

Infraestructura Big Data para la segmentación de clientes basado en RFM

Alberto Mur López

Tamara Ruiz Delgado

Leonardo Elisei

Tutora: Cristina Giner Pérez

Curso 2020/2021



Alberto Mur López, Tamara Ruiz Delgado y Leonardo Elisei

Infraestructura Big Data para la segmentación de clientes basado en

RFM

Trabajo Fin de Máster. Curso 2020/2021

Tutora: Cristina Giner Pérez

Máster en Big Data Engineer

Universidad de Barcelona

Instituto de Formación Continua

C/ Ciutat de Granada, 131

08018, Barcelona

Resumen

La segmentación de clientes a través de la exploración de sus hábitos de compra es un campo de estudio muy amplio con el que se pretende entender y mejorar la relación con los clientes por parte de las empresas. A través del estudio de un gran volumen variado de datos, entre los que se incluyen perfiles y transacciones de clientes, es posible agruparlos según sus hábitos para dirigir campañas de fidelización, captación o promoción. En este trabajo se propone un marco de trabajo que engloba el análisis, el tratamiento y procesado, la segmentación y una propuesta de indicadores para el seguimiento del rendimiento de dichas campañas.

Palabras clave — Hábitos de compra, RFM, K-Means, Segmentación retail, Big Data

Índice general

1. Introducción	1
1.1. Organización de la memoria	1
1.2. Objetivos	2
2. Estudios de los datos y el caso de uso	3
2.1. Origen y tipología de los datos	3
2.2. Caso de uso	3
2.3. Plataforma para el análisis	4
2.4. Análisis descriptivo	5
2.4.1. Tiendas	5
2.4.2. Clientes	6
2.4.3. Productos	8
2.4.4. Tickets	11
Sobre la forma de pago	12
Sobre el volumen de la cesta	13
2.5. Análisis cruzado	13
2.5.1. Relación entre clientes	14
2.5.2. Relación con productos	14
3. Segmentación	15
3.1. Análisis RFM	15
3.1.1. Análisis descriptivo	15
Compra más reciente	15
Frecuencia de compra	16
Gasto	17
3.1.2. Conclusión RFM	17
3.2. Agrupación con K-Means	19
3.2.1. Elección del número de segmentos	19
3.2.2. Interpretación de los segmentos	20
4. Resultados	23
4.0.1. Segmento: BEST	23
4.0.2. Segmento: PROMISING	24

4.0.3. Segmento: LOYAL	24
4.0.4. Segmento: NEW	25
4.0.5. Segmento: RISK	25
4.0.6. Segmento: LOST	26
5. Despliegue	27
5.1. Creación del clúster	27
5.2. Ejecución de tareas en el cluster	28
5.3. Entrenamiento del algoritmo	28
5.4. Ejecución a demanda del modelo entrenado	29
6. Conclusiones y líneas futuras	31
6.1. Conclusiones	31
6.2. Limitaciones y trabajo futuro	31
6.2.1. Limitaciones	31
6.2.2. Trabajo futuro	32
Bibliografía	34
A. Apéndice	35
A.1. Integración Notebook EMR con GIT	35
A.1.1. Configuración de una VPC	35
A.1.2. Creación del clúster EMR	39
A.1.3. Creación del repositorio de Git	43
A.1.4. Creación del cuaderno Jupyter	45
A.1.5. Vinculación del repositorio	49
A.2. Presentación y resultados para marketing	51
A.3. Scripts para el despliegue	63
A.3.1. Script de entrenamiento	63
A.3.2. Script de ejecución	67

Introducción

Este trabajo se enmarca dentro de la segmentación de clientes del sector retail. Con los datos proporcionados se pretende analizar los hábitos de compra e intereses de los clientes así como realizar una agrupación de los mismos.

Dada a la cantidad de datos proporcionados se hace imprescindible la utilización de herramientas de Big Data capaces de manipular grandes volúmenes de información. Para poder establecer las relaciones necesarias se hará uso de herramientas de análisis como Jupyter Notebooks en un entorno de ejecución PySpark conectados a un clúster de Amazon EMR.

