Generación de reglas de asociación para productos de *retail* utilizando el algoritmo FP-Growth paralelo

Renato Pérez-Gómez
renato08lasalle13@gmail.com / Universidad de Lima, Perú

Recepción: 19-6-2019 / Aceptación: 8-8-2019

Resumen. Las organizaciones minoristas actuales tienen varias sucursales conectadas bajo el mismo sistema de gestión distribuido. Estos sistemas almacenan y registran la información de todas las transacciones dadas en las tiendas. Paralelamente, con el rápido crecimiento e implementación de las tecnologías de la información e Internet, la cantidad de datos generados en cada transacción o venta realizada es sustancial. Las técnicas de minería de datos tienen como objetivo identificar patrones y tendencias en una gran recopilación de datos. Su uso tiene un atractivo para los minoristas, ya que quieren convertir la gran cantidad de datos que tienen en información y conocimiento útiles. Una aplicación de minería de datos que atrae a los minoristas es el descubrimiento de reglas de asociación. El descubrimiento de estas reglas es la base de muchas decisiones comerciales, como el diseño de la canasta de productos, la elección de la estrategia de promoción y la combinación de productos. La intención de la investigación es la aplicación de las técnicas y metodologías de aprendizaje de asociación de reglas para la realidad de un comercio minorista con sucursales ubicadas en Lima. El conjunto de datos utilizado en esta investigación corresponderá a las transacciones realizadas con el tiempo para productos de consumo masivo.

Palabras clave: minería de datos, análisis de la cesta de la compra, análisis de reglas de asociación, conjuntos de elementos frecuentes, FP-Growth, computación paralela

Generation of Association Rules for Retail Products Using the Parallel FP-Growth Algorithm

Abstract. Retail organizations today have several branches connected under the same distributed management system. These systems store and record the information of all transactions taken place in stores. In parallel, with the rapid growth and implementation of information technologies and the Internet, the amount of data generated in each transaction or sale is substantial. The data mining techniques are aimed at identifying patterns and trends in a large data collection. The use of these techniques is attractive for retailers, as they want to convert the large amount of data they have into useful information and knowledge. A data mining application that attracts retailers is the discovery of association rules. The discovery of these rules is the basis of many commercial decisions, such as the design of the product basket, the choice of the promotion strategy and the combination of products. This research aims to apply learning techniques and methodologies of association rules for a retailer with branches located in Lima. The data set used in this research will match the transactions made over time for mass consumption products.

Keywords: data mining, market basket analysis, association rule analysis, frequent itemsets, FP-growth, parallel computing

1. INTRODUCCIÓN

Con la tendencia, en los últimos años, de registrar y almacenar todo dato posible relacionado a los procesos operativos de la organización, la cantidad de *data* generada es abundante. Respecto a la industria del *retail*, gran parte de la *data* generada proviene de los miles de transacciones efectuadas en cada una de las sucursales. Los grandes representantes del sector a nivel mundial han identificado el gran potencial en minar la información oculta dentro de estos conjuntos de datos. Sin embargo, la aplicación de las técnicas de minería de datos no es el estándar en la mayoría de las organizaciones. La mayor aplicación de la minería de datos en el sector *retail* es la minería de reglas de asociación. Esta práctica supone un atractivo para las organizaciones debido a que tiene la intención de convertir la gran cantidad de datos que poseen en información útil y conocimiento. El descubrimiento de estas reglas es la base de muchas decisiones comerciales, como el diseño de la cesta de productos, la elección de la estrategia de promoción y la combinación de productos. Para la minería de reglas de asociación normalmente se hace uso del algoritmo Apriori. No obstante, en el desarrollo del algoritmo no estuvo pensado para manejar grandes volúmenes de datos por lo que en la actualidad es costoso, tanto computacional como en memoria. Existen otras alternativas óptimas respecto al manejo de memoria y que requieren de menos capacidad computacional; el algoritmo FP-Growth demuestra tener mejores resultados en rendimiento. Este algoritmo busca resolver las desventajas que posee el algoritmo clásico. Ya que este consiste en dos pasos para la generación de las reglas de asociación, es menos costoso en tiempo y en recursos. Si a este algoritmo se le suma la capacidad de procesar información en paralelo, el tiempo necesario para la ejecución del algoritmo será reducido drásticamente. Para lograr esto, en esta investigación, se hizo uso del *framework* Apache Spark y de su implementación del algoritmo FP-Growth.

2. ESTADO DEL ARTE

Los algoritmos existentes para la minería de reglas de asociación funcionan en datos estáticos, (Kaur y Kang, 2016). Estos encuentran las mejores reglas de asociación en base a métricas como soporte, confianza, elevación, etc. Sin embargo, para la próxima vez que se realiza la extracción de datos, estos algoritmos no capturan automáticamente los cambios en los datos. Por ello se usa algún otro algoritmo de comparación para rastrear el cambio en los datos. Este último, se utiliza para comprender la dinámica del proceso de generación de datos mediante la examinación de los cambios que se han producido en los patrones descubiertos. Se concluye que la minería periódica es un nuevo enfoque en la minería de datos que adquiere importancia. Este campo está evolucionando debido a las necesidades en diferentes aplicaciones y limitaciones de la minería de datos. Esto aumentaría el poder de las técnicas existentes de extracción de datos.

Según Di Fatta (2019), podría decirse que algunos factores limitantes han impedido una adopción más generalizada de ARM (*association rule mining)*. En primer lugar, el descubrimiento

de patrones interesantes a partir de datos requiere algoritmos combinatorios complejos y, a menudo, se benefician de la computación de alto rendimiento. Para problemas del mundo real, el espacio de búsqueda puede ser prohibitivamente grande. En segundo lugar, no es raro que incluso para un conjunto de datos de entrada relativamente pequeño, el conjunto de patrones descubiertos sea muy grande, incluso más grande que el conjunto de datos de entrada. En este caso, ARM es solo un primer paso en flujos de trabajo de datos más complejos que incluyen otras técnicas de extracción de datos. Por estas razones, un enfoque multidisciplinario que combina la experiencia tanto de la informática como del dominio de la aplicación específica es a menudo crítico.

