usaría en los casos donde no deban usarse los demás modelos.

4.3.2. Ejemplo de recomendación con el modelo para cold start

Suponga que se tiene un producto nuevo que ingresará al marketplace y del cual se tiene la siguiente información: título (*Girls pink tutu dress*), marca (*Hello kitty*), precio (19,99$) y categorías relacionadas (*Dance, Sport & Outdoors, Clothing, Shoes & Jewelry*). Mediante el proceso propuesto (planteado en la sección 3), encontramos los cinco productos más afines a este producto, los cuales se muestran en la figura 1. Nótese que los productos seleccionados guardan similitud en características como el nombre, el precio, las categorías y en algunos casos en la marca con el producto nuevo que ingresaría al marketplace. De esta forma, a las personas que hayan comprado alguno de estos 5 productos se les podría recomendar que compren también el producto nuevo que ingresa.

asin	price	title	brand	category
B00QGMY3EU	$18.60 - $34.99	Clementine Little Girls' Girls Ta...	Clementine Apparel	['Clothing, Shoes & Jewelry', 'Gi...]
B006PFJMGW	$29.99	Rare Editions Little Girls' Tutu ...	Rare Editions	['Clothing, Shoes & Jewelry', 'Gi...]
B00SH78APA	$10.13 - $22.99	Hello Kitty Baby Girls' Tutu Dress	Hello Kitty	['Clothing, Shoes & Jewelry', 'No...]
B0053WLHEI	$19.95	Rare Editions Little Girls' Tutu ...	Rare Editions	['Clothing, Shoes & Jewelry', 'Gi...]
B00P835VGC	$25.99 - $28.00	Bloch Girls' Toddler Tiffany Dres...	Bloch	['Clothing, Shoes & Jewelry', 'Gi...]

Figura 1: Ejemplo de salida del proceso propuesto de Cold Start para un producto nuevo

5. Uso de herramienta de Big Data

Para el desarrollo del trabajo, se consumieron los servicios de la nube de Amazon utilizando una cuenta de AWS Educate. Dentro de los servicios utilizados, se menciona el uso de buckets de S3 para realizar el almacenamiento tanto de información de entrada para los modelos como archivos de salida, además del uso de un clúster de EMR para realizar los procesos de análisis de datos y modelamiento utilizando Pyspark.
El clúster de EMR utilizado consiste de 4 nodos (1 master node y 3 core nodes) con carácterísticas m5.xlarge, además de tener la integración con Spark 2.4.5. Este clúster fue asociado a un Notebook de Jupyter, en el cual se utiliza el kernel de PySpark para realizar el procesamiento de la información.
Por otro lado, en S3 guardamos la información de la metadata de los productos en formato CSV de forma particionada, además de información muestreada de los reviews en formato parquet para luego ser consumidos de forma más rápida.

Tabla 5: Resultados escalamiento: relación entre tiempo de cómputo y dimensiones del conjunto de datos al entrenar el sistema de recomendación seleccionado

Cantidad de reviews	**Número productos**	**Número usuarios**	**Tiempo ejecución (mins)**	**RMSE TEST**
14919839	115434	765213	4.4	1.01988102
24241726	115437	2227252	5.2	1.09051578
44865227	444874	3549734	16.6	1.08858997
49290136	570405	3805335	21.8	1.08874656
55019257	779124	4124959	32.3	1.089730156
58547703	941397	4319604	43.5	1.090725719
62749771	1178599	4548310	-	-

Finalmente, se pudo evidenciar que los procesos creados funcionan satisfactoriamente en diferentes cantidades de reviews tal como se muestra en la tabla 5. En esta tabla es posible ver la dimensión de diferentes muestras de conjuntos de datos creados para realizar tanto el entrenamiento como la predicción del sistema de recomendación con la mejor arquitectura encontrada en este trabajo. Se menciona que es posible escalar el sistema de recomendación a una cantidad de datos mayor ya que en Amazon EMR se tiene la posibilidad tanto de realizar escalamiento vertical (aumentar la capacidad de cada una de las máquinas del

clúster) como de escalamiento horizontal (agregar una mayor cantidad de nodos para realizar el procesamiento). Pues, con las limitaciones de la cuenta estudiantil de AWS, se pudo escalar el modelo hasta 58.5 millones de registros (el cual tardó 43.5 minutos en entrenar y realizar las predicciones en el conjunto de prueba), ya que al intentar una muestra levemente mas grande (con 62.7 millones de registros), los límites de la cuenta y del clúster seleccionado no permitieron realizar la creación del modelo. Teniendo menos restricciones en los recursos computacionales disponibles, se podría abarcar la totalidad del conjunto de datos para la creación del sistema de recomendación.

