ESCUELA SUPERIOR POLITÉCNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"MEJORAMIENTO DE LA GESTIÓN DE PLANIFICACIÓN DE COMBOS EN LA INDUSTRIA DE E-COMMERCE, MEDIANTE UN MODELO PREDICTIVO DE REGLAS DE ASOCIACIÓN – MINERÍA DE DATOS"

TRABAJO DE TITULACIÓN

Previa a la obtención del Título de:

MAGÍSTER EN SISTEMAS DE INFORMACIÓN GERENCIAL

Autor:

STALIN ALBERTO ARROYABE MERCHÁN

Guayaquil - Ecuador

2020

AGRADECIMIENTO

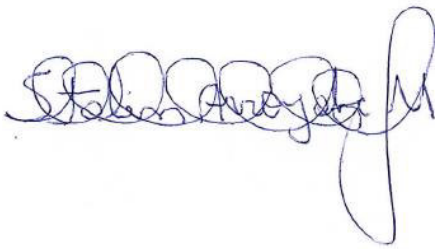
A mi amado Dios, Creador y Señor del Universo que nos bendice diariamente.

A mi amada Madre Angelita, que desde el cielo me cuida y guía hacía el crecimiento.

A mi amada hija Victoria, que me llena de mucha alegría e impulsa a progresar.

A mi familia, quienes han sido y son un pilar fundamental en mi vida.

A mis profesores, por todos los conocimientos y consejos impartidos.



Stalin Alberto Arroyabe Merchán.

DEDICATORIA

A Dios

A mi Madre

A mi Padre

A mi Hija

A mis Hermanos

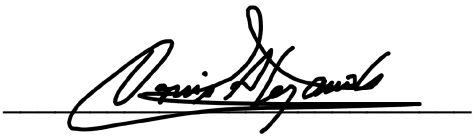
A Mirna y Leonardo

A mis Profesores

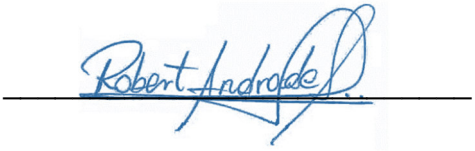
A mis Amigos

TRIBUNAL DE SUSTENTACIÓN

Mgs. Lenin Freire Cobo
COORDINADOR MSIG



Ph.D. Otilia Alejandro Molina
DIRECTORA DEL PROYECTO DE GRADUACIÓN



Mgs. Robert Andrade Troya
MIEMBRO DEL TRIBUNAL

RESUMEN

Las empresas utilizan las promociones de artículos para incrementar el consumo, bajar los niveles de stock, captar y retener a los clientes. Las promociones son “Combos de artículos” enfocados a grupos de clientes y su frecuencia de uso puede ser semanal, mensual e incluso diaria.

Por lo general en las empresas el departamento de Marketing es el encargado de definir los “Combos de artículos” que luego son aprobados por las áreas comerciales, logísticas, financieras y de producción. Usualmente el proceso de generación de los “Combos de artículos” es manual y complejo originando así que demore mucho tiempo su elaboración.

Existen herramientas de software que mediante algoritmos de Data Mining permiten encontrar los patrones que asocian a los artículos con los clientes. A estos patrones se los denomina “Reglas de asociación” y es muy importante que el nivel de detalle sea el más específico(bajo) posible.

Evaluaremos 2 algoritmos predictivos de asociación que se encuentran en distintas plataformas: APRIORI en Java(Weka) y FP-GROWTH en Python. Seleccionaremos aquel modelo predictivo cuyo tiempo de ejecución sea el más bajo y que genere “Reglas de asociación” que involucren el mayor número de categorías de artículos y clientes.

ÍNDICE GENERAL

AGRADECIMIENTO I

DEDICATORIA II

TRIBUNAL DE SUSTENTACIÓN III

RESUMEN IV

ÍNDICE GENERAL V

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABLAS X

INTRODUCCIÓN XI

CAPÍTULO 1 1

GENERALIDADES 1

1.1 Antecedentes 1

1.2 Definición del problema 3

1.3 Solución Propuesta 6

1.4 Objetivo General 6

1.5 Objetivos específicos 7

CAPÍTULO 2 9

MARCO TEÓRICO 9

2.1. Conceptos Básicos 9

2.1.1. Conjunto de Datos (DataSet) 9

2.1.2. Atributo (Item) 10

2.1.3. Transacciones (ItemSet) 10

2.1.4. Data Mining – Minería de Datos 10

2.1.5. Canasta de Compra 14

2.2. Reglas de Asociación 16

2.2.1. Métricas de Evaluación 17

2.2.1.1. Soporte 18

2.2.1.2. Confianza 19

2.2.1.3. Lift 20

2.2.1.4. Conviction 23

2.3. Herramientas utilizadas en la elaboración de los modelos 25

2.3.1. Lenguajes de Programación Utilizados 25

2.3.1.1. Java y el Data Mining 25

2.3.1.2. Python y el Data Mining 27

2.3.1.3. Pandas 27

2.3.2. Entornos de Trabajos 27

2.3.2.1. Weka 27

2.3.2.2. Anaconda 28

2.3.2.3. Jupyter Notebook 28
2.3.3. Algoritmos de Reglas de Asociación 28
2.3.3.1. APRIORI 29
2.3.3.2. FP-Growth 30
2.3.4. Otros Algoritmos de Reglas de Asociación 31
2.3.4.1. Algoritmo PAFI 31
2.3.4.2. Algoritmo ECLAT 32
2.4. Fases en la implementación de los modelos 33
2.4.1. Principales Metodologías en proyectos de Data Mining. 33
2.4.2. Definición de los Pasos en el desarrollo del modelo predictivo . 34
CAPÍTULO 3 36
ANÁLISIS, DISEÑO E IMPLEMENTACIÓN 36
Ejecución de la Fases 36
3.1 Comprensión de los Datos 36
3.1.1 Datos Originales 36
3.1.2 Especificación de los Datos 37
3.1.3 Exploración de los Datos 38
3.2 Preparación de los Datos 39
3.3 Construcción de los Modelos 42

3.3.1 Modelo APRIORI JAVA 42

3.3.2 Modelo FP-Growth PYTHON 48

CAPÍTULO 4 64

PRUEBAS Y ANÁLISIS DE RESULTADOS 64

4.1. Análisis Comparativo de Modelos 64

4.2. Análisis de los Combos de Artículos Generados 67

4.3. Selección del Modelo Predictivo 75

CONCLUSIONES Y RECOMENDACIONES 77

BIBLIOGRAFÍA 82

ÍNDICE DE FIGURAS

Figura 2.1 - Ejemplo de transacciones_____ 10

Figura 2.2 - Data Mining dentro de las fases de Los Sistemas KDD _____ 12

Figura 2.3 - Ejemplo del Algoritmo APRIORI [54] _____ 30

Figura 2.4 - Evolución de las Metodologías DM [9] _____ 34

Figura 3.1 - Diagrama de Modelo Relacional de los Datos [58]_____ 37

Figura 4.1 - FP Growth: Las 10 mejores Reglas de Asociación - Indicadores Estadísticos _____ 68

Figura 4.2 - FP Growth: % Participación de las categorías en los Combos_____ 69

Figura 4.3 - FP Growth: % Participación de las categorías en los condicionales_ 70

Figura 4.4 - APRIORI: Las 10 mejores Reglas de Asociación en - Indicadores Estadísticos _____ 72

Figura 4.5 - APRIORI: % Participación de las categorías en los Combos _____ 73

Figura 4.6 - APRIORI: % Participación de las categorías en los condicionales__ 74

ÍNDICE DE TABLAS

Tabla 1 - Exploración de Datos.....39

Tabla 2 - Preparación de Datos 41

Tabla 3 - Paso b) del modelo APRIORI - Métricas por Atributos 43

Tabla 4 - Paso d) del modelo APRIORI - Parámetros de Entrada Algoritmo 44

Tabla 5 - Paso e) del modelo APRIORI - Resultados 45

Tabla 6 - Paso a) Python FP-GROWTH - Parámetros de Entrada 49

Tabla 7 - Paso b) Python FP-GROWTH – Resultados de Categorías por Código.50

Tabla 8 - Paso c) Python FP-GROWTH – Resultados de Códigos Booleanos.....51

Tabla 9 - Paso f) Python FP-GROWTH – Resultados de las 10 Primeras Reglas Generadas 55

Tabla 10 - Paso g) Python FP-GROWTH – Resultados de Reglas Únicas 58

Tabla 11 - Paso i) Python FP-GROWTH – Resultados de las Mejores 10 Reglas 61

Tabla 12 - Comparación de Modelos – APRIORI y FP - Growth 65

Tabla 13 - Modelo FP Growth Python – Las 3 mejores Reglas 70

Tabla 14 - Modelo APRIORI Java – Las 3 mejores Reglas 74

Tabla 15 - Evaluación de los objetivos planteados 77

INTRODUCCIÓN

Las empresas del tipo “Industrial-Comercial” y las de “Retail”, tienen un mercado muy competitivo y en algunos casos pequeño. Estas empresas se esfuerzan para permanecer en los mercados y por eso ponen en marcha diferentes tipos de estrategias para así cumplir con los objetivos empresariales de aumentar las ventas, sin que afecte la rentabilidad del negocio, y de captar-retener a los clientes.

Una estrategia muy utilizada son las ofertas-promociones, también llamados “Combos de artículos”. Este tipo de estrategia permite focalizar, por grupo de clientes, la venta de productos con algún incentivo de compra (más unidades por precio, artículos asociados con descuento, entre otros).

Los “Combos de artículos” son asociaciones de productos en donde se definen unidades y precios con descuento. Un ejemplo de combo es vender leche y pan a un precio de $4 cuando por separado cuestan $5.

En la definición de los “Combos de artículos” es muy relevante las características de los clientes que se ingresan al sistema central, por ejemplo: edad, sexo, genero, lugar de residencia, entre otros. Muchas veces hay errores en estos datos y constantemente se actualizan para la correcta ejecución de los procesos empresariales. Un ejemplo de proceso empresarial es la definición y ejecución de la estrategia “Combos de artículos”.

Las características de los clientes permiten su clasificación o segmentación. Un ejemplo de segmentación de clientes en las empresas del tipo “Industrial-Comercial” son los Clientes Minoristas (Tiendas de Barrio) y los Clientes Mayoristas (Supermercados, Gasolineras, entre otros). Para las empresas del tipo Retail, la segmentación también se puede realizar mediante los hábitos de compra.

A la unificación de los “Combos de artículos” con la segmentación de los clientes se le conoce como “Canasta de compra” [33]. Un ejemplo de “Canasta de compra” puede ser un combo de artículos para los clientes de entre las edades de 20 a 25 de la región sur del país que compran desodorante con detergente los lunes y los miércoles.

Una de las formas de implementar la “Canasta de compra” es mediante técnicas de Data Mining. Una de las principales características del Data Mining, es encontrar patrones en grandes volúmenes de datos. La validación y la ejecución de patrones, mediante alguna estrategia comercial, permite generar nuevo valor empresarial y si la estrategia es exitosa, con el paso del tiempo, se transforma en conocimiento [2].

Una de las técnicas del Data Mining usada para implementar la “Canasta de compra”, son las “Reglas de asociación”. Esta técnica permite encontrar patrones frecuentes y los representa en forma de reglas, en donde una “Regla de asociación” es una fuerte relación entre la venta de varios productos [10].

Los patrones frecuentes utilizados en las “Reglas de asociación” también pueden ser encontrados por otra técnica del Data Mining conocida como “Árboles de

decisión" [32], que optimiza el uso de los recursos computacionales y reduce el tiempo de ejecución en la generación de las reglas.

Un ejemplo del uso de "Reglas de asociación" es el estudio realizado en 1992 por la empresa de consultoría NCR, la cual encontró que en las farmacias OSCO existía una relación entre la venta de pañales y cerveza en los horarios de 5pm – 7pm [53].

Otro ejemplo del uso de "Reglas de asociación" es el orden de ejecución de las 4Ps(Price, Product, Place, Promotion) en un conjunto de Retailers en Gweru Zimbabue. Se encontró que, en el proceso de decisión de compra los clientes otorgan más importancia a los precios, es por eso por lo que se deben establecer una mayor cantidad de promociones con descuento para diferentes grupos de productos [55].

Otro ejemplo del uso de "Reglas de asociación" es el diseño de las estanterías o perchas en un retail minorista en Turquía. Se ubicó, en 300 metros cuadrados de espacios, los 6 grupos de productos con mayor frecuencia de compra [56]. Para este tipo de casos de estudio se puede agregar promociones entre grupos de productos afines para así incentivar al cliente a realizar un viaje más lago por la tienda, por ejemplo: si los grupos de productos lácteos y cereales se encuentran muy separados, se puede colocar una promoción de descuento en el pasillo que separa las parches [57].

Tomando en cuenta todo lo enunciado anteriormente, se implementará para un negocio de E-commerce, un modelo predictivo basado en "Reglas de asociación"

que genere la “Canasta de compra” con los mejores “Combos de artículos” por segmentación de cliente.

CAPÍTULO 1

GENERALIDADES

1.1 Antecedentes

Las campañas promocionales son las estrategias de ventas más utilizadas por las empresas. Un ejemplo de estas campañas es los “Combos de artículos” definidos por grupos de clientes. Este tipo de estrategia es una de las áreas de aplicación de la “Canasta de Compra” [34].

Se pueden definir “Combos de artículos” a nivel macro-general o a nivel micro-específico. Este tipo de promociones son agrupaciones de productos por grupo de clientes mayoristas o por grupos de clientes

minoristas. Ambos necesitan manipular grandes volúmenes de datos, cada uno tiene sus ventajas o desventajas.

Los “Combos de artículos” micro-específico ofrecen ventajas tales como: la personalización por grupos de clientes, la estabilización del nivel de inventario en los puntos de venta, la formación de promociones entre los artículos con sobre stock y los artículos de alta rotación. Las desventajas de los “Combos de artículos” micro-específico son: involucran más trabajo operativo por el empaquetamiento - en una envoltura maestra – de diferentes unidades y tipos de artículos, muchas promociones a registrar en el sistema de las campañas y el costo-beneficio de la publicidad.

Los “Combos de artículos” macro-general ofrecen ventajas tales como: realización de campañas publicitarias por sectores geográficos, menor costo y esfuerzo al empaquetarlos - porque se establece una envoltura maestra – y nuevos descuentos con proveedores por los grandes volúmenes de compra en materias primas o productos terminados (Retail). Las desventajas de los “Combos de artículos” macro-general son: no se definen a nivel de características específicas de clientes, no se priorizan las existencias de los puntos de venta y pueden demandar mayor costo y esfuerzo en movilización la mercadería a los puntos de venta.

Para la elaboración de los “Combos de artículos” se puede usar herramientas de software comercial que incluyan técnicas de Data Mining, por ejemplo, Microsoft ofrece modelos basados en “Reglas de asociación” en su programa de Business Intelligence Development Studio (BIDS) [36].

Otras herramientas son: IBM SPSS Modeler, Oracle Data Mining, SAS Enterprise Miner, STATISTICA Data Miner [37].

También se pueden usar herramientas open source - en plataformas como JAVA, C y Python - que incluyan técnicas de Data Mining para la elaboración de los “Combos de artículos”, tales como: RapidMiner, Weka, KNIME, R, Orange, Scikit-Learn [38].

Aunque las herramientas existan, las empresas no aplican el Data Mining ya sea por desconocimiento, costo, o falta de personal calificado en esta temática. Al no usar estas herramientas se ocasiona la pérdida del beneficio de identificación y generación de patrones e impide la automatización de las estrategias enfocadas a mejorar la venta y rentabilidad empresarial como, por ejemplo: los “Combos de artículos”. Es por eso por lo que es más común la generación manual de los “Combos de artículos”.

1.2 Definición del problema

La elaboración manual de los diferentes tipos de “Combos de artículos”, requiere de varios aspectos: tener consolidada toda la plataforma de Inteligencia de Negocios(BI) - Data Warehouse y Herramienta de Análisis -, tener datos confiables que sean consistentes entre los diferentes sistemas de la empresa y tener usuarios funcionales de tipo gerencial, tales como: Analistas Senior, Jefes y Gerentes de las áreas Comerciales, Marketing e Información, que conozcan del negocio y lo analicen a profundidad.