En la investigación realizada por Griva, Bardaki, Pramatari y Papakiriakopoulos (2018) se propone un enfoque de análisis de negocios que extrae segmentos de visitas de clientes a partir de datos de venta. Se caracteriza por visita al cliente según las categorías de productos comprados en la canasta e identifica la intención de compra o la misión detrás de la visita. Además, se sugiere un enfoque de selección de funciones semisupervisadas que utiliza la taxonomía del producto como entrada y sugiere categorías personalizadas como salida. Este enfoque se utiliza para equilibrar el árbol de taxonomía del producto que tiene un efecto significativo en los resultados de la extracción de datos. En general, el resultado de esta investigación permite observar el comportamiento de compra de los clientes desde un punto de vista alternativo, no como sus necesidades e intenciones generales de compra, sino que se centran en sus requisitos y motivos por cada viaje o visita de compras.

Referente a Bhandari, Gupta y Das (2015), identifican la principal limitación de Apriori como el costo de tiempo para mantener un gran número de conjuntos candidatos con conjuntos de elementos frecuentes, soporte mínimo o conjuntos de elementos grandes. Por lo que proponen modificar el algoritmo para generar un FP-Tree en lugar de iterar el *dataset* una gran cantidad de veces. Con el algoritmo de Apriori se escaneó la base de datos dos veces, pero evidenciaron una mejora al usar el algoritmo paralelo y el concepto de partición. Presentan una fórmula matemática para seleccionar el clúster, ya que hay muchos clústeres. El resultado, un híbrido de Apriori con generación de FP-Tree. Concluyen, así, que el espacio de memoria se reduce drásticamente cuando se realiza una gran cantidad de transacciones desde los *data-warehouse* y al reducir el tiempo consumido en el escaneo de transacciones y también al reducir el número de transacciones a escanear.

En Galarreta Vásquez (2016) se identifican reglas de asociación entre categorías de productos de electrodomésticos de un *retail.* Para la prueba utilizaron *data* histórica de diez años para tres tiendas. El modelo se construyó con el *software* RapidMiner conectado a una BD en SQL Server. Eligió el algoritmo FP-Growth al ser más eficiente que Apriori e hizo uso de la totalidad de registros disponibles en la BD. Lo productos presentaban la siguiente jerarquía: familia, subfamilia, marca y es oferta. La *data* fue filtrada por oferta para dividir los productos en campañas promocionales, ya que podían alterar el resultado. La BD fue transformada para

generar una matriz, la cual, luego, fue procesada por el algoritmo. La matriz almacenó la información cruzada de todo cliente que compró en el local y todas las subfamilias de productos. Si un cliente compró, por lo menos una vez un producto de la subfamilia, entonces fue registrado como TRUE, sea contrario como FALSE. Se concluyó que los productos con bajo precio incentivan la compra de productos con mayor precio.

Respecto a Xue, Wang, Liu y Li (2010), diseñan un modelo de minería de reglas de asociación para una venta de libros *online.* Para esto tienen 623 registros en la BD, los cuales fueron separados en 4 tablas. Definieron los parámetros soporte y confianza con 2 % y 60 %, respectivamente. Para la minería crearon una librería basada en ASP.NET. Luego, con un experimento analizaron los hábitos de compra de los clientes al encontrar asociaciones entre los artículos de pedido en la base de datos. La información obtenida del resultado del ejercicio concluyó que se lleva un aumento de las ventas al ayudar a los *retailers* a realizar una comercialización selectiva y planificar el espacio de la librería.

En Chunhua y Dongjun (2008) se presentan dos modelos de minería capaces de adaptarse a dos tipos de organización. Los autores plantean que existen dos tipos de organizaciones, aunque todas adoptan el enfoque de minería distribuida y el mismo patrón de toma de decisiones. En general, la carga de comunicación y la frecuencia de comunicación son los dos factores que afectan la eficiencia de la minería de datos. Se puede analizar un sistema de gestión distribuido desde tres aspectos: 1) cantidad de la tienda de sucursales; 2) el ancho de banda de comunicación; 3) entre las características del conjunto de datos local, las características incluyen si es denso o escaso, las cantidades de datos de transacciones, etc. Se definen dos tipos de organizaciones de *retail.* El primer tipo corresponde a las organizaciones que tienen varias sucursales y el bando de ancho de comunicación es largo (gran escala), mientras que la *data* de transacción de cada sucursal es pequeña. El segundo tipo, tienen pocas sucursales y comunicación pequeña, mientras que la *data* de transacción es grande (pequeña escala).

3. ANTECEDENTES

La minería de *itemsets* frecuentes dentro del *dataset* es el punto clave de la generación de reglas de asociación. Lo que se pretende con esta técnica es extraer los *itemsets* más frecuentes y grandes dentro de una gran lista de transacciones que contienen varios elementos cada uno. A continuación, una breve descripción de dos algoritmos para la minería de reglas de asociación.

3.1 Algoritmos de minería de asociación

La primera propuesta de solución presentada se dio en Agrawal, Imienlinski y Swami (1993), en donde se presenta el algoritmo Apriori. Durante veinte años, este algoritmo se presentó como la solución clásica en el área. Sin embargo, muchas otras soluciones han sido propuestas

para obtención de las reglas de asociación para base de datos transaccionales. Las dos grandes desventajas que se pueden identificar de Apriori son el tiempo de ejecución que crece exponencialmente con la cantidad de registros y los requisitos computacionales para la generación de estas reglas. Ante estos puntos, se presenta el algoritmo FP-Growth como la alternativa ideal (Han, Pei y Yin, 2000).

3.2 FP-Growth

Es una mejora de Apriori diseñada para eliminar algunos de los cuellos de botella en este (Han *et al*., 2000). El algoritmo se diseñó teniendo en cuenta los beneficios de mapReduce, por lo que funciona bien con cualquier sistema distribuido centrado en mapReduce. FP-Growth simplifica todos los problemas presentes en Apriori mediante el uso de una estructura llamada FP-Tree. En un árbol FP, cada nodo representa un elemento y su recuento actual, y cada rama representa una asociación diferente.