Una vez configurado el sistema de análisis se procederá a la exploración descriptiva con el fin de extraer información relevante. Una vez realizado el análisis descriptivo se procederá a aplicar un método de segmentación basado en RFM complementado con el algoritmo de K-Means para la agrupación por segmentos de la cartera de clientes. Una vez identificados los segmentos y consolidado los indicadores clave de rendimiento, en inglés Key Performance Indicators (KPI), se establecerá el flujo para el despliegue de la plataforma de segmentación.

Finalmente, se elaborará una serie de propuestas y mejoras a realizar en futuras etapas.

1.1 Organización de la memoria

En esta memoria se detallan las fases que se han sucedido desde el inicio hasta la conclusión del proyecto.

En el capítulo dos se define el origen de los datos y el caso de uso para el cuál han sido proporcionados. Además se realiza un análisis descriptivo de los mismos.

En el tercer capítulo se procede a formalizar la segmentación que se va a realizar y la interpretación de los segmentos obtenidos.

En el capítulo cuatro se presenta un resumen descriptivo de los resultados obtenidos de la segmentación.

En el quinto capítulo se detallan las instrucciones de despliegue de la herramienta encargada de realizar la segmentación.

Por último se cierra la memoria con las conclusiones y líneas futuras del trabajo realizado.

1.2 Objetivos

Con este trabajo se pretende realizar una segmentación de clientes para el sector retail a través de un conjunto de datos proporcionado que contiene el perfil socio-demográfico y las transacciones realizadas por los clientes, así como un conjunto de productos categorizados. Debido al volumen de la información se espera que se realice un análisis de los datos y su segmentación empleando herramientas de Big Data.

Adicionalmente, se tratará de interpretar los segmentos obtenidos con la finalidad de que un departamento de marketing pueda utilizar esa información para realizar y gestionar nuevas campañas de captación, fidelización o promoción.

Finalmente se propondrán las herramientas de despliegue necesarias para la obtención de los segmentos.

Estudios de los datos y el caso de uso

En este capítulo se va realizar el análisis de los distintos conjuntos de datos proporcionados. Con esto se pretende conocer como están relacionados los datos y que información puede ser útil para la extracción de información.

2.1 Origen y tipología de los datos

Se han proporcionado 4 conjuntos de datos en total correspondientes a tiendas, clientes, productos y líneas de transacciones. Todos correspondientes a la cadena de supermercados Lidl en España. Aunque en el desarrollo del trabajo se proporcionaron en dos fases, en las cuales la primera solo contenía una muestra y en la segunda se ofrecía la información completa, en las siguientes secciones solo se detallará el análisis descriptivos del conjunto completo de los datos. Cabe destacar que algunas de las conclusiones obtenidas en la primera fase fueron alteradas debido a que en el conjunto completo se proporcionó nueva información.

2.2 Caso de uso

En los últimos años ha habido un aumento entre distintas compañías para posicionarse dentro del campo de la segmentación de clientes ya que los beneficios de la compañía pueden verse incrementados utilizando modelos de segmentación. La retención de clientes es más importante que la adquisición de nuevos clientes. De acuerdo al principio de Pareto (Srivastava, 2016), el 20% de los clientes contribuyen más a los ingresos que el resto de clientes.

La segmentación de clientes se puede realizar utilizando características únicas de los clientes para ayudar al negocio a personalizar planes de marketing, identificar tendencias, campañas de publicidad y ofrecer productos relevantes para sus clientes. La segmentación de clientes personaliza los mensajes hacia individuales mejorando

la comunicación con los grupos objetivo. Los atributos más utilizados en la segmentación de clientes son la localización, la edad, el género, los ingresos, el estilo de vida y las compras anteriores (Kotler, Armstrong, Rodríguez, Ibáñez & Roche, 2004, p. 250). De todas ellas, la más fácil de obtener son las compras anteriores debido a que es una información automática existente en el sistema. El resto de información, salvo la edad y el género, que a las edades que se manejan ya estaría consolidado, puede variar y es más complicado de mantener actualizada.