La elaboración manual de los “Combos de artículos” es un proceso largo debido a las distintas fases y actores involucrados. Además, la información a procesar es de varios años y se encuentra en un nivel muy bajo de granularidad-detalle: cliente o alguna categoría y artículo o alguna categoría.

En el transcurso del proceso de elaboración manual de los “Combos de artículos”, el usuario pierde el interés por las dificultades en el procesamiento de datos, tiempo y limpieza. Al final se establecen los mismos combos-promociones de periodos anteriores. Los resultados de estas decisiones son casi siempre los mismos: mínima participación en la venta total, son globales para todos los clientes, altos costos de publicidad - en comparación a la venta obtenida - y baja captación de nuevos clientes.

Por ejemplo, una empresa cervecera necesita definir una estrategia de venta que le asegure un incremento mínimo del 5% en los días del siguiente feriado de carnaval versus los días del carnaval anterior. Esta estrategia debe presentarse como máximo con 12 semanas de anticipación al carnaval. La Dirección General le solicita, la tercera semana de octubre, al departamento de marketing que elabore la estrategia y ellos tienen alrededor de 4 semanas, es decir el plazo vence la tercera semana de noviembre puesto que el feriado de carnaval es la tercera semana de febrero. El proceso de elaboración manual de “Combos de artículos” es el siguiente: Los usuarios funcionales: Analistas Senior y Jefes, descargan, consolidan y cuadran la información de venta por cliente, producto y día

del carnaval anterior. Luego intentan identificar patrones y simular escenarios de venta en base a los objetivos esperados – como mínimo incrementar en un 5% la venta en este carnaval vs el carnaval anterior. Los usuarios funcionales evalúan que no podrán cumplir en el tiempo acordado – porque también tienen que definir otras estrategias y realizar las actividades del día a día - y se reúnen con la Gerencia de Marketing para acordar repetir los combos y definir el objetivo de venta en un 7% (2% más de lo mínimo esperado). Luego presentan la propuesta a las áreas de producción, finanzas y comercial para alcanzar acuerdos grupales tales como: cantidades mínimas a producir, tiempo de entrega en los puntos de venta, descuento aplicado por cada promoción, gasto de publicidad y comisión por cumplimiento de venta a los vendedores para que ellos incentiven la compra del cliente. El área de marketing envía la propuesta a la Dirección General y espera su aceptación. Este proceso puede volverse cíclico hasta que la Dirección General apruebe la estrategia.

La elaboración manual de los “Combos de artículos” requiere aspectos de procesamiento de datos, aprobación e intervención de usuarios. Los usuarios funcionales pierden la paciencia y las ganas de trabajar ya que ocupa demasiado tiempo y capacidad de cómputo, siendo una posible solución automatizar el proceso de generación manual de estos combos a través del uso de herramientas de Data Mining como las reglas de Asociación.

1.3 Solución Propuesta

Generar automáticamente los “Combos de artículos” con la mayor probabilidad de éxito a través de un modelo predictivo - de tipo asociativo usando herramientas de software open source.

Los siguientes pasos son necesarios para realizar la solución propuesta:

- Definir un conjunto de datos con las mínimas características esperadas para un proyecto de Data Mining
    - Cantidad de historia y nivel de detalle.
- Depurar el conjunto de datos
    - Elegir las dimensiones más apropiadas.
    - Disminuir la dispersión de las dimensiones elegidas.
- Utilizar herramientas open source
    - Configurar los parámetros mínimos requeridos para la ejecución de los algoritmos APRIORI y FP-Growth.
- Evaluar los resultados de la ejecución de los algoritmos escogidos, en base a indicadores estadísticos y a las reglas de asociación arrojadas.

1.4 Objetivo General

Elaboración de un modelo predictivo - del tipo asociativo - que permita mejorar la generación de los “Combos de artículos”, a través de la focalización de las promociones, llevándolas al nivel de alguna categoría de artículo y alguna categoría cliente.

1.5 Objetivos específicos

Se definen los siguientes objetivos específicos en el contexto de la solución propuesta y al objetivo general planteado:

- Especificar las características del conjunto de datos
    - Análisis del conjunto de datos, para así definir:
        - Cantidad mínima de información.
        - Dimensionalidad.
        - Dispersión apropiada de los valores de las dimensiones.
- Realizar los modelos predictivos, de tipo asociativo, en las herramientas open source escogidas.
    - Aplicar el algoritmo APRIORI al conjunto de datos, usando la herramienta Weka.
    - Aplicar el algoritmo FP-GROWTH al conjunto de datos, usando las herramientas basadas en Python.
        - Realizar modificaciones al algoritmo FP-GROWTH para mejorar el desempeño y la visualización de las reglas generadas.
        - Realizar modificaciones al conjunto de datos utilizado por el algoritmo FP-GROWTH.
- Evaluar los modelos predictivos y escoger el mejor en base a las reglas arrojadas, métricas de evaluación y costo computacional
    - Comparar los tiempos de ejecución.
    - Comparar las reglas de asociación generadas:

- Nivel de detalle o granularidad.
- Indicadores estadísticos que miden la probabilidad de éxito.
- Cantidad de reglas arrojadas.

En el siguiente capítulo nos enfocaremos en estudiar los conceptos teóricos necesarios para comprender la solución planteada.

CAPÍTULO 2

MARCO TEÓRICO

El objetivo del capítulo es realizar una breve introducción en la terminología, conceptos e ideas relacionadas a la temática que se está tratando.

Nos introducimos en los conceptos Data Mining, Reglas de Asociación (junto a sus métricas de evaluación) y las Tecnologías usadas en la implementación de los modelos predictivos a desarrollar.

2.1. Conceptos Básicos

2.1.1. Conjunto de Datos (DataSet)

Se entiende como “Conjunto de Datos” o “DataSet” a una colección que almacena información [40].

2.1.2. Atributo (Item)

Se entiende como atributo o ítem o columna o campo a subcomponentes de un registro de datos. Por ejemplo, en el registro de datos de una población, los atributos serían apellido, sexo y edad [39].

2.1.3. Transacciones (ItemSet)

Las transacciones son registros conformados por atributos que por lo menos tienen un valor distinto para cada fila del conjunto de datos. Por ejemplo, si un cliente realiza varias compras, cada una sería un registro diferente, con elementos asociados al ID de cliente. A esta forma de representación también se le conoce como datos tabulares [23].

ID_Cliente	Mantequilla	Pan	Leche	Huevos	Mermelada
1	TRUE	FALSE	FALSE	TRUE	FALSE
2	FALSE	FALSE	TRUE	TRUE	FALSE
3	TRUE	TRUE	FALSE	TRUE	FALSE
4	TRUE	TRUE	TRUE	TRUE	FALSE

Figura 2.1 - Ejemplo de transacciones

2.1.4. Data Mining – Minería de Datos

Se puede definir al Data Mining como el descubrimiento automático de patrones o modelos no obvios, escondidos en una base de datos,

los cuales tienen un gran potencial para contribuir en los aspectos principales del negocio [2].

Al Data Mining también se lo encasilla como una fase de los Sistemas KDD (Knowledge Discovery from Databases – Conocimiento extraído de las bases de datos).

- Los sistemas de Extracción del Conocimiento – KDD – permiten encontrar, desde los datos, patrones válidos y comprensibles [3].
- Las Fases previas al Data Mining (Selección, Pre-Procesamiento y Transformación), dependen mucho del alcance del escenario planteado para encontrar nuevos conocimientos.
- Las Fases posteriores al Data Mining (Evaluación y Conocimiento), dependen mucho de la interpretación que los usuarios le dan al nuevo conocimiento y su confianza para ejecutarlo en el negocio.

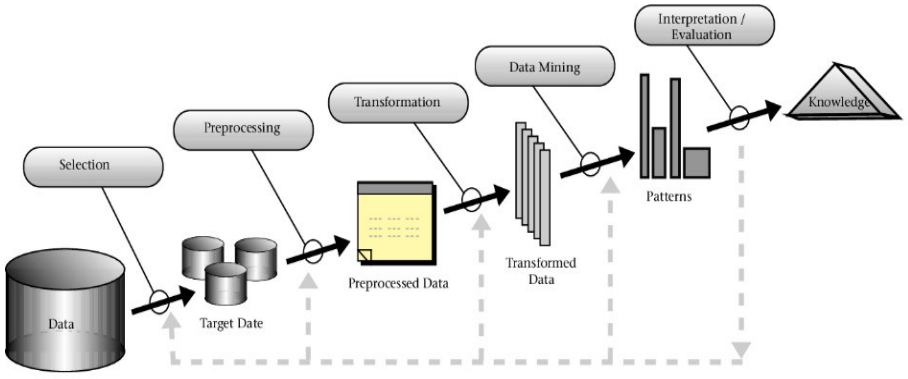


Figura 2.2 - Data Mining dentro de las fases de Los Sistemas KDD

Desde que en 1980 apareció el término de Data Mining, muchas empresas han realizado este tipo de proyectos para ayudar a la toma de decisiones [5]. Por eso existen numerosas técnicas de Data Mining que apoyan a la generación de nuevo conocimiento:

- Un ejemplo es el uso de la técnica de “Clustering” en la segmentación de clientes RFM – Proximidad de la última compra, frecuencia de compra y Monetización - realizada por un Banco Comercial en Tailandia. El caso de estudio utilizó los algoritmos SOM y K-Means e identificó el uso por grupo de clientes de la plataforma web del banco. El grupo de clientes más pequeño utiliza con mayor frecuencia la plataforma y el servicio más utilizado fue la transferencia a terceros, además la mayoría de los clientes del grupo más grande se encuentran inactivos [45].
    - Este tipo de técnicas ayudan a fortalecer las estrategias que permiten la retención y el aumento del consumo de los

clientes, mientras se satisfacen del bien o servicio ofrecido (Customer-Share) [41].

- Otro ejemplo es el uso de la técnica de “Redes Neuronales” en la predicción del precio de las acciones DAX - índice calculado para las 30 compañías más grandes de Alemania. El modelo predictivo es del tipo BNNMAS (Bat-Neural Network Multi-Agent System) y sus principales ventajas son la precisión y confianza en la predicción del precio de la acción - para largos periodos de tiempo y con variaciones significativas por las crisis financieras no esperadas - en comparación con otros tipos de redes neuronales: GANN (Algoritmos Genéticos) y GRNN (Regresión Generalizada) [46].
- Otro ejemplo es el uso de la técnica de “Support Vector Machine (SVM)” en la predicción de las preferencias de existencias de acciones, con un día de anticipación, del mercado de valores en Mumbai (BSE-Sensex). La solución a este tipo de problemas, por lo general, se lo plantea como una regresión, pero la técnica SVM lo aborda como una clasificación y con esto incrementa la precisión de la predicción [47].
- Otro ejemplo es el uso de la técnica de “Árboles de Decisión” en la definición de las estrategias de marketing que utilizó la compañía de seguros de mascotas “PetCoverCo” para ingresar al mercado estadounidense, sugiriendo las áreas y los datos

demográficos en donde era mejor implementarlas, focalizándose en los clientes objetivos para así maximizar la rentabilidad y el impacto de las campañas [48].

- Otro ejemplo es el uso de la técnica de “Reglas de Asociación” en el cambio constante de las promociones de artículos y sus ubicaciones físicas, en los diferentes almacenes de la cadena mundial Walmart [49].

Las técnicas de “Árboles de Decisión” y “Reglas de Asociación” permiten identificar de manera granular los grupos de productos con mayor frecuencia de compra, para así maximizar la rentabilidad y generar ventaja competitiva. Se presentará información más detallada sobre estas 2 técnicas en las siguientes secciones de este capítulo.

2.1.5. Canasta de Compra

La “Canasta de compra” se define como el elemento - por cliente o grupo - que contiene los productos comprados con mayor frecuencia y sus cantidades por cada tipo de artículo. Es un elemento muy importante para encontrar patrones escondidos, no comunes, y de importante valor para las empresas.

La implementación de la “Canasta de compra” permite predecir futuras tendencias y comportamientos de clientes al comprar artículos. Las empresas pueden desarrollar ofertas hacia sus clientes y captar nuevos. De esta manera las empresas están toman decisiones basadas en Data Mining [21].

Por ejemplo, en un estudio realizado a un grupo de retail en el Reino Unido - Tesco, ASDA, Sainsbury's and Morrison's – se utilizaron técnicas de "Reglas de asociación" para realizar el análisis de la "Canasta de compra" y así identificar las mejores combinaciones de artículos [42].

Otras técnicas de Data Mining que nos ayudan en la implementación eficiente de la "Canasta de compra", son "Support Vector Machine (SVM)" y las "Redes Neuronales".

La técnica de "Support Vector Machine (SVM)" permite agrupar a las "Reglas de Asociación" para luego obtener indicadores de más alto nivel como, por ejemplo: la cantidad de clientes impactados por el combo de la "Canasta de compra" [29].

La técnica de "Redes Neuronales", permite establecer relaciones entre "Canastas de compras", para luego realizar un análisis de interdependencia a nivel de las categorías de producto.

- Esto tiene especial importancia para decidir en dónde colocar los productos en la percha y para establecer el precio de las promociones [30].
- Otro complemento que ofrece la técnica de "Redes Neuronales" es actualizar, de manera más eficiente, el listado de reglas de asociación de mayor importancia en base a indicadores no propios de la transacción, por ejemplo, el "Puntaje de la regla

basado en el % de ganancia de la venta cruzada de los artículos" [31].

En la siguiente sección nos enfocaremos en la técnica de "Reglas de asociación" y sus métricas de evaluación, las cuales nos permiten escoger las mejores reglas generadas.

2.2. Reglas de Asociación

Las "Reglas de asociación" relacionan una determinada conclusión (Consecuente) con un conjunto de condiciones (Antecedente).

Es una expresión de la forma $X \rightarrow Y$, donde $X \wedge Y$ son conjuntos de elementos disjuntos, es decir, $X \cap Y = \emptyset$.

Un ejemplo para explicar las "Reglas de Asociación", en una empresa supermercados, es [22]:

- Vegetales y Congelados → Cerveza
    - Antecedente:
        - Condición 1: Vegetales
        - Condición 2: Congelados
    - Consecuente: Cerveza

Esta regla expresa que, los clientes que compran Vegetales y Congelados posiblemente compren Cerveza, es decir que cuando ocurre la venta de Vegetales y Congelados, también puede ocurrir la venta de Cerveza. Cabe destacar que no son iguales las reglas *"Vegetales y Congelados → Cerveza"* con *"Cerveza → Vegetales y Congelados"*, es decir que en las

“Reglas de asociación” no se aplica la ley de “Bicondicional”. También podrían existir otros consecuentes distintos a Cerveza - en el listado de reglas generadas por algún algoritmo - pero queda a elección del usuario seleccionar aquella regla que en la ejecución retorne más beneficios para la empresa.

Por lo general, las “Reglas de asociación” son utilizadas en los Retail para establecer la “Canasta de compra”. Por ejemplo, en un Retail de electrodomésticos de Perú, se descubrió importantes reglas entre las categorías de productos que determinan el comportamiento de los clientes y así conocer la probabilidad que un determinado cliente compre un producto categorizado como Y dado que anteriormente compro un producto categorizado como X [35].

Una “Regla de asociación” emplea diferentes mediciones, tales como Soporte, Confianza, Lift y Conviction. Las más utilizadas son Soporte y Confianza [10].

2.2.1. Métricas de Evaluación

Las métricas de evaluación nos permiten valorar las “Reglas de asociación” generadas por los modelos predictivos y escoger las que sean más seguras para ejecutarlas como estrategia del negocio.

En la siguiente sección se especifica la forma de obtener el Soporte, Confianza, Lift y Conviction.

2.2.1.1. Soporte

El soporte determina con qué frecuencia una regla está presente en el conjunto de datos [10].

$$Soporte, S(x \rightarrow y) = \frac{x\ U\ y}{N}$$

Donde:

- x: número de transacciones que contienen el antecedente
- y: número de transacciones que contienen el consecuente
- x U y: número de transacciones que contienen las condiciones del antecedente y las del consecuente
- N: total de transacciones del conjunto de datos

El soporte permite elegir únicamente aquellas reglas que satisfacen un nivel mínimo de soporte, el cual es establecido en las configuraciones del modelo predictivo.