El árbol FP se construye leyendo el conjunto de datos una transacción a la vez y asignando cada transacción a una ruta en el árbol de FP (Han, Kamber y Pei, 2011). Como diferentes transacciones pueden tener varios elementos en común, sus rutas pueden superponerse (Tan, Steinbach y Kumar, 2005). Cuanto más se superponen las rutas, más compresión se puede lograr utilizando la estructura del árbol de FP. Si el tamaño del árbol de FP es lo suficientemente pequeño como para caber en la memoria principal, esto permitirá extraer conjuntos de elementos frecuentes directamente de la estructura en la memoria en lugar de realizar pases repetidos sobre los datos almacenados en el disco.

Para minar el árbol FP se empieza desde cada patrón de longitud 1 (patrón sufijo inicial), se construye su base de patrón condicional (consiste en el set de rutas de prefijo en el árbol que concurre con el patrón sufijo), luego se construye su árbol FP (condicional), y se realiza la minería de forma recursiva en el árbol (Han *et al*., 2011). El crecimiento del patrón se logra mediante la concatenación del patrón sufijo con los patrones frecuentes generados a partir de un árbol FP condicional.

La mayor ventaja que se encuentra en FP-Growth es el hecho de que el algoritmo solo necesita leer el archivo dos veces, como se explica a continuación, quien lo lee una vez por cada iteración. Otra gran ventaja es que elimina la necesidad de calcular los pares que se van a contar, lo cual es muy pesado en el procesamiento, ya que utiliza el FP-Tree (Fournier-Viger *et al*., 2017). Esto lo hace O (n) que es mucho más rápido que Apriori. El algoritmo FP-Growth almacena en memoria una versión compacta de la base de datos.

4. METODOLOGÍA

4.1 Exploración de la *data*

Antes de entrar a detallar el modelo y su funcionamiento, es importante dedicar esfuerzos a entender la *data* con la que se está trabajando. El análisis exploratorio de datos se refiere al proceso crítico de realizar investigaciones iniciales sobre datos para descubrir patrones, detectar anomalías, probar hipótesis y verificar suposiciones con la ayuda de estadísticas de resumen y representaciones gráficas. Es una buena práctica comprender los datos primero y tratar de recopilar tantos conocimientos a partir de ellos.

Luego de haber realizado la limpieza necesaria al *dataset* para que este se encuentre listo y sea alimentado al modelo, se realiza un análisis exploratorio a la *data*. De esta manera, la información que se obtenga permitirá generar visión de negocio. Es decir, conocimiento sobre cuál es el comportamiento entre las entidades del caso de estudio. Para este caso en específico, sería la relación entre productos y departamentos o categorías, los diez productos con más órdenes, las horas o días de la semana con más órdenes, etcétera.

4.1.1 Distribución de productos por categoría

Esta observación permite entender cuáles son los productos que más espacio ocupan en estantes y a qué categoría pertenecen (véase la figura 1). El espacio en los estantes influye en la efectividad de ventas y *marketing*. Evidentemente, existe una diferencia en la visibilidad y atención recibida entre los estantes inferiores y los estantes superiores, por lo que se puede observar una diferencia significativa entre las ventas de los productos y marcas colocadas en dos estantes diferentes. Esta distribución es fundamental para que los *retailers* puedan establecer sus estrategias de negocio y *marketing*.

Figura 1. Distribución de productos por categoría
Elaboración propia

4.1.2 Productos más vendidos

Es importante también tener en cuenta cuáles son los productos que más ventas u órdenes registran ya que son estos los puntos clave para las estrategias de negocio que elaboren los *retailers*. En la figura 2 se pueden observar los diez productos más vendidos. Esto, junto con el análisis anterior, permite saber más si la distribución de ventas de la categoría depende de uno o un grupo selecto de productos o del conjunto total.

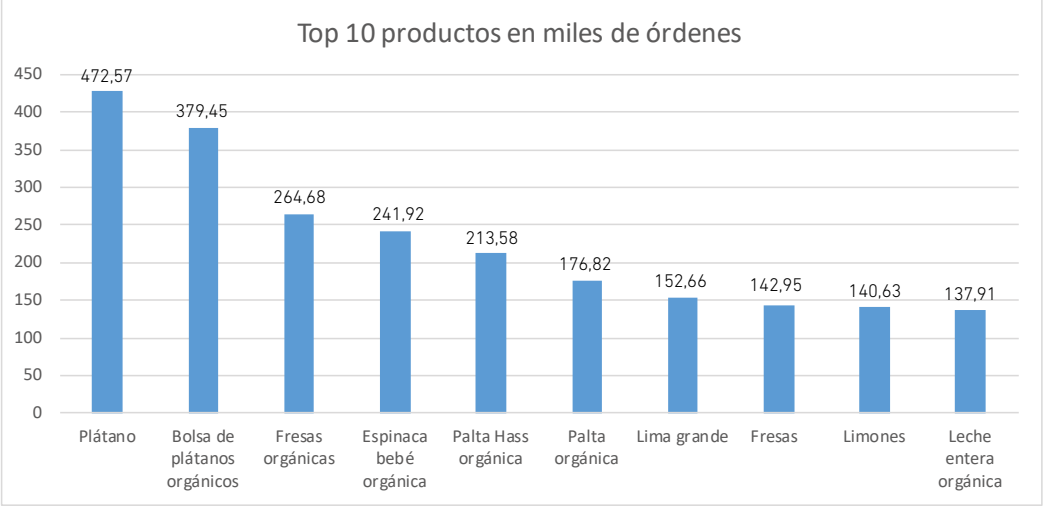

Figura 2. Top 10 productos en miles de órdenes
Elaboración propia

4.1.3 Número de productos por transacción u órdenes

De igual manera que las visualizaciones anteriores, otro aspecto importante a tomar en cuenta es la cantidad de productos que se registran por transacción u orden. Esto quiere decir, cuántos productos compra cada cliente. De esta forma, los *retailers* podrán crear ofertas personalizadas que logren atraer más a un cliente según esta métrica.