En este caso, se va a proponer una segmentación basada en el hábito de compra, ya que es una información disponible. Concretamente, el análisis RFM (Recencia, Frecuencia y Monetario) es una famosa técnica utilizada para evaluar a los consumidores basándose en su hábito de compra. Con esa información se va a desarrollar un método de puntuación basada en cuartiles para evaluar la Recencia, Frecuencia y el Gasto. Dicha puntuación, en el rango 444 a 111 y denominada puntuación RFM puede ser utilizada para analizar los hábitos de compra de cada cliente así como su evolución (Miglautsch, 2000).

A continuación, para realizar la segmentación por grupos se aplica el algoritmo de K-Means a las variables RFM. Finalmente, se analiza cada uno de los grupos obtenidos para interpretar que tipo de grupos se han obtenido.

2.3 Plataforma para el análisis

Los cuadernos para la realización del análisis se pueden encontrar en el siguiente repositorio de GitHub¹. Los pasos para vincular dicho repositorio a un cuaderno de AWS vinculado a un clúster de Amazon EMR se encuentran detallados en el anexo A.1.

Mediante Jupyter Notebooks conectados a un cluster de Amazon EMR, se ha realizado todo el análisis y la propuesta de segmentación. Finalmente, una vez desarrollado y validado el proceso, se empaqueta en un trabajo de PySpark para su ejecución a demanda.

¹<https://github.com/almul0/mbde-tfm-rfm>

2.4 Análisis descriptivo

2.4.1 Tiendas

El conjunto de tiendas proporcionado contiene la siguiente información:

city Ciudad donde se encuentra la tienda

isocountrycode Código ISO del país donde se encuentra la tienda

startdate Fecha en la que la tienda comenzó a operar

storeid Identificador único de la tienda

En el cuadro 2.1 se muestra la información descriptiva sobre los datos proporcionados. Se trata de 1127 registros en los cuales, más del 50% del campo startdate es desconocido. Como información se puede extraer que todas las tiendas pertenecen al territorio español y que hay tiendas en 552 ciudades de España.

summary	city	isocountrycode	startdate	storeid
Registros	1127	1127	560	1127
Únicos	552	1	27	1127
Min.	-	-	2015-12-17	ES0010
Max.	-	-	2019-08-22	ES7013
Ausentes (%)	0.0	0.0	50.31	0.0

Cuadro 2.1: Estadística descriptiva del conjunto de datos stores

En la imagen 2.1 se muestra la distribución de tiendas por ciudad para las 10 ciudades con más tiendas. Esto se corresponde en gran medida con el nivel de población para cada una de esas ciudades (Tabla 2.2), es decir, a mayor población mayor número de tiendas.

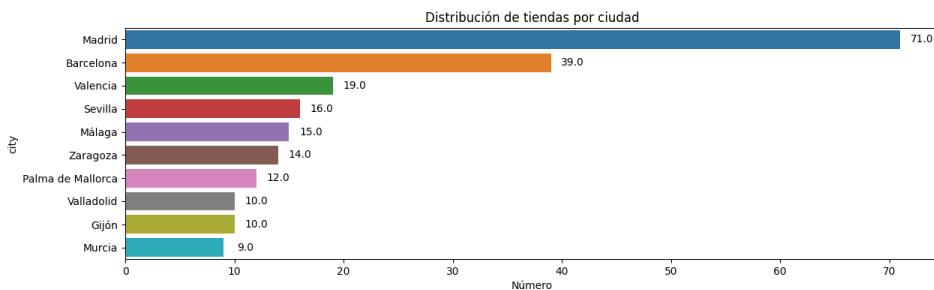


Figura 2.1.: Distribución de tiendas por ciudad

Pos.	Población	Habitantes
1	Madrid	3.266.126
2	Barcelona	1.636.762
3	Valencia	794.288
4	Sevilla	688.592
5	Zaragoza	674.997
6	Málaga	574.654
7	Murcia	453.258
8	Palma	416.065
9	Las Palmas de Gran Canaria	379.925
10	Bilbao	346.843