6. Entregables

Como entregables del proyecto, se incluyen los códigos y notebooks realizados en el proceso de modelamiento y construcción del sistema de recomendación. De esta manera, en el repositorio de github (`https://github.com/psaldar/trabajo-mineria-datos`) se puede encontrar el Notebook de Jupyter utilizado en Amazon EMR en el cual se realizó la implementación y la comparación de los modelos y el código utilizado para procesar y particionar la metadata, junto con un README que describe el contenido del repositorio.

7. Conclusiones

- La reducción de la dimensionalidad que realiza el modelo de factores latentes facilita el procesamiento y la paralelización de grandes volúmenes de información en la construcción de un sistema de recomendación.
- Para la aplicación de modelos de factores latentes, generar y comparar diferentes combinaciones entre los hiperparámetros del algoritmo ayuda notablemente a minimizar el error, permitiendo así seleccionar los mejores hiperparámetros para el modelo final.
- El proceso de dos fases que combina un MinHashLSH y el posterior cálculo de scores de afinidad es una buena alternativa para cuando se tenga que dar recomendaciones en presencia de productos nuevos (cold start).
- En el contexto de Big Data, es necesario tener en cuenta las limitaciones de los recursos computacionales dado que se cuenta con una gran cantidad de información.

8. Trabajo Futuro

- Ejecutar el modelo en un clúster con mayor capacidad de cómputo para lograr escalarlo a la totalidad de los datos.
- Realizar un ajuste o tuneo de los hiperparámetros del algoritmo propuesto para la problemática de cold start en busca de mejorar su rendimiento.

- Evaluar mejoras en los modelos tales como la inclusión del factor temporal de las reseñas, el uso de técnicas NLP en los comentarios de cada reseña y la extracción de características importantes de las imágenes de los productos.

9. Implicaciones éticas.

La naturaleza de los algoritmos de recomendación consiste en dar forma a las preferencias de los usuarios y guiar sus elecciones individuales y sociales, lo que conlleva en si mismo varios cuestionamientos éticos interesantes. Para este proyecto, se identificaron las siguientes implicaciones éticas (Milano, 2020):

- El riesgo latente de que la información del usuario sea filtrada o empleada indebidamente para violar su privacidad.
- Los impactos negativos que puedan generarse en las partes involucradas (productores-usuarios) que implique la quiebra de unos, junto con la falta de transparencia y explicabilidad en las recomendaciones hechas a los usuarios.
- Exponer al usuario a recomendaciones con contenido dañino u ofensivo según sus preferencias culturales, ideológicas y morales.

10. Aspectos legales y comerciales.

Este proyecto brinda a los E-commerce, almacenes de cadena y grandes superficies una herramienta para ayudar a sus clientes a tomar, de manera informada, una decisión ágil de compra, permitiendo optimizar tiempo y dinero tanto a las empresas como a los consumidores. Adicionalmente, los proveedores y productores tendrán un impacto directo en sus ingresos al considerarse que un producto altamente recomendado puede tener mayor oportunidad de compra. Sin embargo, el diseño de sistemas de recomendación trae grandes desafíos desde el punto de vista jurídico y legal debido a que sus resultados dependen de la información suministrada por otros usuarios, lo que podría llevar erróneamente a difamaciones o tergiversaciones que representen un daño reputacional para una marca o una exposición indebida de los clientes (Burgess, 2011).

11. Ejecución del plan

En la planeación realizada inicialmente, resultó beneficiosa la antelación con la que se inició la búsqueda del problema y la selección de las bases de datos, así como la división de tareas con el fin de segregar funciones para luego unir los resultados bajo un esquema de trabajo en equipo, lo anterior nos permitió ganar tiempo para llevar a cabo el plan.
Sin embargo, el desarrollo del proyecto contó con la aplicación de diferentes técnicas en una muestra de los datos para posteriormente escalarlo. Allí tuvimos algunos inconvenientes, ya que de acuerdo a las capacidades de las máquinas no logramos finalmente manejar de

forma eficiente las cuentas de AWS, debido a que teníamos altos tiempos de ejecución al escalar los modelos a una cantidad de datos mayor, por lo cual se vio impactada la fase de escalamiento de modelos, generando retrasos en la última parte del cronograma.
Lo anterior nos lleva a entender que es muy importante definir la cantidad de datos de la muestra que se usa para comparar los modelos y tener claro el volumen de datos a los que se deben escalar, contemplando siempre los recursos computacionales disponibles. En un inicio pretendimos aplicar las técnicas a la totalidad del conjunto de datos, lo que nos hizo perder tiempo y replantear la base de datos inicial aplicando por ejemplo algunos filtros que ayudaran a disminuir los tiempos de ejecución. Adicionalmente, es muy importante tener claras algunas funcionalidades de los lenguajes utilizados, pues la correcta programación del script nos ayudó a mejorar los tiempos de ejecución.