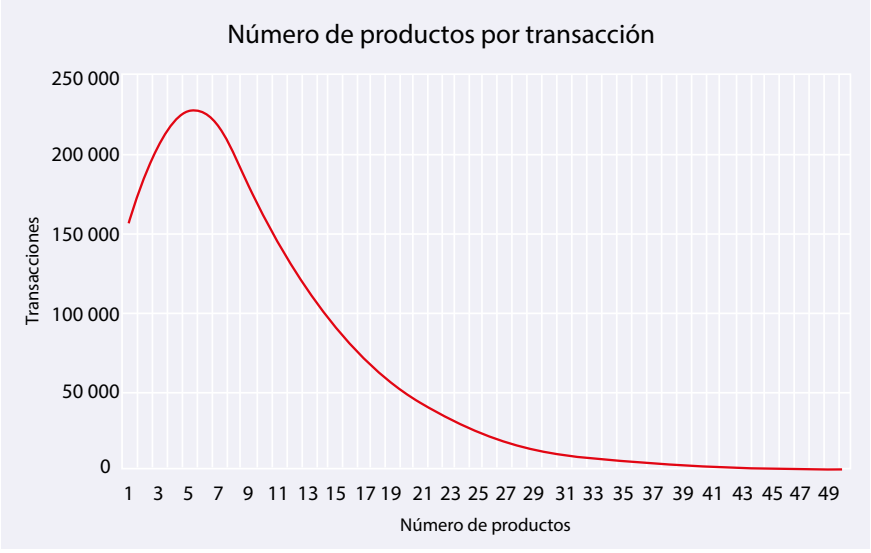

Figura 3. Número de productos por transacción
Elaboración propia

4.1.4 Cantidad de órdenes en el tiempo

Por último, esta visualización permite conocer las frecuencias de compra de los clientes. Qué días de la semana y a qué hora de dicho día se registran las mayores cantidades de transacciones. Con estos datos es posible determinar la probabilidad de que un cliente existente con un historial de compras adquiera un producto en un momento determinado. Esta información permite no solo dirigirse a los clientes que tienen más probabilidades de comprar algo, sino también adaptar la oferta a lo que es más probable que les atraiga.

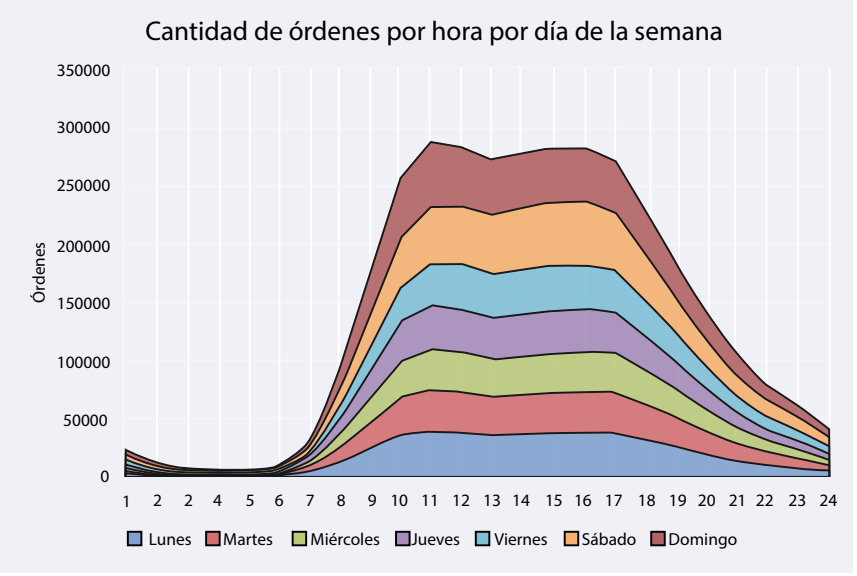

Figura 4. Cantidad de órdenes por hora por día de la semana

Elaboración propia

4.2 Modelo y experimentación

Se tomó la decisión de desarrollar la propuesta de solución en el lenguaje Python. Este es el lenguaje más utilizado en el área de Data Science (por lo que tiene una excelente comunidad y extenso conjunto de librerías), es liviano y eficiente en la ejecución de código, pero también es multifuncional. De igual forma, se utilizó el *framework* de computación en clúster y de código abierto Apache Spark, este provee una interfaz para la programación con paralelización de la *data* implícita y tolerancia a fallos.

La legibilidad y la simplicidad inherentes de Python hacen que sea relativamente fácil de recopilar y la cantidad de bibliotecas analíticas dedicadas disponibles en la actualidad significa que se puedan encontrar paquetes ya adaptados a las necesidades del problema que se pueden descargar libremente. Esto se debe a que Python es un lenguaje de programación

multiparadigma: una especie de navaja suiza para el mundo de la codificación. Es compatible con la programación orientada a objetos, la programación estructurada y los patrones de programación funcional, entre otros.

Apache Spark es un potente motor de código abierto que ofrece procesamiento de flujo en tiempo real, procesamiento interactivo, procesamiento de gráficos, procesamiento en memoria y procesamiento por lotes con una velocidad rápida, facilidad de uso e interfaz estándar. Dentro de los componentes existentes en el ecosistema de Spark, en esta propuesta de solución se utilizarán los módulos MLlib y SQL. Esta interfaz contiene implementaciones de algoritmos y modelos especializados de *machine learning*. En este módulo se encuentra la implementación del algoritmo FP-Growth.

Respecto al modelo planteado en este trabajo para la generación de reglas de asociación utilizando el algoritmo FP-Growth, lo primero que se realizó fue identificar los módulos necesarios para la solución. Estos son, un módulo para el preprocesamiento y exploración de la *data* y otro módulo para el algoritmo de generación de ítems frecuentes y generación de reglas de asociación. Lo que en el área de *machine learning* se conoce como *pipelines*. Esto se refiere a la estructura secuencial de pasos a seguir para la implementación del modelo, siendo la salida de cada paso la entada del siguiente.