Usando el ejemplo anterior:

- Vegetales y Congelados → Cerveza
- N: Total de transacciones del conjunto de datos (1018 transacciones)
- x U y: Cantidad transacciones que tienen vegetales y congelados y también cerveza (173 transacciones)

- $Soporte, S(x \rightarrow y) = \frac{173}{1018} = 17\%$
    - En el 17% de las transacciones del conjunto de datos se encuentra presente la regla: “Vegetales y Congelados → Cerveza”.

2.2.1.2. Confianza

La confianza determina con qué frecuencia los elementos en Y aparecen en transacciones que contienen X, donde X y Y, son conjuntos de elementos disjuntos, es decir, X ∩ Y = ∅ [10].

$$Confianza, C(x \rightarrow y) = \frac{x\ U\ y}{X}$$

Donde:

- x: número de transacciones que contienen el antecedente
- y: número de transacciones que contienen el consecuente
- x U y: número de transacciones que contienen las condiciones del antecedente y consecuente
- X: número de transacciones que contienen solamente las condiciones del antecedente

La confianza permite elegir aquellas reglas que satisfacen un mínimo valor de probabilidad, el cual es establecido en las configuraciones del modelo predictivo.

Usando el ejemplo anterior:

- Vegetales y Congelados → Cerveza
- x U y: Cantidad de transacciones que tienen vegetales y congelados y también cerveza (173 transacciones)
- X: Cantidad de transacciones que tienen solamente las condiciones del antecedente (206 transacciones)
- $Confianza, C(x \rightarrow y) = \frac{173}{206} = 84\%$
    - En el 84% de las transacciones en donde se encuentran en el antecedente los productos “Vegetales y Congelados”, también se encuentra presente en el consecuente el producto “Cerveza”.

2.2.1.3. Lift

Lift ayuda a diferenciar si una regla es importante o no. La confianza tiene el problema de que se calcula sobre el

subconjunto de datos implicados en la regla - no toma en cuenta el total del conjunto de datos.

También se define la métrica Lift como el aumento de probabilidad de que ocurra el consecuente cuando ocurre el antecedente [17]:

$$Fórmula\ 1)\ Lift(x \rightarrow y) = Lift\ (y \rightarrow x) = \frac{C(x \rightarrow y)}{S(y)}$$

$$Fórmula\ 2)\ Lift(x \rightarrow y) = Lift\ (y \rightarrow x) = \frac{P(x|y)}{P(y)}$$

$$Fórmula\ 3)\ Lift(x \rightarrow y) = Lift\ (y \rightarrow x) = \frac{P(x \cap y)}{P(x)P(y)}$$

Donde:

- Fórmula 1:
  - C (x → y): Confianza de X y Y
  - S (y): Soporte de Y
- Fórmula 2 y 3:
  - P (x | y): Probabilidad o Fracción o Relación entre las transacciones en donde aparece el consecuente y las transacciones en donde aparece el antecedente

- P(x): Probabilidad o Fracción o Relación entre las transacciones del antecedente y el total de transacciones del conjunto de datos.
- P(y): Probabilidad o Fracción o Relación entre las transacciones del consecuente y el total de transacciones del conjunto de datos.
- P (x ∩ y): Probabilidad o Fracción o Relación entre las transacciones que contienen solamente las condiciones del antecedente junto a las del consecuente y el total de transacciones del conjunto de datos

- El valor de Lift nos indica:
  - Cuando Lift(X→Y) > 1: que el antecedente y el consecuente de la regla se encuentran usualmente más veces de lo que esperado
  - Cuando Lift(X→Y) < 1: que el antecedente y el consecuente de la regla se encuentran usualmente menos veces de lo que esperado

Usando el ejemplo anterior:

- Vegetales y Congelados → Cerveza
- C (x → y): 84%
- S (y): 190 transacciones tienen cerveza de un total de 1018 transacciones

- $Lift(x \rightarrow y) = Lift\ (y \rightarrow x) = \frac{C(x \rightarrow y)}{S(y)} = \frac{0.84}{\frac{190}{1018}} = 4.5$
    - La posibilidad de que en una regla aparezca el producto “Cerveza” está estrechamente relacionada cuando aparecen los productos “Vegetales y Congelados”.

2.2.1.4. Conviction

Dada la regla X→Y, la convicción compara la probabilidad de que aparezca x sin y. Asumiendo la independencia de ambos.

$$Fórmula\ 1)\ Conv(x \rightarrow y) = \frac{1 - S(y)}{1 - C(x \rightarrow y)}$$

$$Fórmula\ 2)\ Conv(x \rightarrow y) = \frac{P(x)P(\neg y)}{P(x, \neg y)}$$

Donde:

- Fórmula 1:
    - C (x → y): Confianza de X y Y
    - S (y): Soporte de Y
- Fórmula 2:
    - P(x): Probabilidad o Fracción o Relación entre las transacciones del antecedente y el total de transacciones del conjunto de datos.

- P(¬y): Probabilidad o Fracción o Relación entre las transacciones que no contienen el consecuente y el total de transacciones del conjunto de datos.
- P(x,¬y): Probabilidad o Fracción o Relación entre las transacciones que contienen el antecedente, pero no contienen el consecuente, y el total de transacciones del conjunto de datos.

- Conv(X→Y) no es igual a Conv(Y→X).
- Cuando Conv(X→Y) < 1: negativa dependencia
- Cuando Conv(X→Y) = 1: independencia
- Cuando Conv(X→Y) > 1: positiva dependencia
  - Altos valores para Conv (X→Y) afirman que la regla representa una causalidad [20].

Usando el ejemplo anterior:

- Vegetales y Congelados → Cerveza
- C (x → y): 84%
- S (y): 190 transacciones tienen cerveza de un total de 1018 transacciones
- $Conv(x \rightarrow y) = \frac{1-S(y)}{1-C(x \rightarrow y)} = \frac{1-\frac{190}{1018}}{1-0.84} = 0.16$

- La mayoría de las veces que en una regla aparezcan los productos “Vegetales y Congelados”, también encontraremos el producto “Cerveza”.

2.3. Herramientas utilizadas en la elaboración de los modelos

Las herramientas open source Weka-JAVA y Python permiten la generación de las reglas de asociación. Estas herramientas poseen las siguientes características:

- Los algoritmos, librerías (APIs) y entornos de trabajo (IDE) llevan mucho tiempo en el mercado y han sido utilizados en proyectos de distinta índole (académicos, empresariales, entre otros).
- La mayoría tienen alguna versión de uso sin costo.
- Pueden ser utilizadas directamente por los usuarios de negocios debido a que algunas herramientas se usan en base a configuraciones.
- Existen empresas como “Google” que las integran en sus soluciones de la nube para incentivar el desarrollo de proyectos de Data Mining.

2.3.1. Lenguajes de Programación Utilizados

2.3.1.1. Java y el Data Mining

Java es un lenguaje de programación de propósito general orientado a objetos. Todo programa ejecutado en java debe ser

compilado, y el código generado (bytecode) debe ser interpretado por una máquina virtual [13].

El estándar de Java para el desarrollo de aplicaciones de Data Mining basado en la especificación JSR-73, proporciona una forma estándar de crear, almacenar, acceder y mantener los datos, además de los metadatos compatibles con modelos de Data Mining [27].

Históricamente, los desarrolladores de aplicaciones codificaron algoritmos de Data Mining y los empaquetaron en librerías (APIs). Sin embargo, es difícil incrustar Data Mining en aplicaciones comerciales porque la solución es patentada. Esto dificulta a los usuarios la selección de un producto en particular [28].

- La capacidad de aprovechar la funcionalidad del Data Mining a través de una API estándar, reduce en gran medida el riesgo y el costo. Una API estándar aumenta la accesibilidad al Data Mining y permite su uso entre diferentes fabricantes de software.
- Una API estándar permite a las empresas aprovechar las fortalezas de múltiples proveedores de Data Mining para así resolver problemas comerciales mediante el uso del algoritmo más apropiado.

2.3.1.2. Python y el Data Mining

Es un lenguaje de programación interpretado que posee muchas virtudes y ventajas, así como también numerosas aplicaciones prácticas que permiten implementar algoritmos, dispone de una gramática excepcional que responde de manera natural a muchas necesidades clásicas [14].

2.3.1.3. Pandas

Pandas es una biblioteca de código abierto con licencia BSD (Berkeley Software Distribution). Permite su uso en código fuente de software no libre. Además, proporciona estructuras de datos y herramientas de análisis de datos de alto rendimiento y fáciles de usar, sobre el lenguaje de programación Python [19].

2.3.2. Entornos de Trabajos

2.3.2.1. Weka

Weka (Waikato Environment for Knowledge Analysis), es una plataforma de software para el aprendizaje automático y la minería de datos escrito en Java y desarrollado por la Universidad de Waikato de Nueva Zelanda. Weka es un software libre bajo licencia GNU-GPL [11].

2.3.2.2. Anaconda

Anaconda es un entorno de trabajo, para los lenguajes Python y R, de distribución libre. Su principal uso es en la ciencia de datos y el aprendizaje automático. Esto incluye procesamiento de grandes volúmenes de información, análisis predictivo y cómputos científicos. Está orientado a simplificar el despliegue de los paquetes de software [15].

2.3.2.3. Jupyter Notebook

Jupyter Notebook (anteriormente IPython Notebooks) es un entorno de trabajo basado en la web. El término "notebook" hace referencia a diferentes entidades, principalmente la aplicación web Jupyter, el servidor web Jupyter Python o el formato de documento Jupyter según el contexto. Un documento de Jupyter Notebook es un documento JSON, que sigue un esquema versionado y que contiene una lista ordenada de celdas de entrada/salida que contienen código, texto, matemáticas, gráficos y texto enriquecidos [16].

2.3.3. Algoritmos de Reglas de Asociación

Por lo general, los algoritmos de APRIORI y Fp-Growth son utilizados para encontrar reglas de asociación. Destacan su rendimiento y facilidad de uso en diferentes plataformas (Weka, Rapid Miner, Python, R, entre otras), además de su eficacia en la generación de

reglas [50]. Por lo tanto, son los seleccionados para desarrollar los modelos predictivos.

2.3.3.1. APRIORI

El algoritmo Apriori está diseñado para la obtención de “Reglas de asociación” a partir de bases o conjuntos de datos transaccionales. Usa la propiedad de “Clausura Descendente”, la cual define que todo el subconjunto de datos es frecuente si y sólo si, el conjunto de datos del padre también lo es.

Este algoritmo genera muchos candidatos y el proceso de conteo de la frecuencia es costoso [12]. El proceso de generación de reglas es el siguiente [21]:

- Se accede directamente a la base de datos para encontrar todos los conjuntos unitarios frecuentes. De ahí se mezclan entre si para generar los conjuntos candidatos – de varios elementos – y luego se eligen los más frecuentes.
- Se mezclan los conjuntos candidatos – de varios elementos – hasta generar nuevos conjuntos frecuentes y esto se repite de forma cíclica hasta que no se obtengan más conjuntos frecuentes.

En la siguiente figura se muestra un ejemplo del funcionamiento del algoritmo APRIORI. En la “Fase 1” las transacciones se representan con unos y ceros, siendo “1”

cuando se encuentra presente el atributo en la transacción y "0" cuando no se encuentra presente el atributo en la transacción. En la "Fase 2" se realiza un ciclo de iteraciones. En cada iteración se forman itemset de diferente tamaño y se obtienen aquellos que se encuentran por encima del mínimo valor de soporte. En la "Fase 3" se obtienen solo aquellos que se encuentren por encima del mínimo valor de confianza y en la "Fase 4" se muestran los itemset que cumplieron con los mínimos valores de soporte y confianza

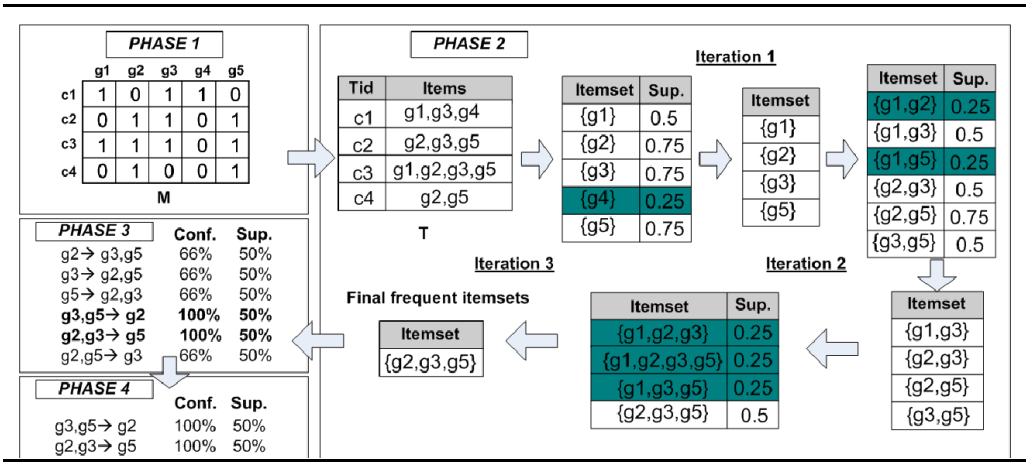


Figura 2.3 - Ejemplo del Algoritmo APRIORI [54]

2.3.3.2. FP-Growth

El algoritmo FP-Growth obtiene un grupo completo de asociaciones frecuentes de un conjunto de datos, evitando tener que realizar una costosa fase de generación de candidatos, para ello hace uso de una estructura llamada FP-tree que contiene la lista de asociaciones frecuentes. Esta lista

es de longitud “1”, es decir que cada nodo de la lista tiene un solo item dentro de su estructura de datos [12].

- FP-Tree (árbol de patrones frecuentes), es una estructura de datos en donde cada nodo es un conjunto de elementos que almacena la información del item, de las transacciones en donde se encuentra el item, la frecuencia del item y el enlace al puntero del siguiente nodo que tiene relación con el mismo elemento [26].
    - El campo principal contiene un puntero al nodo padre (nulo para la raíz).

Este algoritmo se encuentra implementado en el lenguaje de programación Python [18].

2.3.4. Otros Algoritmos de Reglas de Asociación

Los algoritmos PAFI y ECLAT no se los utilizaran en el presente estudio porque necesitan ser ejecutados en un ambiente de mejores capacidades tecnológicas a las disponibles. Tanto APRIORI y FP-Growth se ejecutaron en un equipo de cómputo Core i7 de 8 GB de RAM.

2.3.4.1. Algoritmo PAFI

El algoritmo “Partition Algorithm Frequent Items - PAFI”, requiere que todas las transacciones estén en la memoria y por

eso recorre la base de datos solamente 2 veces y la divide en tantas partes como que fuese necesario. En el primer recorrido las particiones se llenan de los conjuntos de ítems más frecuentes y luego se mezclan entre sí estos conjuntos para generar todos los conjuntos de ítems candidatos [21].

La desventaja de este algoritmo y por lo cual no se lo utilizo, es que requiere mucha memoria para almacenar por cada regla candidata la lista de las transacciones en donde aparece. Debido a esto, se debe usar este algoritmo junto a técnicas de agrupamiento [51].

2.3.4.2. Algoritmo ECLAT

El algoritmo “Equivalence Class Clustering and bottom-up Lattice Traversal - ECLAT”, reduce la cantidad de operaciones de E/S porque escanea la base de datos solo una vez. Funciona mediante el agrupamiento de ítems, para así aproximarse al conjunto de ítems de máxima frecuencia, luego selecciona los ítems más frecuentes contenidos en cada grupo [21].

La desventaja de este algoritmo y por lo cual no se lo utilizo, es que la cantidad de reglas generadas puede ser mayor que el conjunto de datos original y así se satura la memoria e impide que el algoritmo termine su ejecución [52].

2.4. Fases en la implementación de los modelos

Durante el desarrollo del proyecto, las siguientes metodologías fueron analizadas para identificar las fases que se requieren en la implementación de un proyecto de Data Mining con “Reglas de asociación”.