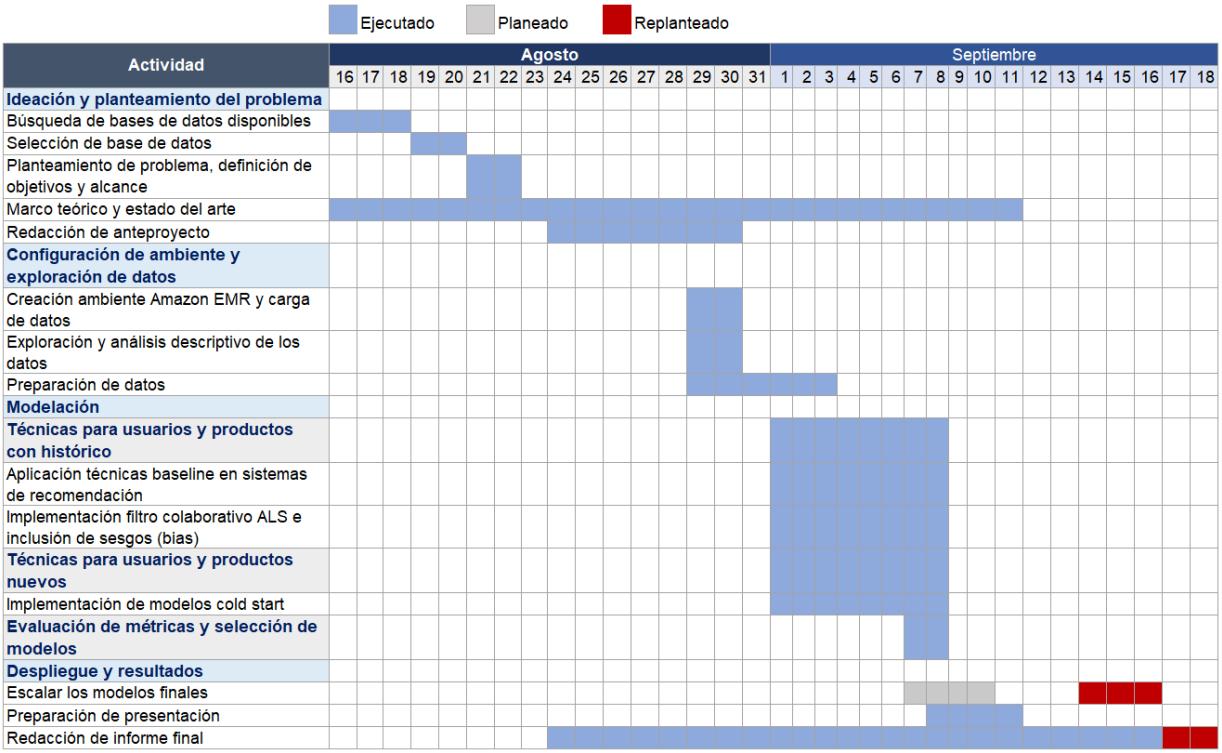

Figura 2: Diagrama Gantt para el desarollo del proyecto

Referencias

Aung, T. H., & Jiamthapthaksin, R. 2015. Alternating Least Squares with Incremental Learning Bias. *Pages 297–302 of: 2015 12th International Joint Conference on Computer Science and Software Engineering (JCSSE).*

Bianchi, Mattia, Cesaro, Federico, Ciceri, Filippo, Dagrada, Mattia, Gasparin, Alberto, Grattarola, Daniele, Inajjar, Ilyas, Metelli, Alberto Maria, & Cella, Leonardo. 2017. Content-based

approaches for cold-start job recommendations. *Pages 1–5 of: Proceedings of the Recommender Systems Challenge 2017*.

Burgess, Stephen, Sellitto Carmine Cox Carmen. 2011. Trust perceptions of online travel information by different content creators: Some social and legal implications. *Page 221–235 of: Information Systems Frontiers*. Springer.

Chi, Xiaoxiao, Yan, Chao, & Wang, Hao. 2020. Amplified locality-sensitive hashing-based recommender systems with privacy protection. *In: Concurrency and Computation Practice and Experience*.

Jieun, Son, & Bum Kim, Seoung. 2017. Content-based filtering for recommendation systems using multiattribute networks. *Pages 404–414 of: Expert Systems With Applications*.

Koren, Yehuda. 2010. Collaborative Filtering with Temporal Dynamics. *Pages 89–97 of: Communications of the ACM*.

Lü L, Medo M, Yeung CH Zhang YC Zhang ZK Zhou T. 2012. "Recommender systems. *Pages 1–49 of: Physics Reports*.

Milano, Silvia, Floridi Luciano. 2020. Recommender systems and their ethical challenges. *In: AI SOCIETYl*.

Ning, Xia, & Desrosiers, Christian. 2015. A Comprehensive Survey of Neighborhood-Based Recommendation Methods. *In: Recommender Systems Handbook*. Springe.

Smirnov A V., Shilov NG, Ponomarev A V. Kashevnik AM Parfenov VG. 2014. Group context-aware recommendation systems. *Pages 325–334 of: Scientific and Technical Information Processing*. Springer.