El flujo de trabajo representativo de la propuesta de solución se puede observar en la figura 5. A partir de lo explicado en esta sección, se pueden abstraer cinco pasos críticos que contienen los procedimientos necesarios a llevar a cabo desde la lectura del *dataset* (*input*) hasta la generación de las reglas de asociación (*output*).

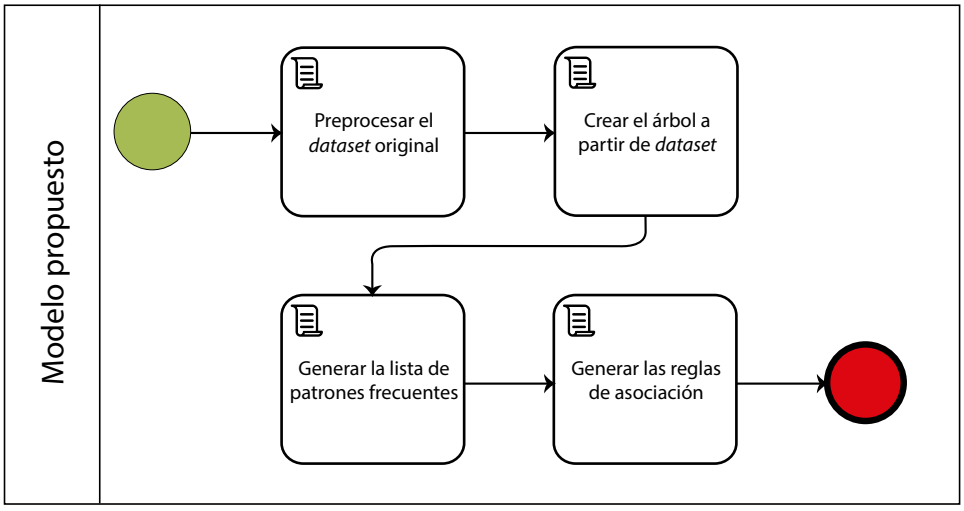

Figura 5. Flujo del modelo propuesto

Elaboración propia

Antes de describir en qué consisten los pasos mencionados, es necesario entender la estructura del *dataset* que se posee, porque esta estructura define el enfoque a tomar para las acciones respectivas al preprocesamiento, a cómo realizar un análisis exploratorio y cómo leer esta *data*, y la estructura que este conjunto de datos debe tener para poder ser alimentado al modelo. El *dataset* en posesión es una fracción en dimensión de tiempo de los datos almacenados en una base de datos (BD) relacional para un *retailer* del sector autoservicio. El nombre del *retailer* permanecerá confidencial por mutuo acuerdo. Sin embargo, es posible detallar que esta fracción pertenece a registros correspondientes a la temporada verano. Este dato será clave en el apartado de resultados, ya que se da contexto al *output* del modelo. Al ser un reflejo de la BD, se tiene un archivo de tipo *csv* para cada entidad, en total cuatro: pasillos, categorías, órdenes y productos. Estas se relacionan a partir de identificadores (Id) únicos para cada registro de cada tabla, como una BD transaccional.

Para el módulo de preprocesamiento se eliminaron todos los registros NA, como primer paso. Después se procedió a determinar las variables o campos factores, que en este caso vendrían a ser ProductID y OrderId. Estas variables permiten la agrupación de los ítems por transacción. Esto quiere decir, si n número de ítems pertenecen a la misma transacción t, esto se traducirá a una lista de transacciones donde t contendrá los n ítems. Finalmente, ya que el modelo necesita una lista de tuplas compuestas del índice de la transacción y de la lista de ítems en esa transacción, se selecciona la lista de ítems producto de la agrupación realizada en el paso anterior. Asimismo, para la exploración de la *data* se utilizó el componente SQL de Spark. Este permite realizar consultas SQL a los *dataframes* de Spark, algo que facilita el tratamiento de la *data* ya que esta posee una estructura relacional. No se detallarán los resultados de la exploración de la *data* dado que eso fue realizado en el apartado 4.1.

Con respecto al módulo para la generación de los ítems frecuentes y las reglas de asociación, la implementación del algoritmo en Spark requiere de cuatro parámetros, los cuales son la columna en el *dataset* que contiene a la lista de ítems por transacción, el soporte mínimo, la confianza mínima y el número de particiones a realizar el paralelismo del procesamiento. La columna para evaluar es simplemente el nombre de la columna en el *dataframe* Spark que contiene la lista de ítems. El soporte y la confianza mínimos son valores que van en el rango de 0 a 1, y se expresan en porcentaje en el análisis. Los valores para estos dos parámetros dependen de las necesidades de la empresa y es ella la encargada de realizar el estudio correspondiente para establecer estos valores. Debido a que el análisis de las necesidades de la empresa es personalizado e implica muchos aspectos a considerar, este escapa del alcance de esta investigación. El número de particiones es necesario cuando se desea ejecutar el modelo creado en un clúster de computadoras. Puesto que en este trabajo todo procesamiento es ejecutado desde un solo ordenador, este parámetro no impacta en el desempeño real del modelo. Posteriormente, ya definidos los parámetros para el modelo, se pasa a realizar el ajuste de este con el conjunto de datos especificado. El resultado será el modelo adaptado a los valores dados a partir de la *data* alimentada. Una vez realizado esto, se pueden obtener los ítems frecuentes y reglas de asociación gracias a las funciones ya definidas en el *framework* de Apache Spark.