2.4.1. Principales Metodologías en proyectos de Data Mining.

SEMMA [6]: el instituto de SAS definió un ciclo con 5 etapas: Sample, Explore, Modify, Model, Assess.

5A’s [7]: tiene 5 etapas: Assess, Access, Analyze, Act, Automate.

Two Crows [8]: se basa en la implementación en cascada y tiene 7 etapas : Definición del problema de negocio, Construcción de la base de datos, Exploración de los datos, Preparación de los datos, Construcción del modelo, Evaluación del modelo, Ejecución y Monitoreo de los resultados.

CRISP-DM [24]: tiene 6 etapas: Entendimiento del Negocio [25], Entendimiento de los Datos, Preparación de los Datos, Modelamiento, Evaluación, Implantación.

En la siguiente Figura visualizamos las diferentes metodologías que se usan en proyectos de Data Mining, destacándose CRISP-DM por ser una recopilación de otras – SEMMA, 5A’s, Two Crows - y por tener una última fecha de publicación más reciente.

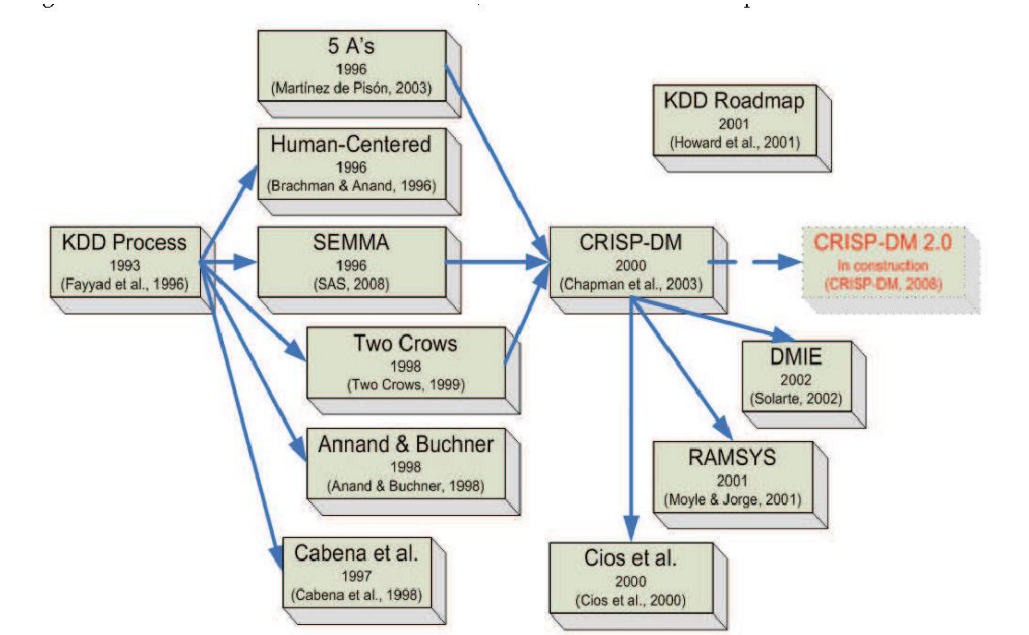


Figura 2.4 - Evolución de las Metodologías DM [9]

2.4.2. Definición de los Pasos en el desarrollo del modelo predictivo

Se identificaron las siguientes fases para la implementación del modelo:

- Compresión de los datos:
  - Lectura del conjunto de datos original
  - Selección de un subconjunto de datos acorde a los objetivos planteados
- Preparación de los datos:
  - Limpieza del subconjunto de datos
  - Modificaciones al subconjunto de datos para que sea utilizado por los algoritmos APRIORI y FP-Growth
- Construcción de los modelos

        - Implementación del modelo en APRIORI, usando la herramienta Weka – Java.
        - Implementación del modelo en FP-Growth, usando en lenguaje de programación Python.
    - Interpretación de Resultados
        - Análisis de reglas generadas
        - Selección del Modelo predictivo

En el siguiente capítulo nos enfocaremos en comprender la base de datos utilizada para el estudio, definiremos el conjunto de datos a utilizar por los modelos predictivos que luego se evaluaran.

CAPÍTULO 3

ANÁLISIS, DISEÑO E IMPLEMENTACIÓN

El objetivo del capítulo es mostrar el desarrollo de los modelos predictivos mediante la secuencia de fases y tareas utilizadas en la implementación. Para la definición de las fases y tareas se tomó en cuenta el conjunto de pasos definidos en el Capítulo 2.

Ejecución de la Fases

3.1 Comprensión de los Datos

3.1.1 Datos Originales

Los datos utilizados son de un repositorio libre y le pertenecen a un E-commerce de Brasil que unifica a diferentes tiendas minoristas.

El conjunto de datos tiene alrededor de 100,000 órdenes de compra desde los años 2016 al 2018, con 8 tablas y 52 campos. En la siguiente Figura visualizamos el modelo relacional de base de datos.

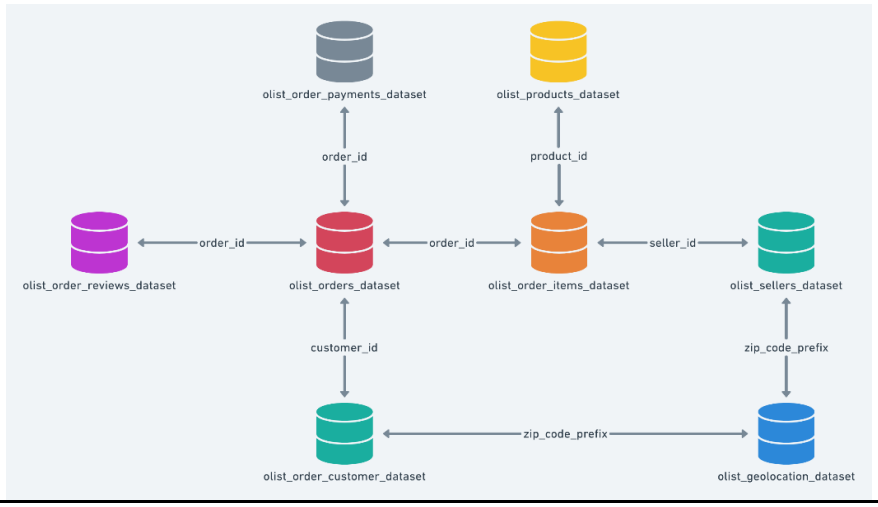


Figura 3.1 - Diagrama de Modelo Relacional de los Datos [58]

3.1.2 Especificación de los Datos

Cada tabla o instancia se describe de la siguiente manera:

- *olist_product_dataset*: Tabla de los datos asociados a los productos.
- *olist_order_payments_dataset*: Tabla de los datos asociados a las transacciones de pagos.
- *olist_sellerts_dataset*: Tabla de los datos asociados a los vendedores.

- *olist_order_custumer_dataset*: Tabla de datos asociados a órdenes de venta y datos de clientes.
- *olist_geolocation_dataset*: Tabla de los datos asociados a localización y ubicación de los vendedores.
- *olist_orders_dataset*: Tabla de datos asociados a las órdenes de ventas.
- *olist_order_items_dataset*: Tabla de datos asociados a los detalles o ítems de las órdenes de venta.
- *olist_order_reviews_dataset*: Tabla de datos asociados a las órdenes de ventas que se están revisando para su despacho.

3.1.3 Exploración de los Datos

Posterior a realizar el proceso de examinación de las tablas iniciales, listadas anteriormente, escogemos aquellos campos - *categorías de clientes, ubicaciones geográficas de clientes, categorías de artículos y formas de pago* - que nos permiten obtener "Combos de artículos" muy parecidos a los utilizados en las estrategias habituales de marketing - por ciudad, por género, por forma de pago, entre otros. Se eligieron los campos de las categorías de producto, la forma de pago y el Estado-Ciudad de los clientes y vendedores. El resto de los campos, en su mayoría, eran campos técnicos y con poca importancia para nuestro estudio. El

nuevo conjunto de datos tiene alrededor de 116 mil transacciones y 11 campos:

Tabla 1 - Exploración de Datos

Campo – Atributo	**Descripción**	**Observaciones**
order_delivered_customer_date	Fecha de Entrega	Las órdenes van desde Octubre 2016 a Septiembre 2018
customer_city	Ciudad del Cliente	Los clientes son de 4095 ciudades
customer_state	Estado del Cliente	Los clientes son de 27 estados
seller_city	Ciudad del Vendedor	Los vendedores son de 694 ciudades
seller_state	Estado del Vendedor	Los vendedores son de 23 estados
product_category_name	Nombre de la Categoría de Artículo	Las categorías son 71
product_category_name_english	Nombre en Inglés de la Categoría de Artículo	Las categorías son 71
price	Precio	Utilizados para definir si tuvo venta
freight_value	Valor del Flete	
payment_type	Tipo de Pago	Son 4 formas de pago: crédito, débito, voucher, boleto descuento
payment_value	Valor de Pago	

3.2 Preparación de los Datos

Con la finalidad de tener un nuevo conjunto de datos sin mucha dispersión de valores y con transacciones del tipo boolean para las

columnas de categorías de artículos, al conjunto de datos resultante de la tarea “Exploración Datos”, se le realizaron las siguientes acciones:

- Eliminación de campos:
  - Por tener valores muy dispersos
    - “customer_city” y “seller_city”
  - Otro campo tiene un valor similar
    - “product_category_name”
  - Utilizados para definir el nuevo campo “monto”
    - “price”, “freight_value” “payment_value”
- Modificación de campos:
  - Se cambió su valor de fecha diaria a fecha mensual (YYYY-MM)
  - El nuevo campo “monto” cambió de valor decimal a boolean
  - A los siguientes campos se les disminuye la dispersión, tomando en cuenta solamente los 5 valores más frecuentes en el 90% de las órdenes y a los valores restantes se les coloca un valor por defecto “Otros_SS” y “Otros_CS” respectivamente:
    - “seller_state”, “customer_state”
  - Al campo “product_category_name_english”

- Se le disminuye la dispersión, tomando en cuenta solamente las 10 categorías de productos que representan alrededor del 80% de la venta
- Se transpone este atributo de forma vertical a horizontal

Luego de aplicar todas las acciones antes mencionados, se origina un nuevo conjunto de datos con 2194 transacciones y 14 columnas

Tabla 2 - Preparación de Datos

Columna – Atributo	**Acción Realizada**
periodo	Texto
seller_state	Texto
customer_state	Texto
payment_type	Texto
Las 10 Categorías de artículos (de 71) que representan el 80% de la venta	
auto	Boolean
bed_bath_table	Boolean
computers_accessories	Boolean
cool_stuff	Boolean
furniture_decor	Boolean
garden_tools	Boolean
health_beauty	Boolean
housewares	Boolean
sports_leisure	Boolean
watches_gifts	Boolean

3.3 Construcción de los Modelos

En esta sección se describe la forma de utilizar los algoritmos de Apriori implementado en JAVA y Fp-Growth implementado en Python.

3.3.1 Modelo APRIORI JAVA

Se utilizará el algoritmo APRIORI que se encuentra en la herramienta WEKA. A continuación, mostraremos los 5 pasos - *carga del conjunto de datos, eliminación de atributos no relevantes, selección del algoritmo APRIORI, configuración de los parámetros de entrada del algoritmo APRIORI, Ejecución del algoritmo e interpretación de resultados* - que se realizaron para obtener los resultados:

a) Abrimos la herramienta WEKA y realizamos la carga del conjunto de datos que se encuentra en formato de columnas separadas por ";".
b) Una vez cargados los datos sobre WEKA, la plataforma realiza una estructuración por columnas y establece varios parámetros estadísticos sobre los atributos, tales como: cantidad de valores únicos, frecuencia por valor único, peso por valor y tipo de dato
    - Del conjunto de datos resultante en la tarea "Preparación de los datos", solo eliminamos el

atributo “periodo” porque su nivel de detalle es usado como contador de frecuencia de las reglas.

Los atributos resultantes en Weka son:

Tabla 3 - Paso b) del modelo APRIORI - Métricas por Atributos

Atributos	
No.	**Nombre**
1	***seller_state***
2	*customer_state*
3	*payment_type*
4	Auto
5	bed_bath_table
6	computers_accesories
7	cool_stuff
8	furniture_decor
9	garden_tools
10	health_beauthy
11	housewares
12	sport_leisure
13	watch_gifts
Métricas de Atributo	

Name	seller_state	6 Valores	Tipo de Dato “Nominal”
No.	**Etiqueta**	**Conteo**	**Peso**
1	MG	380	380
2	PR	374	374
3	RJ	285	295
4	SC	312	312
5	SP	502	502
6	OTROS_SS	331	331

En la Tabla anterior se muestran los campos utilizados por APRIORI. Se puede visualizar por cada campo los valores que lo conforman, y los indicadores de conteo y peso. Si el atributo fuera numérico se visualizan los indicadores de “mínimo, máximo, media y desviación estándar”

c) Ahora procedemos a seleccionar en WEKA la opción “Associate” para escoger y ejecutar el algoritmo de Asociación APRIORI.

d) El algoritmo por defecto es APRIORI. Establecemos las siguientes configuraciones de entrada como, por ejemplo: el número de reglas a generar, la confianza mínima, el mínimo nivel de soporte y el tipo de métrica a utilizar para agrupar las reglas.

Tabla 4 - Paso d) del modelo APRIORI - Parámetros de Entrada Algoritmo

APRIORI	
Número de Reglas de Salida	10

Tipo de Métrica (Confianza)	Confianza
Valor Mínimo de la Métrica (Confianza)	0.9
Límite Inferior para Soporte Mínimo	0.1

e) Una vez que se ejecuta el algoritmo se muestran los resultados. A continuación, presentamos una tabla con la información arrojada por APRIORI.

Tabla 5 - Paso e) del modelo APRIORI - Resultados

Instancias	**2194**
Atributos	**13**
	seller_state
	customer_state
	payment_type
	auto
	bed_bath_table
	computers_accessories
	cool_stuff
	furniture_decor
	garden_tools
	health_beauty
	housewares

	sports_leisure
	watches_gifts
Apriori	
Mínimo soporte	**0.1**
Confianza Mínima	**0.9**
Número de Ciclos	**16**
Mejores Reglas Encontradas	
1. auto=TRUE computers_accessories=TRUE housewares=TRUE sports_leisure=TRUE 459 ==> health_beauty=TRUE 439 <conf:(0.96)> lift:(1.75) lev:(0.09) [188] conv:(9.93)	
2. auto=TRUE bed_bath_table=TRUE health_beauty=TRUE 470 ==> sports_leisure=TRUE 444 <conf:(0.94)> lift:(1.66) lev:(0.08) [177] conv:(7.52)	
3. computers_accessories=TRUE sports_leisure=TRUE watches_gifts=TRUE 475 ==> health_beauty=TRUE 448 <conf:(0.94)> lift:(1.73) lev:(0.09) [188] conv:(7.71)	
4. computers_accessories=TRUE health_beauty=TRUE watches_gifts=TRUE 476 ==> sports_leisure=TRUE 448 <conf:(0.94)> lift:(1.66) lev:(0.08) [177] conv:(7.09)	
5. auto=TRUE computers_accessories=TRUE sports_leisure=TRUE 523 ==> health_beauty=TRUE 492 <conf:(0.94)> lift:(1.72) lev:(0.09) [206] conv:(7.43)	
6. auto=TRUE computers_accessories=TRUE health_beauty=TRUE	

housewares=TRUE 467 ==> sports_leisure=TRUE 439 <conf:(0.94)> lift:(1.66) lev:(0.08) [173] conv:(6.96)
7. computers_accessories=TRUE furniture_decor=TRUE health_beauty=TRUE housewares=TRUE 512 ==> sports_leisure=TRUE 480 <conf:(0.94)> lift:(1.65) lev:(0.09) [189] conv:(6.7)
8. auto=TRUE furniture_decor=TRUE health_beauty=TRUE 494 ==> sports_leisure=TRUE 463 <conf:(0.94)> lift:(1.65) lev:(0.08) [182] conv:(6.67)
9. health_beauty=TRUE housewares=TRUE watches_gifts=TRUE 470 ==> sports_leisure=TRUE 440 <conf:(0.94)> lift:(1.65) lev:(0.08) [173] conv:(6.55)
10. cool_stuff=TRUE furniture_decor=TRUE health_beauty=TRUE 484 ==> sports_leisure=TRUE 453 <conf:(0.94)> lift:(1.65) lev:(0.08) [178] conv:(6.54)

En los resultados de la Tabla anterior, se pueden visualizar las 10 reglas que tienen como mínimo el valor de soporte y de confianza configurados en los parámetros de entrada. Además, para encontrarlas se realizaron 16 interacciones sobre el conjunto de datos. Las reglas tienen de 3 a 4 artículos en su antecedente y solo un artículo en el consecuente. Cada regla muestra sus indicadores de evaluación (Confianza, Lift y Conviction).

Por ejemplo, la regla número 10 tiene una frecuencia de 453 transacciones (20.64% de Soporte), una Confianza del 94%, con un

Lift y Conviction positivo que nos indica que existe una relación entre la compra de los artículos “cool_stuff, furniture_decor y Health_beauty” del antecedente y la compra de los artículos “sports_leisure” del consecuente.

Con los parámetros de entrada definidos en la “Tabla 4: confianza = 0.90 y mínimo soporte = 0.1”, logramos que el modelo asociativo de APRIORI genere reglas – aunque el mínimo soporte sea bien bajo -, esto se debe a que no existen patrones de transacciones con altos valores de frecuencia en el conjunto de datos original (E-commerce de Brasil).

3.3.2 Modelo FP-Growth PYTHON

Se utilizará el algoritmo “FP-Growth” implementado en Python. Usaremos la herramienta “Jupyter Notebook” como entorno de desarrollo. A continuación, mostraremos los 9 pasos – *carga del conjunto de datos y configuración de los parámetros de entrada del modelo, cambió de los nombres de las cabeceras de artículos, cambió de los valores de los campos de artículos, encontrar los patrones de transacciones frecuentes, eliminación de patrones con transacciones que tengan un artículo sin relacionarse con los otros, generación de las reglas de asociación, eliminación de las reglas repetidas, división por columnas de los elementos de las reglas generadas, visualización de las 10 mejores reglas en soporte y confianza* - que se realizaron para obtener los resultados:

a) Cargar el conjunto de datos a partir de un archivo en formato '.csv' y establecer los siguientes parámetros de entrada:

Tabla 6 - Paso a) Python FP-GROWTH - Parámetros de Entrada

Ruta de Archivo	/data/EcommerceBrasil_DS_v9A.csv
Cantidad de meses mínimos	16
Valor Mínimo de Soporte	0.1
Valor Mínimo de Confianza	0.9
Cantidad Máxima de Antecedentes	4
Cantidad Máxima de Consecuentes	1

b) Cambiar los nombres de las columnas cabeceras, de las categorías de artículo, por códigos numéricos para mejorar la eficiencia y los tiempos de ejecución del algoritmo.

Código Python implementado:

```
## Permite cambiar los nombres de las columnas cabeceras
def create_dict_new_columns(list_columns):
    new_list_columns = {}
    list_index_columns = []
    ## Recorre la lista de columnas del dataframe pandas
    for obj in list_columns:
        index = list_columns.index(obj)
        if index > (CANTIDAD_DIMENSIONES_NO_SKU - 1):
            list_index_columns.append(str(index*1000))
        else:
            list_index_columns.append(obj)

    d_columns = {'Columnas_ORI':list_columns,
    'Columnas_TMP':list_index_columns}
    df_columns = pd.DataFrame(data=d_columns)
    return df_columns

    df.columns = df_new_list_columns['Columnas_TMP'].values.tolist()
```

Después de ejecutar la función “create_dict_new_columns”, visualizamos el cambio de nombre de las columnas cabeceras

Tabla 7 - Paso b) Python FP-GROWTH – Resultados de Categorías por Código

Seller_ state	Custumer_ state	Payment_t ype	3000	4000	5000	6000	7000	8000	9000	1000
MG	MG	boleto	True	True	True	True	True	True	True	True
MG	MG	credit_card	True	True	True	True	True	True	True	True
MG	MG	debit_card	False	False	True	True	False	False	True	True
MG	MG	voucher	True	True	True	True	True	True	True	True
MG	PR	boleto	True	True	True	True	True	True	True	True

c) Cambiar los valores de las categorías de artículo de boolean a texto con el siguiente formato para el valor: “Boolean_Codigo”.

Código Python implementado:

```
##Transformacion de dimensiones a Texto
df[DIMENSION_1] = df[DIMENSION_1].astype(str)
df[DIMENSION_2] = df[DIMENSION_2].astype(str)
df[DIMENSION_3] = df[DIMENSION_3].astype(str)

def concatenar_prefijo_categoria(categoria):
    return {True:'True_'+categoria,False:'False_'+categoria }

## Recorre los atributos del dataframe que sean del tipo boolean
booleandf = df.select_dtypes(include=['bool'])
for column in booleandf:
    if column != DIMENSION_1
    and column != DIMENSION_2
    and column != DIMENSION_3:
        df[column] = df[column].map(concatenar_prefijo_categoria(column))
```

Después de ejecutar la función “concatenar_prefijo_categoria”, visualizamos los nuevos valores en las columnas de categorías artículo.

Tabla 8 - Paso c) Python FP-GROWTH – Resultados de Códigos Booleanos

Seller_state	Custumer_state	Payment_type	3000	4000	5000	6000	7000	8000
MG	MG	boleto	True_3000	True_4000	True_5000	True_6000	True_7000	True_8000
MG	MG	credit_card	True_3000	True_4000	True_5000	True_6000	True_7000	True_8000
MG	MG	debit_card	False_3000	False_4000	True_5000	True_6000	False_7000	False_8000
MG	MG	voucher	True_3000	True_4000	True_5000	True_6000	True_7000	True_8000
MG	PR	boleto	True_3000	True_4000	True_5000	True_6000	True_7000	True_8000

d) Encontrar los patrones de transacciones que cumplan con la mínima cantidad de meses repetidos en el conjunto de datos

Código Python implementado:

patterns = pyfpgrowth.find_frequent_patterns(transactions, CANTIDAD_MAX_REPETICIONES)

e) Eliminar los patrones que tengan algún valor por NO, para así mejorar los tiempos de ejecución del algoritmo que encuentra las reglas.

Código Python implementado:

```
## Remueve las transacciones en donde aparezca el "NO" en algún condicional
def remove_itemset_NO(patterns, num_itemset_iterator):
    new_patterns = {}
    i = 1
    ## Recorre la lista de transacciones frecuentes generadas
    for key in patterns.keys():
     if i <= num_itemset_iterator or num_itemset_iterator == 0:
        key_NO = list(filter(lambda x: "False_" in x, key))
        if len(key_NO) == 0:
            new_patterns[key] = patterns[key]
     else:
        break
     i = i + 1

  return new_patterns

new_patterns = remove_itemset_NO(patterns,0)
```

f) Ejecutar el método que encuentra las "Reglas de asociación".

- Al método original "generate_association_rules" se le realizaron los siguientes cambios:
  - Filtrado de las reglas generadas por cantidad mínima de condicionales en el antecedente y consecuente.
  - Se cambio el valor de los campos antecedente y consecuente de "Boolean_Codigo" a "Boolean_NombreCategoria"
  - Obtención de los siguientes campos en la estructura de salida:
    - antecedent: condiciones en el antecedente

- consequent: condiciones en el consecuente
- support: soporte de la regla
- confidence: confianza de la regla
- upper_support: soporte del antecedente
- original_dataframe: cantidad de registros del conjunto de datos

Código Python implementado:

```
## Permite obtener el nombre original de las categorías de productos
def GetName_ORI_Category(df, value):
    array_new = []
   ## Reemplaza los paréntesis y comillas dobles de cada transacción
    value_tmp = str(value)
    value_tmp = value_tmp.replace('(','')
    value_tmp = value_tmp.replace(',)','')
    value_tmp = value_tmp.replace(')','')
    value_tmp = value_tmp.replace('\'','')
    ## Separa la cadena de texto de la transacción en columnas
    array_value = value_tmp.split(',')
    for element in array_value:
        found = False
        value_new = ''
        for index, row in df.iterrows():
            text_find = '_' + row['Columnas_TMP']
            if (element.find(text_find) != -1):
                found = True
                text_new = '_' + row['Columnas_ORI']
                value_new = element.replace(text_find,text_new)
                break
        if found:
            array_new.append(value_new)
        else:
            array_new.append(element)

    value_return = tuple(array_new)
    return value_return

## Genera las reglas de asociación que cumplan con el mínimo de
## Soporte y Confianza
def generate_association_rules(original_dataframe
                               , patterns
                               , confidence_threshold
                               , mim_support
                               , array_cedi
                               , max_num_antecedent
```

```
                                , max_num_consequent
                            ):
    rules_lis=[]
    rules = {}

    print('Inicio generate_association_rules')
    now = datetime.datetime.now()
    print (now.strftime("%Y-%m-%d %H:%M:%S"))
    print(len(patterns))

    row_ori_df = len(original_dataframe)
    i_itemset = 0
    ## Recorre la lista de transacciones
    for itemset in patterns.keys():

        i_itemset = i_itemset + 1
        now = datetime.datetime.now()

        upper_support = patterns[itemset]

        ## Recorre por cada transacción las columnas
        ## de antecedente y consecuente
        for i in range(1, len(itemset)):

            ## Recorre cada condición del antecedente
            for antecedent in itertools.combinations(itemset, i):
                antecedent = tuple(sorted(antecedent))
                consequent = tuple(sorted(set(itemset) - set(antecedent)))
                ## Calcula las métricas por cada regla
                if antecedent in patterns:
                    lower_support = patterns[antecedent]
                    confidence = float(upper_support) / lower_support
                    antecedent_support= lower_support/row_ori_df
                    rule_support= float(upper_support)/row_ori_df
                    lift_rule = float(confidence) / ( upper_support / row_ori_df)
                    conviction_rule = 0
                    if ( 1 - float(confidence) ) > 0:
                        conviction_rule = float( 1 - ( upper_support / row_ori_df) )
                                          /  ( 1 - float(confidence) )
                    ## Valida que la regla cumpla con el mínimo valor
                    ## de soporte y confianza, además de la cantidad
                    ## de productos en el antecedente y consecuente
                    if confidence >= confidence_threshold
                        and rule_support >=mim_support
                        and len(antecedent)<=max_num_antecedent
                        and len(consequent)<=max_num_consequent:
                        rules_lis.append
                                    ({
                                    'antecedent': GetName_ORI_Category(
                                                    df_new_list_columns
                                                    ,antecedent)
                                    ,'consequent': GetName_ORI_Category(
                                                    df_new_list_columns
                                                    ,consequent)
                                    ,'support': rule_support
```

```
                    ,'support_antecedent':antecedent_support
                    ,'confidence':Confidence
                    ,'lift':lift_rule
                    ,'conviction': conviction_rule
                    ,'upper_support':upper_support
                    ,'original_dataframe':row_ori_df
                    } )


    ## Ejecuta el método de generación de reglas
    rules_df = generate_association_rules(
                    df
                    ,new_patterns
                    ,MIN_CONFIDENCE
                    ,MIN_SUPPORT
                    ,array_seller_state
                    ,CANTIDAD_MAXIMA_COND_ANTECEDENTE
                    ,CANTIDAD_MAXIMA_COND_CONSECUENTE
                    )

rules_df = rules_df[ [ 'antecedent', 'consequent', 'support', 'confidence', 'lift', 'conviction', 'upper_support','original_dataframe'] ]
```

Después de ejecutar la función “generate_association_rules”, visualizamos las 10 primeras reglas generadas junto a sus métricas de evaluación(Soporte, Confianza, Lift, Conviction). Cabe destacar que los condicionales del antecedente y consecuente se encuentran como cadena de texto – en una misma columna - separados cada artículo por coma.

Tabla 9 - Paso f) Python FP-GROWTH – Resultados de las 10 Primeras Reglas Generadas

Antecedente	Consecuente	Soporte	Confianza	Lift	Conviction	Soporte del antecedente	Cantidad Registros en el dataset
('True_housewares' , '	('True_health_ beauty',)	0.20	0.93	4.65	11.79	440	2194

True_sports_leisure', ' True_watches_gifts')							
('True_housewares', ' True_watches_gifts', ' True_health_beauty')	('True_sports_leisure',)	0.20	0.94	4.67	12.52	440	2194
('True_sports_leisure', ' True_watches_gifts', ' True_computers_accessories')	('True_health_beauty',)	0.20	0.94	4.62	14.00	448	2194
('True_watches_gifts', ' True_computers_accessories', ' True_health_beauty')	('True_sports_leisure',)	0.20	0.94	4.61	13.53	448	2194
('True_sports_leisure', ' True_auto', ' True_bed_bath_table')	('True_health_beauty',)	0.20	0.92	4.56	10.37	444	2194
('True_auto', ' True_bed_bath_table', ' True_health_beauty')	('True_sports_leisure',)	0.20	0.94	4.67	14.41	444	2194
('True_auto', ' True_furniture_decor', ' True_health_beauty')	('True_computers_accessories',)	0.20	0.91	4.44	8.55	448	2194

('True_sports_leisure', ' True_auto', ' True_furniture_decor')	('True_health_beauty',)	0.21	0.93	4.41	11.53	463	2194
('True_auto', ' True_furniture_decor', ' True_health_beauty')	('True_sports_leisure',)	0.21	0.94	4.44	12.57	463	2194
('True_housewares', ' True_sports_leisure', ' True_auto', ' True_computers_accessories')	('True_health_beauty',)	0.20	0.96	4.78	18.36	439	2194

g) Para un mejor análisis de las reglas encontradas se eliminan las reglas repetidas.

Código Python implementado:

```
rules_df_tmp = rules_df
list_antcedent = rules_df_tmp['antecedent'].values.tolist()
list_consequent = rules_df_tmp['consequent'].values.tolist()

## Transforma el dataframe en listas de tuplas
def concat_list_to_list(a_tuple,b_tuple):
    a_list=list(a_tuple)
    b_list=list(b_tuple)
    for i in range(len(b_list)):
        a_list.append(b_list[i])

  return a_list

## Valida si existe la regla en el dataframe
def if_key_exist(array_key,key):
    keys=[]
    if len(array_key)>0:
        keys=array_key.keys()

    if key in keys:
        ban=True
    else:
```

```
                ban=False

    return ban

## Devuelve las reglas repetidas
def get_count_rule(a_list,c_list):
    list_antecedent_consecuent_rules=[]
    list_rules= [''] * len(a_list)
    list_rules_counts= {}
    ## Recorre los condicionales del antecedente y consecuente
    for i in range(len(a_list)):
            list_rule_tmp = concat_list_to_list(a_list[i],c_list[i])
            list_rule_tmp.sort()
            s_tmp= ''.join( str( list_rule_tmp[v]
                        +',' if v < ( len(list_rule_tmp) -1)
                        else list_rule_tmp[v])
                        for v in range(len(list_rule_tmp))
                        )
            if if_key_exist(list_rules_counts,s_tmp):
                list_rules_counts[s_tmp]['conteo']= int(
                        list_rules_counts[s_tmp]['conteo']
                        ) +1
            else:
                list_rules_counts[s_tmp]={'conteo':1,'indice':i}

    ## Recorremos la lista de reglas repetidas
    for r in list_rules_counts:
        list_antecedent_consecuent_rules.append({
                                    'regla':r
                                    ,'indice':list_rules_counts[r]['indice']
                                    ,'conteo':list_rules_counts[r]['conteo']
                                    })

    df_rules_count = pd.DataFrame(data=list_antecedent_consecuent_rules)

    return df_rules_count

## Muestra las reglas y las veces de repetición
rules_counts = get_count_rule(list_antcedent,list_consequent)
new_rules_df = rules_df_tmp.drop(rules_counts['indice'].values.tolist())
```

Después de ejecutar la función “get_count_rule”, visualizamos las reglas que se encuentran repetidas para luego eliminarlas.