La abstracción de la lógica detrás de la generación de ítems frecuentes y reglas de asociación, se definen en los algoritmos *find_frequent_patterns* y *generate_association_rules*. La última es la que genera las reglas de asociación a partir de los patrones frecuentes encontrados. Respecto al algoritmo *find_frequent_patterns*, se instancia un objecto de clase árbol con los parámetros transacciones y umbral de soporte mínimo para filtrar los patrones frecuentes. Este objeto hará llamado a la función *mine-tree*, esta función actuará dependiendo de la condicional de si el árbol contiene un solo camino o rama que nace desde el nodo o contiene múltiples caminos o ramas. Si el árbol tiene un solo camino se generará la lista de patrones frecuentes. Si el árbol tiene muchos caminos se minarán estos subárboles y se tendrá una sublista de patrones por subárbol, estas sublistas luego serán insertadas en el diccionario principal de patrones, el cual tendrá aquellos patrones por subárbol que superen el mínimo de soporte establecido. Con respecto al manejo de la clase árbol, este se generará y trabajará de manera común, con recursividad. Respecto al algoritmo *generate-association-rules*, tomará el diccionario que generó *find-frequent-patterns* y lo iterará para generar un diccionario que tenga como clave al antecedente y como valores al consecuente y a la confianza. La confianza representa la probabilidad de que el consecuente se presente cuando exista el antecedente.

ALGORITHM 1: *find-frequent-patterns* (transacciones, soporte mínimo)

Tree <- Instanciar un objeto árbol con parámetros el dataset y el soporte mínimo deseado

IF tree tiene un solo camino:

    Generar diccionario de patrones frecuentes

ELSE:

Generar sublista de patrones frecuentes por cada subárbol

Insertar sublista de patrones frecuentes en el diccionario principal de patrones frecuentes

Output <- diccionario de patrones frecuentes

End

ALGORITHM 2: *generate-association-rules* (patrones, confianza mínima)

FOR itemset IN patrones.clave:

    soporte_superior <- patrones[itemset]

FOR index IN itemnset.longitud:

FOR antecedente IN itemset[i]:

IF antecedente se encuentra en patterns:

soporte_inferior <- patrones[antecedente]

confianza < - soporte_superior / soporte inferior

IF confianza >= confinaza_mínima:

Reglas[antecedente] = (consecuente, confianza)

Outpur <- Reglas

End

ALGORITHM 3: FP-Growth

Patrones <- find_frequent_patterns(dataset, soporte_min)

Reglas <- generate_association_rules(patrones, confianza_min)

Output <- Reglas

End

5. RESULTADOS Y DISCUSIÓN

El modelo se inicializó con los valores 0,01 para soporte mínimo y 0,1 para confianza. Los ítems frecuentes encontrados que sobrepasen el umbral de soporte están conformados por dos ítems. Esto quiere decir que no existen relaciones entre 3 ítems que pasen el 1 % del total de transacciones. En la figura 6 se puede apreciar los 16 primeros ítems frecuentes según su frecuencia. La relación más fuerte existente entre fresas orgánicas y bolsa de plátanos orgánicos. Existen un total de 131 209 transacciones, por lo que la relación descrita aparece el 2,34 % de veces.

Referente a las reglas de asociación, la relación más fuerte es entre palta Hass orgánica (antecedente) y bolsa de plátanos orgánicos (consecuente). Esta relación se refiere a la probabilidad de que ocurra el consecuente a partir del antecedente. Es decir, hay un 33,18 % de que la bolsa de plátanos orgánicos sea comprado cuando se compra palta Hass orgánica. A diferencia de los ítems frecuentes, este examina la relación pirobalística entre dos o más ítems, mientras que los ítems frecuentes es una descripción frecuencial de las apariciones conjuntas de los ítems. En la figura 7 se pueden apreciar las 20 reglas de asociación más fuertes según su confianza.

La aparición constante de productos de tipo fruta o verdura se debe a que la fracción del *dataset* utilizado corresponde a la temporada de verano, por lo que es comprensible este comportamiento. Esto indica que es importante realizar un análisis exploratorio previo a la *data* para entender, luego, los resultados obtenidos.

No se decidió por realizar más pruebas con diferentes parámetros porque estos dependen enteramente de las necesidades de negocio del *retailer*. Los resultados obtenidos de otra combinación de parámetros no son más ni menos correcta que la realizada en este trabajo. El punto clave para la obtención de los mejores resultados es dedicar el esfuerzo al entendimiento y tratamiento de la *data* de entrada y el estudio profundo de las necesidades de la organización.