Tabla 10 - Paso g) Python FP-GROWTH – Resultados de Reglas Únicas

Regla	Índice	Conteo
True_sports_leisure, True_watches_gifts ,True_health_beauty =>True_housewares	0	1

True_health_beauty, True_watches_gifts ,True_housewares =>True_sports_leisure	1	1
True_computers_accessories, True_watches_gifts ,True_health_beauty =>True_sports_leisure	2	1
True_computers_accessories, True_health_beauty ,True_sports_leisure =>True_watches_gifts	3	1
True_auto, True_bed_bath_table ,True_health_beauty =>True_sports_leisure	4	1
True_bed_bath_table, True_health_beauty ,True_auto =>True_sports_leisure	5	1
True_furniture_decor, True_health_beauty ,True_auto =>True_computers_accessories	6	1
True_auto, True_furniture_decor ,True_health_beauty =>True_sports_leisure	7	1
True_furniture_decor, True_health_beauty ,True_auto =>True_sports_leisure	8	1
True_auto,True_computers_accessories , True_sports_leisure,True_health_beauty =>True_housewares	9	1

h) Para un mejor análisis, se dividen por columnas las condiciones del antecedente y consecuente.

Código Python implementado:

```
from ast import literal_eval
## Permite separar la cadena de texto del antecedente y consecuente
## en columnas independientes
def Separed_GenerateNewArrayRules(array):
array_rules_separated=[]
    i_row = 0
    row_max = len(array)
    max_num_cond_ante = 0
    max_num_cond_conse = 0
   ## Recorre el Arreglo de Reglas Generadas
   for index,row in array.iterrows():
        i_row = i_row + 1

        antecedent = str(row['antecedent'])
        antecedent = literal_eval(antecedent)
        element_antecedent = len(antecedent)
        if element_antecedent > max_num_cond_ante:
            max_num_cond_ante = element_antecedent

        consequent = str(row['consequent'])
        consequent = literal_eval(consequent)
        element_consequent = len(consequent)
        if element_consequent > max_num_cond_conse:
          max_num_cond_conse = element_consequent

    i_row = 0
   ## Recorre el arreglo de condicionales que se generaron al
  ## separar por el delimitador ","
   for index,row in array.iterrows():
            i_row = i_row + 1
            antecedent = str(row['antecedent'])
            antecedent = literal_eval(antecedent)
            element_antecedent = len(antecedent)
            i_i = 0
            Cond_Ante_1=0
            Cond_Ante_2=0
            Cond_Ante_3=0
            Cond_Ante_4=0
            Cond_Conse_1=0
            ## Establece la posición de columna del antecedente
            if element_antecedent < 2:
              Cond_Ante_1 = antecedent[0]
            else:
              if element_antecedent < 3:
                Cond_Ante_1 = antecedent[0]
                Cond_Ante_2 = antecedent[1]
              else:
                if element_antecedent < 4:
                  Cond_Ante_1 = antecedent[0]
                  Cond_Ante_2 = antecedent[1]
                  Cond_Ante_3 = antecedent[2]
                else:
                  Cond_Ante_1 = antecedent[0]
                  Cond_Ante_2 = antecedent[1]
                  Cond_Ante_3 = antecedent[2]
                  Cond_Ante_4 = antecedent[3]
```

```
    consequent = str(row['consequent'])
    consequent = literal_eval(consequent)
    element_consequent = len(consequent)
    if element_consequent < 2:
      Cond_Conse_1 = consequent[0]

   ## Genera un nuevo arreglo con los condicionales, separados
  ## en columnas, del antecedente y del consecuente
    array_rules_separated.append({ 'Cond_Ante_1':Cond_Ante_1
                                  , 'Cond_Ante_2':Cond_Ante_2
                                  , 'Cond_Ante_3':Cond_Ante_3
                                  , 'Cond_Ante_4':Cond_Ante_4
                                  , 'Consequent': Cond_Conse_1
                                  ,'Support':row['support']
                                  ,'Confidence':row['confidence']
                                  ,'Lift':row['lift']
                                  ,'Conviction':row['conviction']
                                  ,'Upper_Support':row['upper_support']
                                  ,'Original_Dataframe':row['original_dataframe'] })

    return array_rules_separated

  ## Generación del nuevo arreglo que tiene las columnas independientes de
  ## los condicionales del antecedente y consecuente
  array_rules_formated = Separed_GenerateNewArrayRules(array_reglas)

  columns = ['Cond_Ante_1', 'Cond_Ante_2','Cond_Ante_3','Cond_Ante_4'
  ,'Consequent','Support','Confidence','Lift','Conviction','Upper_Support','Original
  _Dataframe']

  df_rules_formated = pd.DataFrame(array_rules_formated, columns=columns)
```

i) A continuación, se muestran las 10 reglas generadas por el algoritmo FP-GROWTH con mayor soporte y confianza:

Tabla 11 - Paso i) Python FP-GROWTH – Resultados de las Mejores 10 Reglas

Cod_Ante_1	Cond_Ante_2	Cond_Ante_3	Cond_Ant_4	Consequent	Support	Confidence	Lift	Conviction	Upper	Original

									Support	Data Frame
True_computers_accesories	True_furrniture_decors	True_health_beauty	0	True_sports_leisure	0.253418	0.904065	3.57	7.28	556	2194
True_housewares	True_sports_leisure	True_furniture_decor	0	True_health_beauty	0.240656	0.901024	3.74	7.67	528	2194
True_housewares	True_computers_accessories	True_furniture_decor	0	True_sports_leisure	0.239289	0.905172	3.78	8.02	525	2194
True_computers_accessories	True_garden_tools	True_health_beauty	0	True_sports_leisure	0.226071	0.906764	4.01	8.30	496	2194
True_sports_leisure	True_auto	True_computers_accessories	0	True_health_beauty	0.224248	0.940727	4.20	13.09	492	2194
True_auto	True_computers_accessories	True_health_beauty	0	True_sports_leisure	0.224248	0.912801	4.07	8.90	492	2194
True_housewares	True_auto	True_health_beauty	0	True_sports_leisure	0.220602	0.928983	4.21	10.97	484	2194
True_housewares	True_sports_leisure	True_auto	0	True_health_beauty	0.220602	0.921905	4.18	9.98	484	2194
True_bed_bath_table	True_computers_accessories	True_health_beauty	0	True_sports_leisure	0.219690	0.925144	4.21	10.42	482	2194
True_housewares	True_computers_accessories	True_furniture_decor	True_health_beauty	True_sports_leisure	0.218778	0.937500	4.28	12.50	480	2194

En el siguiente capítulo nos enfocaremos en evaluar los modelos predictivos desarrollados en APRIORI y Fp-Growth. Tomaremos en cuenta la forma de implementación, los tiempos de ejecución y las reglas generadas. Luego de esto, escogeremos el más apropiado para resolver el problema planteado.

CAPÍTULO 4

PRUEBAS Y ANÁLISIS DE RESULTADOS

En este capítulo se evaluarán los modelos predictivos implementados en Weka y en Python respectivamente desde la perspectiva técnica y también tomando en cuenta las reglas generadas.

4.1. Análisis Comparativo de Modelos

El conjunto de datos utilizado es de 2194 transacciones con 13 columnas – atributos.

Tabla 12 - Comparación de Modelos – APRIORI y FP - Growth

	Fp-Growth	APRIORI	Observación
Elaboración del Modelo	Usamos Python "Google Colab"	Usamos la herramienta Weka Más rápido	En Python, aplicamos el siguiente tratamiento al conjunto de datos: Para disminuir tiempos de ejecución, se cambió los nombres de columnas cabeceras de “Texto” a “Entero”. Para entender las reglas que se generan, se cambió los valores de las columnas de “Boolean” a “Texto” (CodigoColumna_ValorBoolean). Para no tener “Combos de artículos” con algún producto sin relación con otros artículos, se eliminaron los patrones frecuentes que tengan el valor Boolean “False”. Esto también sirvió para disminuir aún más el tiempo de generación de las reglas. Para mayor facilidad de análisis y para integrarlos con alguna herramienta software, se muestran en columnas independientes todos los atributos de las “Reglas de asociación” generadas.
Generación de Reglas	En base a la frecuencia, ordenada descendentemente	En base al orden de posición al cargar el conjunto de datos	Ambos modelos predictivos solo pudieron generar reglas con un patrón de frecuencia bajo – 219 transacciones de un total de 2194 transacciones -, es decir que el mínimo soporte utilizado fue 10%. APRIORI y Fp-Growth no generan las mismas reglas. APRIORI toma en cuenta el orden en que se encuentran las columnas de las categorías de artículo. Fp-Growth encuentra los patrones más frecuentes en base a los mas altos valores de frecuencia de cada columna del conjunto de datos.

			APRIORI arroja mayor cantidad de reglas porque en cada iteración origina más reglas candidatas que al final se filtran de acuerdo con el mínimo soporte y confianza configurados.
Repetición de Reglas	Si repite	No repite	APRIORI no repite reglas, ni siquiera cuando los productos dentro de los condicionales del antecedente y consecuente estén en otro orden. En cambio, Fp-Growth arroja reglas repetidas porque no toma en cuenta el orden de los productos dentro de los condicionales del antecedente y consecuente.
Generación de Indicadores Estadísticos de Evaluación	Se los debe calcular	Automáticamente	APRIORI genera las reglas y sus indicadores estadísticos de Lift y Conviction de forma automática. En cambio, Fp-Growth sólo calcula los valores porcentuales de soporte y confianza. El resto de los indicadores (Lift, Conviction, Frecuencia de la regla, entre otros) se los debe calcular al construir el modelo. Ejemplo: APRIORI muestra el soporte numérico del antecedente y consecuente. En cambio, Fp-Growth solo muestra el soporte numérico de la regla (consecuente)
Tiempo de Ejecución	Más rápido 23 segundos	Mas lento 32 segundos	La diferencia se incrementa considerablemente para aquellos conjuntos de datos con mayor cantidad de transacciones y columnas La función que estima el tiempo de ejecución para APRIORI es O(2n) para Fp-Growth es O(n), donde “n” es la cantidad de transacciones del conjunto de datos [43].

4.2. Análisis de los Combos de Artículos Generados

El conjunto de datos tiene 2194 transacciones y 13 atributos - 10 de artículos, 1 de Cliente, 1 de Vendedor y 1 de Periodo de Tiempo - . Los “Combos de artículos” generados fueron:

FP-Growth Python: 45 reglas generadas

- Para filtrado de patrones se configuro que la frecuencia debe ser mayor a la mínima cantidad de repetición y este valor es “16” porque se espera que la regla se encuentre como mínimo en 16 meses de los 24 que se tiene de historia.

Las 10 mejores reglas del modelo tienen un soporte alrededor del 23%, una “Confianza” mayor al 90%, con un “Lift” que indica una correlación positiva entre el antecedente y consecuente, además con un “Conviction” que nos confirma que no son resultado de la causalidad. Hay que tomar en cuenta que las reglas generadas tienen un bajo soporte porque no existen patrones de transacciones con altos valores de frecuencia en el conjunto de datos original (E-commerce de Brasil).

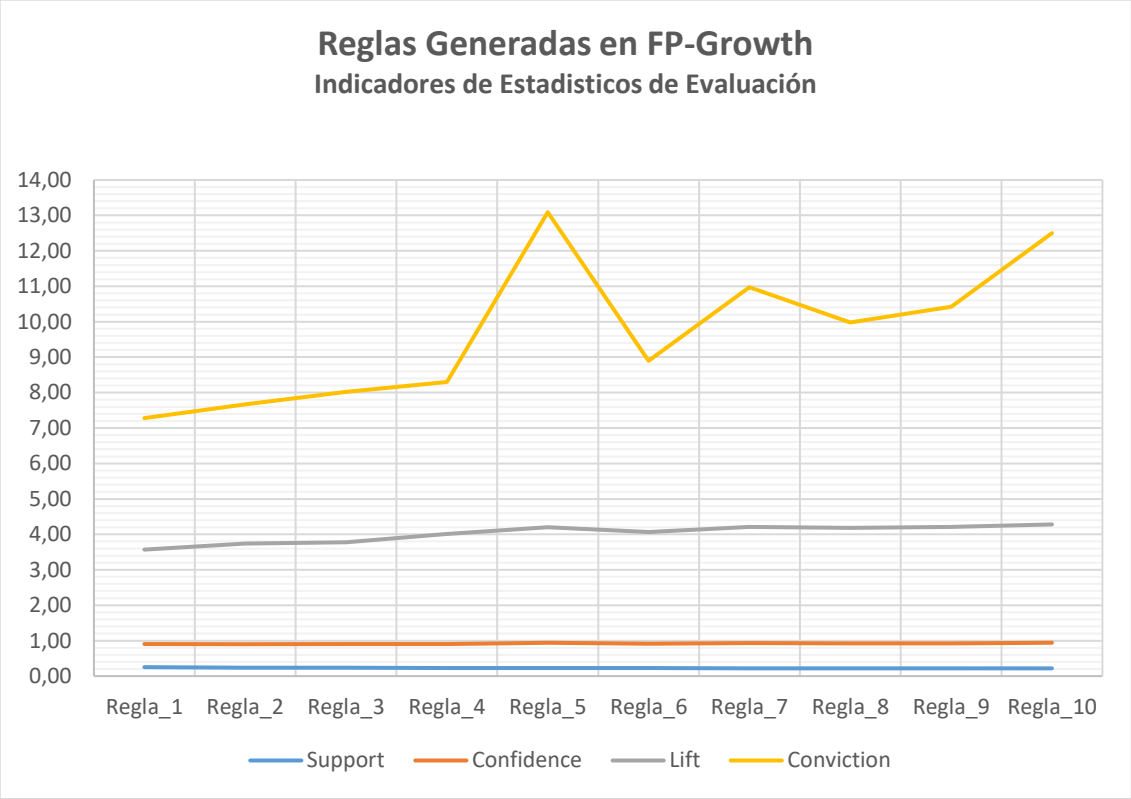


Figura 4.1 - FP Growth: Las 10 mejores Reglas de Asociación - Indicadores Estadísticos

De las 10 categorías de productos, las que más participaron en la generación de los “Combos de artículos” fueron Sports Leisure y Health Beauty. Las que menos participaron fueron Watches Gifts, Garden Tools y Cool Stuff.