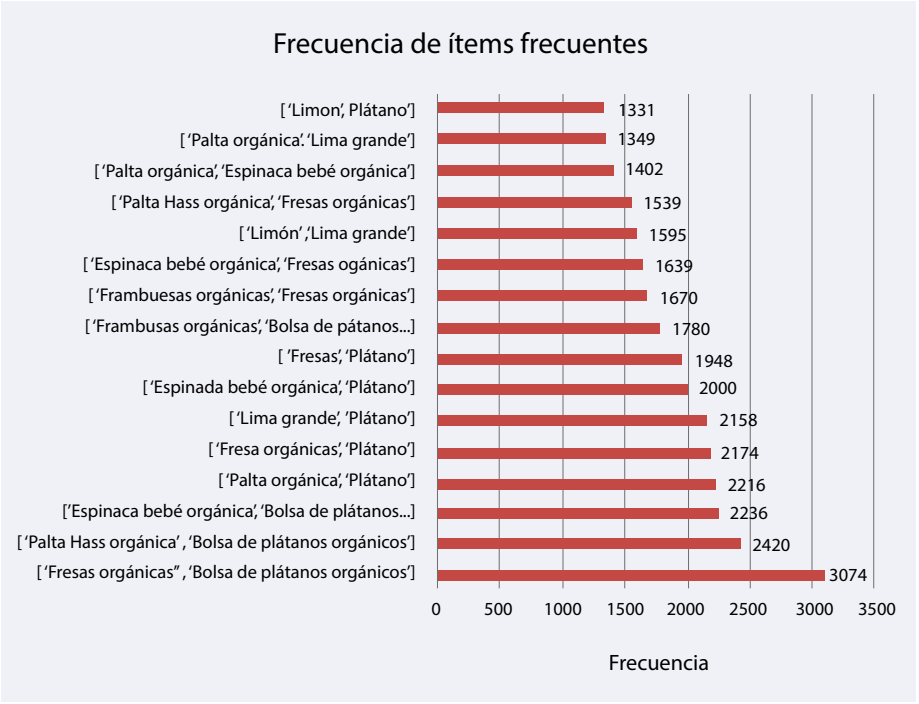

Figura 6. Frecuencia de ítems
Elaboración propia

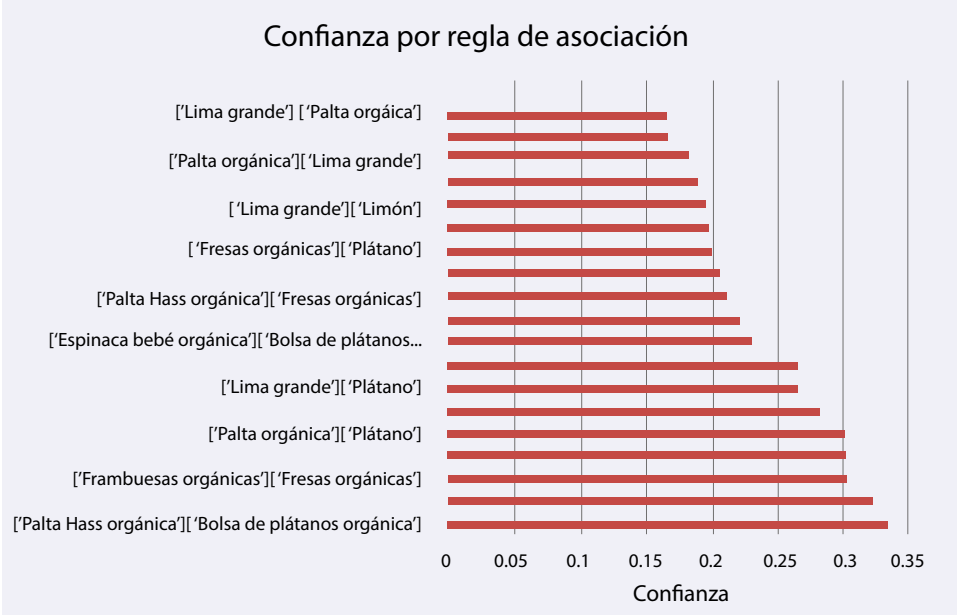

Figura 7. Confianza por regla de asociación
Elaboración propia

Respecto al desempeño del modelo, se realizaron varias pruebas al tiempo de ejecución del modelo y sus componentes. Para esto se identificaron las partes que componen el modelo: el ajuste de la *data* al modelo (*model fit*), la generación de ítems frecuentes, la generación de reglas de asociación, la lectura o carga de la *data*, la preparación de la *data* para alimentar al modelo. En la tabla 1 se pueden observar los tiempos de ejecución para cada uno de estos componentes. El número de evaluaciones hace referencia a cuántas veces se ejecutará el componente correspondiente, mientras que las repeticiones significan cuántas veces esta evaluación será realizada. La salida de este ejercicio es el tiempo más bajo del total de repeticiones, mientras que el tiempo de ejecución es la división del tiempo total de repetición entre el número de evaluaciones. Es decir, para la primera fila, se ejecutó 5 veces el componente *model fit* con 10 evaluaciones cada una, el mínimo tiempo entre estas 5 repeticiones es el tiempo total de repetición (44,21 s), y este tiempo dividido entre el número de evaluaciones da el tiempo promedio (para uso referencial) de cada evaluación (4,42 s).

Es tentador calcular la media y la desviación estándar del vector resultado de las repeticiones. Sin embargo, esto no es muy útil. En un caso típico, el valor más bajo proporciona un límite inferior para la rapidez con la que la máquina pueda ejecutar el fragmento de código dado; los valores más altos en el vector de resultados generalmente no son causados por la variabilidad en la velocidad de Python, sino por otros procesos que interfieren con su precisión de tiempo. Por lo tanto, el mínimo del resultado es probablemente el único número que debería utilizarse como referencia al tiempo de ejecución del componente.

Tabla 1
Tiempo de ejecución de los componentes del modelo en segundos

Tarea	Repeticiones	Número de evaluaciones	Tiempo de ejecución (s)	Tiempo total repetición (s)
model fit	5	10	4,421917761	44,21917761
model fit	10	10	4,375176091	43,75176091
freq items	5	10	0,000235505	0,002355054
freq items	10	10	0,000227227	0,002272273
ass rules	5	10	0,001781023	0,01781023
ass rules	10	10	0,001778434	0,017784339
load data	5	5	11,37215843	56,86079213
load data	5	10	11,8818465	118,818465
prepare data	5	3	8,014044508	24,04213352
prepare data	5	5	7,0520125	35,2600625

Elaboración propia

Si se separan los componentes por metodología, presentados en la tabla 1, en dos grupos, se tendría a *model fit*, *freq ítems* (ítems frecuentes) y *ass rules* (reglas de asociación) como un grupo y a *load data* y *prepare data* como otro. Este último es muy costoso en tiempo porque realiza la carga de la *data* (3 millones de registros para este trabajo) y las asociaciones necesarias. Para el primer grupo, es *model fit* el que toma mayor tiempo pues ajusta la *data* al modelo, el cual crea el árbol FP a partir de los parámetros establecidos, para posteriormente tanto *freq ítems* como *ass rules* realicen la minería de ese árbol. En este punto es donde se aprecian las ventajas de FP-Growth, ya que, una vez realizada la construcción del árbol, la minería de esta y la obtención de información como resultado toma milésimas de segundo para un conjunto de *data* real.

En cuanto al tiempo de ejecución variando la cantidad de registros a procesar, en las tablas 2 y 3 se pueden observar los resultados obtenidos. En la tabla 2 se realiza la misma dinámica que en la tabla 1. Se utilizó el módulo *Timeit* de Python para realizar la medición del tiempo. Como puede observarse, la diferencia de tiempo no es drástica entre las pruebas. Aunque el impacto de subir de 100 000 a 1 000 000 registros fue fuerte y el más notorio entre todos, el siguiente paso arriba (10 000 000) no tiene gran diferencia. La tabla 3 mide lo mismo, la diferencia radica que esta utiliza el módulo *Time* para medir el tiempo. Esa medición, a diferencia de la anterior, es afectada por los procesos que se ejecutan en segundo plano u otras tareas que el ordenador está realizando.

Tabla 2
Tiempo de ejecución del ajuste del modelo en base a la cantidad de registros a procesar en segundos, calculado con el módulo Timeit

TIMEIT.TIMEIT				
Cantidad de registros	Repeticiones	Número de evaluaciones	Tiempo de ejecución (s)	Tiempo total de repetición (s)
1000	5	10	3,535814589	35,35814589
10 000	5	10	3,461978596	34,61978596
100 000	5	10	3,606037736	36,06037736
1 000 000	5	10	4,492759976	44,92759976
10 000 000	5	10	4,421917761	44,21917761

Elaboración propia

Tabla 3
Tiempo de ejecución del ajuste del modelo en base a la cantidad de registros a procesar en segundos, calculado con el módulo Time

TIME.TIME	
Cantidad de registros	Tiempo de ejecución (s)
1000	3,628935814
10 000	3,599936247
100 000	3,862709284
1 000 000	4,239717007
10 000 000	5,348645687

Elaboración propia

6. CONCLUSIONES

Al ser una de las tendencias la generación de *data* y la necesidad de las organizaciones a obtener información y generar conocimientos de esta *data*, unos de los sectores con más oportunidades de minería de *data* es el sector *retail*. Cuando se habla de la minería de reglas de asociaciones siempre se ha tenido como referente al algoritmo Apriori, sin embargo, el algoritmo utilizado en este trabajo se presenta como una alternativa a las deficiencias del algoritmo clásico. FP-Growth propone la abstracción de la *data* en una estructura árbol y luego la minería de este, reduciendo así los costos en tiempo y recursos computacionales.

En este trabajo se realiza la implementación de FP-Growth para un conjunto de datos de una empresa de *retail* peruana perteneciente a los autoservicios. Se realiza una implementación

en paralelo haciendo uso del *framework* Apache Spark, el cual tiene como ventaja hacer procesamiento paralelo, con una limitante, todo fue construido sobre un solo ordenador, lo que no permite explotar las capacidades del procesamiento en clúster. Existen otras soluciones como la implementación en clústeres o en máquinas virtuales, asimismo existen soluciones novedosas implementadas en la nube como es el servicio de Amazon. Ellos brindan un ambiente de clústeres preparados para funcionar con el *framework* Spark.

Finalmente, hay que considerar que las necesidades del negocio y sus objetivos estratégicos juegan un papel dominante y crítico en la implementación de un modelo como el propuesto en este trabajo. Este análisis define los parámetros de funcionamiento del modelo y los resultados obtenidos tienen que ser relevantes y consistentes con la información del negocio.

REFERENCIAS

Agrawal, R., Imienlinski, T., y Swami, A. (1993). Mining association rules between sets of items in large databases. *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, *22*(2), 207-216. doi:10.1145/170035.170072

Bhandari, A., Gupta, A., y Das, D. (2015). Improvised Apriori algorithm using frequent pattern tree for real time applications in data mining. *Procedia Computer Science*, *46*, 644-651. doi:10.1016/j.procs.2015.02.115

Chunhua, J., y Dongjun, N. (2008). Distributed mining model and algorithm of association rules for Chain retail enterprise. *Proceedings - ISECS International Colloquium on Computing, Communication, Control, and Management*, *CCCM 2008*, *3*, 235-239. doi:10.1109/CCCM.2008.129

Di Fatta, G. (2019). Association rules and frequent patterns. En S. Ranganathan, K. Nakai, y C. Schonbach, (Eds.), *Encyclopedia of Bioinformatics and Computational Biology* (pp. 367-373). doi:10.1016/B978-0-12-809633-8.20333-6

Fournier-Viger, P., Lin, J. C. W., Vo, B., Chi, T. T., Zhang, J., y Le, H. B. (2017). A survey of itemset mining. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, *7*(4), 1-41. doi:10.1002/widm.1207

Galarreta Vásquez, J. (2016). Inducción de reglas de asociación de minería de datos en base de datos de entidad retail. *Rev. Ingeniería: Ciencia, Tecnología e Innovación*, *3*(2).

Girotra, M., Kanika, N., Saloni, M., y Neha, S. (2013). Comparative survey on association rule mining algorithms. *International Journal of Computer Applications*, *84*(10), 975-8887. Recuperado de https://pdfs.semanticscholar.org/08a7/a7c571159d14660a402a4460ee1f828c5fed.pdf

Griva, A., Bardaki, C., Pramatari, K., y Papakiriakopoulos, D. (2018). Retail business analytics: Customer visit segmentation using market basket data. *Expert Systems with Applications*, *100*, 1-16. doi:10.1016/j.eswa.2018.01.029

Han, J., Kamber, M., y Pei, J. (2011). *Data mining: Concepts and techniques*. doi:10.1016/C2009-0-61819-5

Han, J., Pei, J., y Yin, Y. (2000). Mining frequent patterns without candidate generation. *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, 1-12.

Hussain, R. Z., y Srivatsa, S. K. (2014). A study of different association rule mining techniques. *International Journal of Computer Applications*, *108*, 10-15.

Kaur, M., y Kang, S. (2016). Market Basket Analysis: Identify the changing trends of market data using association rule mining. Procedia. *Procedia Computer Science*, *85*, 78-85. doi:10.1016/j.procs.2016.05.180

Tan, P., Steinbach, M., y Kumar, V. (2005). *Introduction to Data Mining. Discovering Knowledge in Data*. doi:10.1002/0471687545.ch1

Xue, L., Wang, H., Liu, S., y Li, C. (2010). The application of data mining in online bookstore. *2010 International Conference on Machine Learning and Cybernetics*, 1294-1298. doi:10.1109/ICMLC.2010.5580892