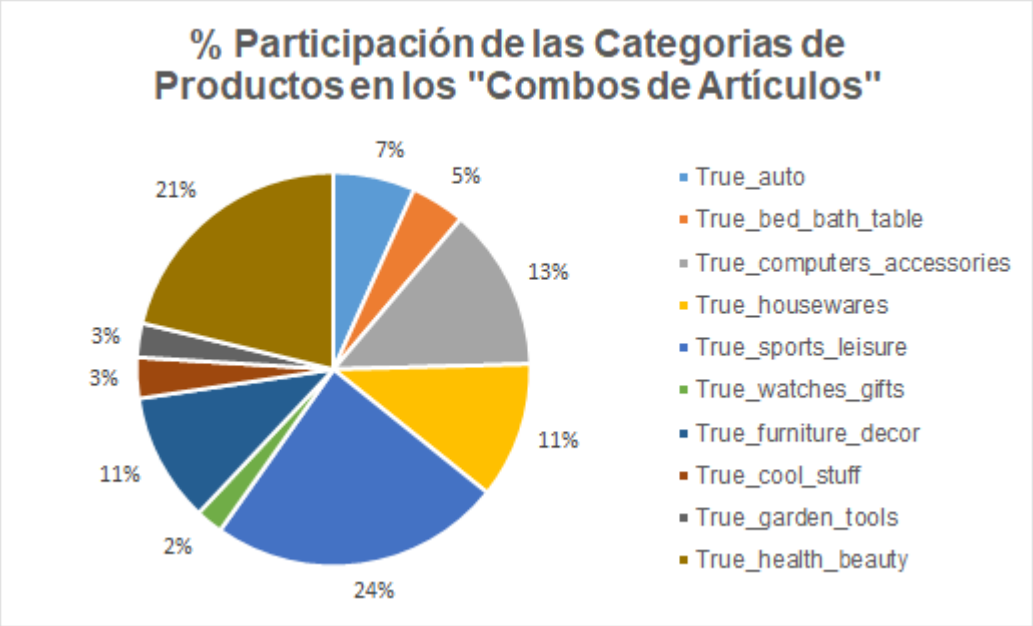


Figura 4.2 - FP Growth: % Participación de las categorías en los Combos

Los “Combos de artículos” generados tienen de 4 a 5 productos. Se destacan las siguientes categorías para los condicionales del antecedente y consecuente:

- *Antecedente - Condicional 1*: Housewares
- *Antecedente - Condicional 2*: Computers Accessories, Sports Leisure, Furniture Decor
- *Antecedente - Condicional 3*: Health Beauty, Furniture Decor
- *Antecedente - Condicional 4*: Health Beauty
- *Consecuente - Condicional 1*: Sports Leisure, Health Beauty

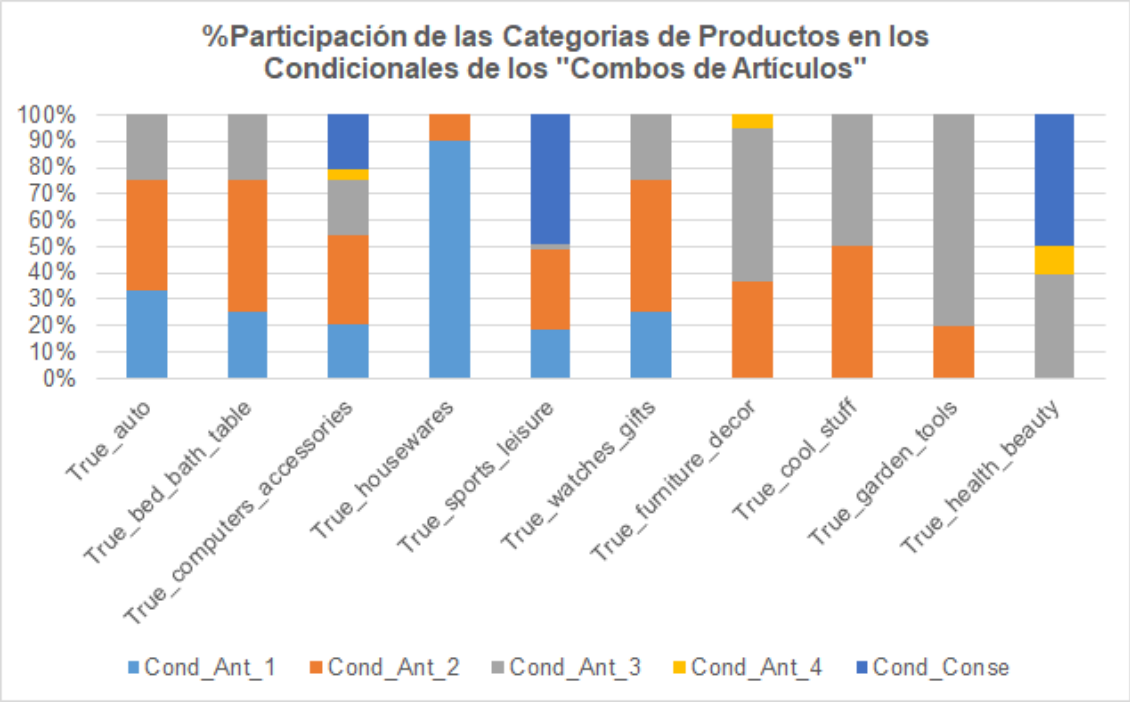


Figura 4.3 - FP Growth: % Participación de las categorías en los condicionales

Las 3 mejores reglas generadas – más alto valor de soporte, confianza y con valores positivos de Lift y Conviction - por este modelo son:

Tabla 13 - Modelo FP Growth Python – Las 3 mejores Reglas

# Regla	Antecedente	Consecuente	Métricas Evaluación
1	computers_accessories furniture_decor	sports_leisure	Soporte 25.34 % Confianza 90.40% Lift 3.57

	health_beauty		Conviction 7.28
2	housewares sports_leisure furniture_decor	health_beauty	Soporte 24.06% Confianza 90.10% Lift 3.74 Conviction 7.67
3	housewares computers_accessories furniture_decor	sports_leisure	Soporte 23.93 % Confianza 90.51% Lift 3.78 Conviction 8.02

APRIORI Java-Weka : 73 reglas generadas

- Se configuro un mínimo soporte del 10%, es decir que 219 veces debe estar repetida la transacción con respecto al universo del conjunto de datos (2194).

Las 10 mejores reglas del modelo APRIORI tienen un soporte alrededor del 21%, una “Confianza” mayor al 90%, con un “Lift” que indica una correlación positiva entre antecedente y consecuente, además con un “Conviction” que nos confirma que la reglas no son resultado de la causalidad. Hay que tomar en cuenta que las reglas generadas tienen un bajo soporte porque no existen patrones de transacciones con altos valores de frecuencia en el conjunto de datos original (E-commerce de Brasil).

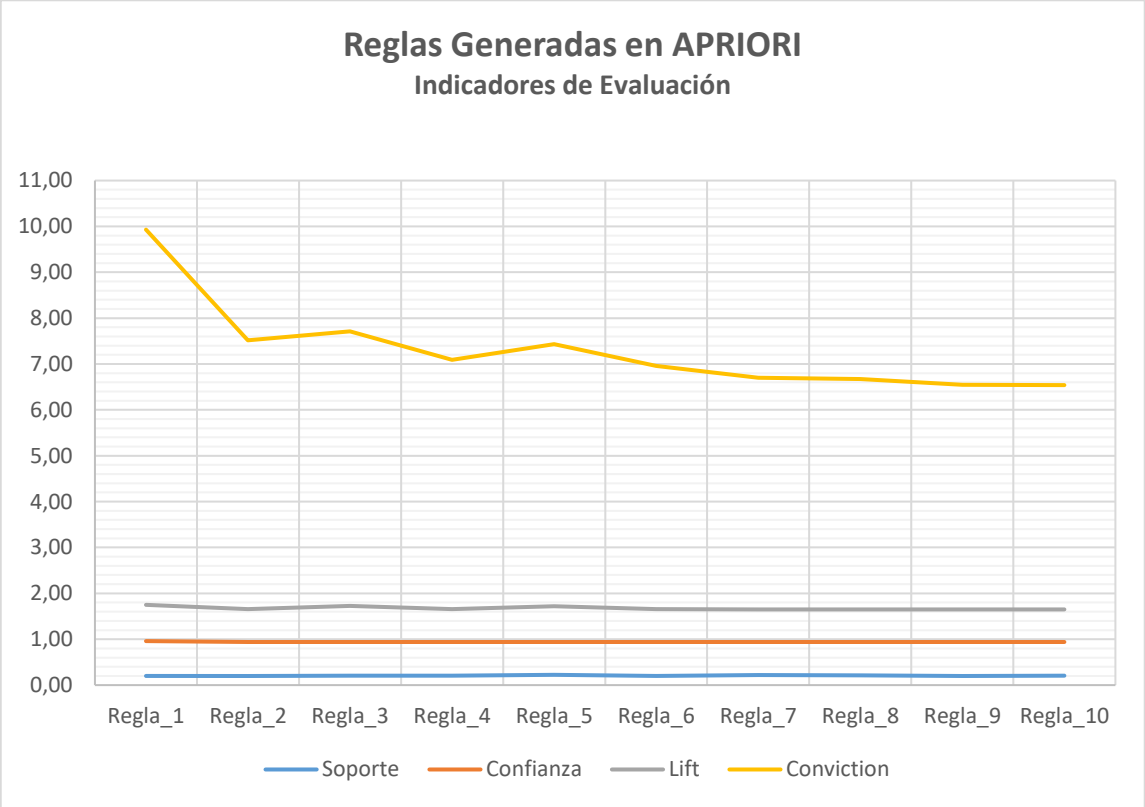


Figura 4.4 - APRIORI: Las 10 mejores Reglas de Asociación en - Indicadores Estadísticos

De las 10 categorías de productos, las que más participaron en la generación de los “Combos de artículos” fueron Sports Leisure y Health Beauty. Las que menos participaron fueron Bed Bath Table y Cool Stuff.

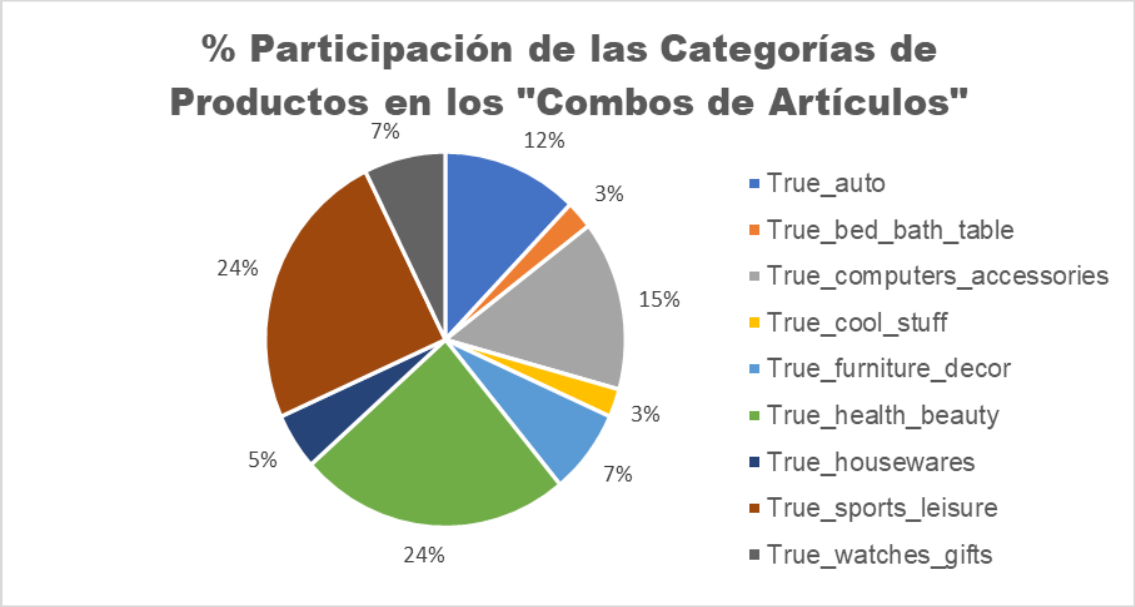


Figura 4.5 - APRIORI: % Participación de las categorías en los Combos

Los "Combos de artículos" generados tienen de 4 a 5 productos. Se destacan las siguientes categorías para los condicionales del antecedente y consecuente:

- *Antecedente - Condicional 1*: Auto, Cool Stuff
- *Antecedente - Condicional 2*: Bed Bath Table, Furniture Decor
- *Antecedente - Condicional 3*: Watches Gifts
- *Antecedente - Condicional 4*: Housewares
- *Consecuente - Condicional 1*: Sports Leisure, Health Beauty

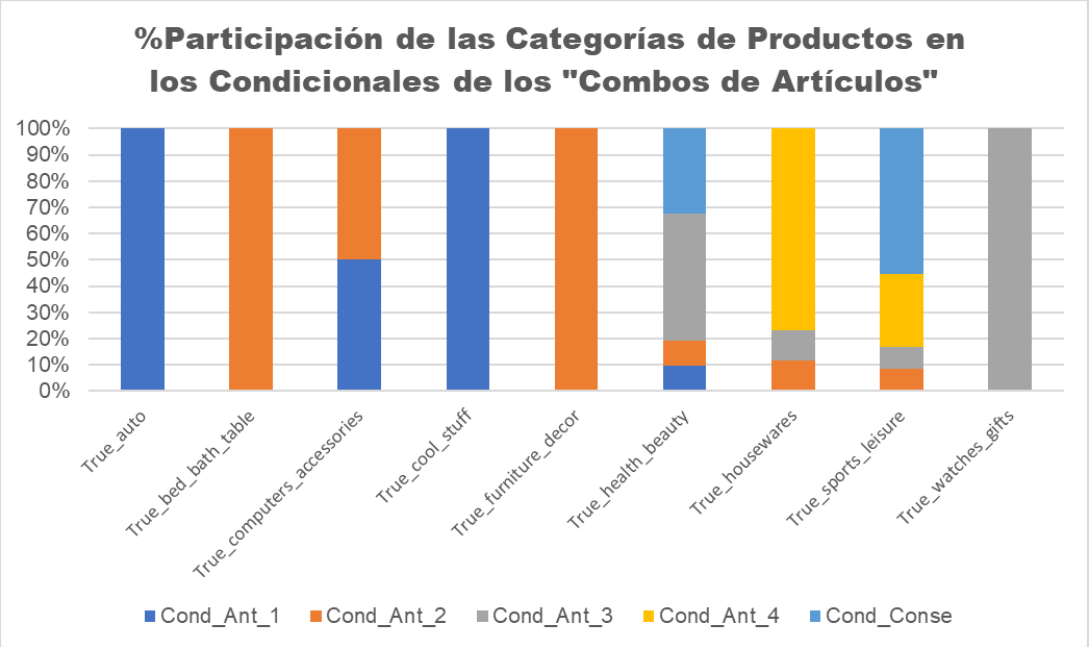


Figura 4.6 - APRIORI: % Participación de las categorías en los condicionales

Las 3 mejores reglas generadas – más alto valor de soporte, confianza y con valores positivos de Lift y Conviction - por este modelo son:

Tabla 14 - Modelo APRIORI Java – Las 3 mejores Reglas

# Regla	Antecedente	Consecuente	Métricas Evaluación
1	auto computers_accessories sports_leisure	health_beauty	Soporte 22.42 % Confianza 94% Lift 1.72 Conviction 7.43

2	computers_accessories furniture_decor health_beauty housewares	sports_leisure	Soporte 21.88 % Confianza 94% Lift 1.65 Conviction 6.67
3	auto furniture_decor health_beauty	sports_leisure	Soporte 21.10 % Confianza 94% Lift 1.65 Conviction 6.55

4.3. Selección del Modelo Predictivo

Se elige el modelo predictivo implementado en Python, basado en Fp-Growth, porque:

- Genera reglas con valores de Soporte más altos, es decir encuentra reglas que son más frecuentes en el conjunto de datos. Esta forma de encontrar reglas también es una de las causas para que existan diferentes reglas entre ambos modelos.
- Genera reglas con una mayor correlación entre los artículos del antecedente y consecuente. Esto se puede interpretar por los valores positivos de los indicadores de evaluación Lift y Conviction.
- Genera reglas segmentadas por cliente, ya sea por el estado de vendedor (Seller_State) o por estado cliente (Customer_State).

- Genera reglas con una mayor variedad de productos, en los condicionales del antecedente y consecuente.
- Tiene un mejor tiempo de ejecución cuando la cantidad de transacciones y columnas se incrementan considerablemente.
- Es más apropiado para el análisis o integración con otras plataformas porque en el formato de salida de las reglas generadas se tiene cada campo como una columna individual.

En el siguiente y último capítulo concluiremos el trabajo de tesis y recomendaremos ciertas sugerencias para futuros estudios que utilicen las “Reglas de asociación” como herramienta para definir “Combos de artículos”.

CONCLUSIONES Y RECOMENDACIONES

Con respecto a los objetivos generales y parciales que se plantearon al inicio, se concluye lo siguiente:

Tabla 15 - Evaluación de los objetivos planteados

Objetivo	Cumplimiento	Observación
Generación automática de los "Combos de artículos" mediante un modelo predictivo del tipo asociativo.	SI	Implementado en FP-Growth de Python.
Focalización de los "Combos de artículos" para que generen promociones por alguna categoria de cliente y alguna categoria de articulo.	SI	Se generaron reglas de asociación por categorías de artículos y por los estados geográficos del vendedor y cliente.
Especificación de las características del conjunto de datos.	SI	Se utilizo un conjunto de datos con 2194 transacciones y 13 columnas – atributos. Inicialmente el conjunto de datos tenía más de 100,000 transacciones con 52 columnas – atributos.
Implementación de los modelos predictivos en herramientas open source.	SI	Se realizaron 2 modelos predictivos: APRIORI – Java y FP-Growth en Python.
Evaluación de los modelos predictivos y Selección del más adecuado	SI	Se evaluó los 2 modelos predictivos mediante sus tiempos de ejecución, reglas

		generadas, formato de salida de las reglas generadas. Se selecciono el modelo predictivo en FP-Growth Python porque fue mejor en todos los puntos antes mencionados

Además, se agregan las siguientes conclusiones y recomendaciones que pueden ser de mucha ayuda para trabajos futuros:

El conjunto de datos, en la medida de lo posible, debe obtenerse desde un Datawarehouse y no desde una Base de datos Relacional. Con esto evitaríamos el sobre esfuerzo en el tratamiento de los datos y posibles errores por la falta de conocimiento de la lógica del negocio involucrada en cada dimensión o indicador utilizado.

El conjunto de datos a utilizar debe tener muchas más características de clientes para si lograr una segmentación más específica. El conjunto de datos utilizado solo tenía la ciudad y estado del cliente.

En cualquier tipo de negocio los “Combos de artículos” se generan con una periodicidad semanal o mensual. El modelo predictivo en Python (Fp-Growth), debe generar estos “Combos de artículos” con la misma frecuencia. Debido a esto, es necesaria la integración con la Plataforma de Inteligencia de Negocios(BI) para así tener las “Reglas de asociación” actualizadas.

- Puede integrarse con el Datawarehouse mediante el proceso de carga de información (ETL).
- Además, se debe evaluar la posibilidad de:
    - Integrar las “Reglas de asociación” con la herramienta visor de BI, para que los usuarios puedan consumir el nuevo conocimiento sin tener que salir de su ambiente habitual de trabajo.
    - Ejecución del modelo predictivo Fp-Growth, desde la herramienta visor de BI (al momento que el usuario interactúa). Con esto, podríamos generar “Reglas de asociación” para todas las dimensiones cargadas en la herramienta visor de BI
        - Exceptuando la dimensión tiempo porque es la que permite encontrar la frecuencia de los patrones.

APRIORI, en el visor Weka es sencillo de usar, pero al intentar consumirlo mediante la librería-API de Weka se complica mucho por la falta de documentación.

- Es necesario consumirlo por medio de la librería-API para así, poder generar el archivo de “Reglas de asociación” y cargarlo en las plataformas antes mencionadas. WEKA arroja el resultado como una cadena de texto.

Fp-Growth es más rápido que APRIORI [44]

- Alrededor de 2 veces más, si se incrementan considerablemente las transacciones.

- Alrededor de 4 veces más, si se incrementan considerablemente las columnas - atributos.

El entorno de trabajo para Python “Google Colab”, es una buena opción para ejecutar modelos predictivos con conjuntos de datos no tan pesados (en transacciones ni en columnas). Además, es gratuito y se encuentra en la nube. Weka también es gratuito, pero debe ejecutarse en una máquina local y por eso depende de las capacidades computacionales que se tiene al alcance.

Para que los usuarios de negocio utilicen las “Reglas de asociación” que conforman los “Combos de artículos”, independiente del modelo predictivo (FP-Growth o APRIORI), es necesario mostrarles que dichas “Reglas de asociación” se pueden encontrar manualmente

- Esto se puede realizar mediante una tabla dinámica de Excel que apunte al conjunto de datos. Luego debe colocarse jerárquicamente los atributos(condicionales) que aparecen en el antecedente y consecuente. Al final se debe poner al campo “Fecha o Periodo” como métrica de “Conteo Distinto”.

Para que los usuarios de negocios no dejen de analizar algún “Combo de artículos”, se deben unificar las “Regla de asociación” que arrojen ambos modelos predictivos (FP-Growth y APRIORI)

- Ambos modelos se podrían ejecutar en la misma plataforma de API Weka o Python, y al final hacer un subproceso que las unifique y etiquete:

- Las existentes en ambos algoritmos
- Las existentes solamente en FP-Growth
- Las existentes solamente en APRIORI

Es importante que las “Reglas de asociación” de los “Combos de artículos” se generen con unidades de compra, para así incrementar los niveles de venta al mismo tiempo que se disminuye los niveles de stock.

Es fundamental generar “Combos de artículos” exclusivos de las temporadas, tales como: carnaval, día de la madre, navidad entre otras. Se debe utilizar la información de las temporadas de años anteriores - pueden ser días distintos por cada año - y también se debe considerar las subcategorías, en vez de los artículos, porque el surtido varía considerablemente de una temporada a otra.

Si se necesita generar “Combos de artículos” para días puntuales, se debe tener un nivel de detalle-granularidad más bajo que la “Fecha Diaria”. Es por eso por lo que se sugiere la factura emitida por el punto de venta(POS).

BIBLIOGRAFÍA

[1] Bing Liu, Wynne Hsu, Shu Chen, Yiming Ma, “Analyzing the Subjective Interestingness of Association Rules”, National University of Singapore, 2000, pág. 53.

[2] Castañeda G, Rodríguez M, “La Minería de Datos como herramienta de Marketing: Delimitación y Evaluación del resultado”, Facultad de CC. EE., Departamento de Comercialización e Investigación de mercados, Universidad de Granada, España, 2005, pág. 2.

[3] Fayyad Usama, Piatesky-Shapiro G., Smyth P., “Knowledge Discovery and Data Mining: Towards a Unifying Framework”, AAAI, 1996, pág. 83.

[4] Fayyad Usama, Piatesky-Shapiro G.; Smyth P., “From Data Mining to Knowledge Discovery in Databases”, AAAI, 1996, pág. 41.

[5] Oscar Marbán, Antonio Amescua, Juan J. Cuadrado, Luis García, “Cost Drivers of a Parametric Cost Estimation Model for Data Mining Projects (DMCOMO)”, Universidad Carlos III de Madrid (UC3M), 2002, pág. 1.

[6] Ana Azevedo, Manuel Santos, "KDD, SEMMA and CRISP-DM: A parallel overview", IADIS - European Conference on Data Mining – Amsterdam The Netherlands, 2008, pág. 183.

[7] F. Javier Martínez de Pisón Ascacibar, “Optimización mediante técnicas de minería de datos del ciclo de recocido de una línea de galvanizado”, Universidad de la Rioja, 2003, pág. 61.

[8] “Introduction to Data Mining and Knowledge Discovery”, Third Edition, Two Crows Corporation, 1999, pág. 22.

[9] Óscar Marbán[1], Gonzalo Mariscal[2], Javier Segovia[1], "A Data Mining & Knowledge Discovery Process Model", [1] Universidad Politécnica de Madrid, [2] Universidad Europea de Madrid, 2009, pág. 3.

[10] Pang-Ning Tan, Michael Stainbach, Anuj Karpatne, Vipin Kumar, “Chapter 5 Association Analysis: Basic Concepts and Algorithms”, Introduction to Data Mining (Second Edition), 2005, pág. 357 - 363.

[11] Weka, https://www.cs.waikato.ac.nz/ml/weka/ , Fecha de Consulta: Diciembre 2019.

[12] Francisco Gabriel Guil Reyes, “Minería de patrones temporales basados en redes de restricciones”, Universidad de Almería - España, 2009, pág. 26.

[13] Wilson Rojas, Mario Silva, “Introduction a Java: Guia de actividades prácticas”, Universidad del Bosque - Colombia, 2016, pág. 18.

[14] Celeste Guagliano, “Programación en Python-Vol1”, Buenos Aires, 2019, pág. 8.

[15] Anaconda, https://www.anaconda.com/distribution/, Fecha de Consulta: Diciembre 2019.

[16] Jupyter Notebook, https://jupyter.org/about, Fecha de Consulta: Diciembre 2019.

[17] María Alejandra Malberti Riveros, Graciela Elida Begueri , “Reglas de Asociación con los datos de una biblioteca universitaria”, Universidad de las Ciencias Informáticas - Cuba, 2015, pág. 35.

[18] pyfpgrowth 1.0, https://pypi.org/project/pyfpgrowth/, Fecha de Consulta: Diciembre 2019.

[19] pandas v0.25.3, https://pandas.pydata.org/, Fecha de Consulta: Diciembre 2019.

[20] Ariel Monteserin, “Reglas de Asociación”, Universidad Nacional del Centro de la Provincia de Buenos Aires - Argentina, 2018, pág. 33.

[21] Roberto Navarro Cuervo, y Luz Marina Sierra Martinez, “Herramienta software para el análisis de canasta de mercado sin selección de candidatos”, Universidad Industrial de Santander - Colombia, 2009, pág. 60 - 68.

[22] IBM, “Reglas de asociación”, https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/clementine/nodes_associationrules.html, Fecha de Consulta: Diciembre 2019.

[23] IBM,“Datos tabulares frente a datos transaccionales”, https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mai

nhelp_client_ddita/clementine/tabularandtransactionaldata.html, Fecha de Consulta: Diciembre 2019.

[24] Pete Chapman, Julian Clinton, Randy Kerber, Thomas Khabaza, Thomas Reinartz, Colin Shearer, Rüdiger Wirth, “CRISP-DM Step-by-step data mining guide”, 2000, pág. 13 - 14.

[25] Víctor Galán Cortina, "Aplicación de la Metodología CRISP-DM a un Proyecto de Minería de Datos en el Entorno Universitario", Universidad Carlos III de Madrid, pág 22.

[26] Eray Ozkura, Cevdet Aykanat , “A Space Optimization for FP-Growth”, Bilkent University Ankara - Turkey, 2004, pág. 2.

[27] S.Sumathi, S.N Sivanandam , “Introduction to Data Mining and its Applications”, Department of Electrical and Electronics Engineering Tamil Nadu - India, 2006, pág. 712.

[28] Oracle News,“Versión final de Java Data Mining (JSR-73)”, http://www.dba-oracle.com/oracle_news/2004_11_02a_jave_data_mining_jsr_73_final_release.htm , Fecha de Consulta: Diciembre 2019.

[29] Ali Anaissi, Maddhu Goyal, “SVM-Based Association Rules for Knowledge Discovery and Classification”, Faculty of Engineering and Information Technology (FEIT) - University of Technology Sydney (UTS) - Australia, 2007, pág. 6.

[30] Reinhold Decker, Katharina Monien, “Market Basket analysis with neural gas networks and self-organising maps”, University of Bielefeld - Alemania, 2003, pág. 374.

[31] Aliakbar Niknafs, Soodabeh Parsa, Journal “A Neural Network Approach for Updating Ranked Association Rules, Based on Data Envelopment Analysis”, University of Kerman - Iran, 2011, pág 286.

[32] Ricardo TIMARÁN-PEREI, “EquipAsso: un algoritmo para el descubrimiento de conjuntos de ítems frecuentes sin generación de candidatos”, Universidad de Manizales, 2011, pág 65.

[33] Jiawei Han, Micheline Kamber, Jian Pei, “Chapter 6 Mining Frequent Patterns, Associations, and Correlations”, Data Mining Concepts and Techniques (Third Edition), 2012, pág. 244.

[34] Marko Svetina, Jože Zupančič, “How to Increase Sales in Retail with Market Basket Analysis”, Faculty of Organizational Sciences - University of Maribor, 2005, pág. 424.

[35] Josue Galarreta Vasquez, “Inducción de Reglas de Asociación en Base de datos de entidad retail”, Facultad de Ingeniería Civil de Sistemas y Arquitectura - Universidad Nacional “Pedro Ruiz Gallo - Lambayeque - Perú”, 2016, pág. 6.

[36] GrishmaKapadia, KavitaKalyandurgmath, “Market Basket Analysis of Consumer Buying Behaviour of a Lifestyle Store”, International Conference on Technology and Business Management March 23-25, 2015, pág 407.

[37] Ruxandra PETRE, "Data Mining Solutions for the Business Environment", University of Economic Studies - Bucharest Romania, 2013, pag 25 - 26.

[38] A. Jović*, K. Brkić*, N. Bogunović*, "An overview of free software tools for general data mining", Faculty of Electrical Engineering and Computing - University of Zagreb, 2014, pág 2.

[39] Economic Commission for Europe of The United Nations (UNECE) , "Glossary of Terms on Statistical Data Editing", Conference of European Statisticians Methodological material - Geneva, 2000, pág 4.

[40] United States Bureau of the Census, Software and Standards Management Branch, Systems Support Division, "Survey Design and Statistical Methodology Metadata",Washington, 1998, pág 14.

[41] H. Escobar, M. Alcivar,A. Puris ,“Aplicaciones de Minería de Datos en Marketing”,Revista Publicando,ISSN-e 1390-9304, Universidad Estatal de Quevedo - Universidad Estatal de Babahoyo, 2016, pág. 509-510.

[42] Raymond Moodley, Francisco Chiclana, Fabio Caraffini, Jenny Carter, “Application of Uninorms to Market Basket Analysis”, University of Huddersfield - UK, 2018, pág. 9-10.

[43] Alagukumar, Lawrance, "A Selective Analysis of Microarray Data using Association Rule Mining", University Tamil Nadu - India, 2015, pág 11.

[44] M.S. Mythili, A.R. Mohamed Shanavas, "Performance Evaluation of Apriori and FP-Growth Algorithms", College Tiruchirappalli - India, 2013, pág. 36.

[45] Waminee Niyagas, Anongnart Srivihok,and Sukumal Kitisin, Clustering e-Banking Customer using DataMining and Marketing Segmentation, University of Bangkok - Thailand 2006, pág 63 - 69.

[46] Reza Hafezi, Jamal Shahrabi, Esmaeil Hadavandi, "A bat-neural network multi-agent system (BNNMAS) for stock price prediction", University of Technology - Teheran Iran, 2015, pág 196 – 210.

[47] Binoy B. Nair, V.P Mohandas, N.R. Sakthivel, "A Genetic Algorithm Optimized Decision Tree SVM based Stock Market Trend Prediction System", Amrita School of Engineering - India, 2010, pág 2981 - 2988.

[48] Rosskyn D'Souza, Michal Krasnodebski, Alan Abrahams, "Implementation study: Using decision tree induction to discover profitable locations to sell pet insurance for a startup company", University of Mumba - University of Pennsylvania - Virginia Polytechnic, 2007, pág 281 - 287.

[49] Savi Gupta, Roopal Mamtora, "A Survey on Association Rule Mining in Market Basket Analysis", University Gurgaon - India, 2014, pág 411.

[50] Kavitha Venkatachari, Issac Davambu Chandrasekaran, "MARKET BASKET ANALYSIS USING FP GROWTH AND APRIORI ALGORITHM: A CASE STUDY OF MUMBAI RETAIL STORE", IBS Business School - Mumbai India, 2016, pág 56 - 62.

[51] M.Subithra, Dr. S.S.Dhenakaran, "Comparison Of Apriori And Partition Algorithms In Extracting Frequent Items", Alagappa University Karaikudi - India, 2016, pág 425 - 428.

[52] Haosong Li, “Scalable Association Rule Learning Algorithm for Very Large Dataset”, University of California Irvine, 2020, pág 4 - 5.

[53] [56] Blattberg R.C., Kim BD., Neslin S.A., "Market Basket Analysis Springer Vol 18", New York - USA, pág 340.

[54] Ronnie Alves, Domingo S. Rodríguez-Baena, J. Aguilar-Ruiz, "Gene association analysis: a survey of frequent pattern mining from gene expression data", Universidad Pablo de Olavide - España, 2009, pág 5.

[55] Samuel Musungwini, Tinashe Gwendolyn Zhou, Raviro Gumbo, Tinomuda Mzikamwi, "The Relationship Between (4ps) & Market Basket Analysis. A Case

Study Of Grocery Retail Shops In Gweru Zimbabwe", Universidad UNISA - Sudáfrica, 2016, pág 263.

[56] Pınar YAZGAN, "ASSOCIATION RULES AND MARKET BASKET ANALYSIS: A CASE STUDY IN RETAIL SECTOR" , Universidad de Estambul - Turquía, 2016, pág 59 – 61.

[57] Jonathan Bermúdez, Kevin Apolinario, Andrés G. Abad, "Layout Optimization and Promotional Strategies Design in a Retail Store based on a Market Basket Analysis", ESPOL - Ecuador, 2016, pág 3 – 4.

[58] Brazilian E-Commerce Public Dataset by Olist, https://www.kaggle.com/olistbr/brazilian-ecommerce, Fecha de Consulta: Octubre 2019.