Cuadro 2.2.: Top 10 ciudades más pobladas de España. Fuente: Revisión del Padrón Municipal. INE (publicado en diciembre de 2019 con los datos a fecha de 01/01/19. Próxima actualización: diciembre de 2020)

2.4.2 Clientes

El conjunto de datos de clientes se compone de 49998 y 8 columnas que se describen a continuación:

age Edad del cliente

customerid Identificador único del cliente

gender Género con el que se identifica el cliente

isocountrycode Código ISO del país del cliente

registrationdate Fecha de alta del cliente

regularstoreid Identificador de la tienda habitual del cliente

unregistrationdate Fecha de baja del cliente

zipcode Código postal del cliente

En el cuadro 2.3 se describe la estadística de los datos sobre clientes proporcionados. De ella se puede extraer: que hay identificadores de clientes repetidos; que sólo el 0.01% se ha dado de baja; que todos pertenecen a España; que hay cinco géneros definidos en los registros; que la mediana y la media de edad es de 45 años. Además de estas conclusiones, aunque se desconoce el código postal del 16% de los clientes se podría asignar el código de postal de la tienda habitual de compra, aunque esa información tampoco está disponible, pero si que se les puede asignar la ciudad de la tienda habitual de compra.

	age	customerid	gender	isocountrycode	registrationdate	regularstoreid	unregistrationdate	zipcode
Registros	44138	49998	44006	49998	49998	49998	7	41889
Media	45.718	-	-	-	-	-	-	-
SD	11.768	-	-	-	-	-	-	-
Min	17.000	-	-	ES	2015-12-17	ES0201	2017-08-31	00260
25 %	38.000	-	-	-	-	-	-	-
50 %	45.000	-	-	-	-	-	-	-
75 %	54.000	-	-	-	-	-	-	-
Max	119.000	-	O	ES	2019-09-01	ES7002	2019-09-03	GX11-1AA
Ausentes (%)	11.72	0.000	11.98	0.0	0.0	0.0	99.99	16.22
Únicos	79	49724	5	1	1139	551	6	4082

Cuadro 2.3.: Estadística descriptivo del conjunto de datos customers

En la imagen 2.2 se descubren los géneros existentes en el conjunto de datos. Se puede observar como de forma mayoritaria los clientes se identifican como hombres y mujeres o no han reportado el género, y tan sólo 122 clientes (0.002%) ha reportado un género distinto.

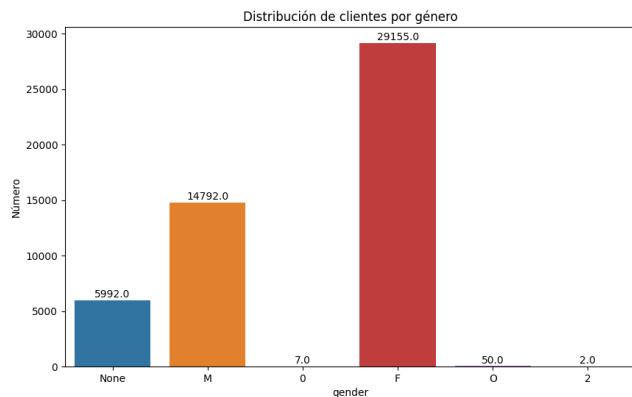
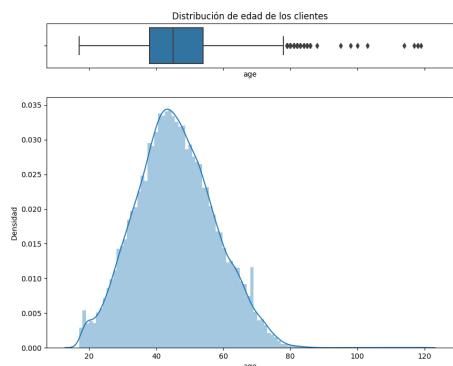
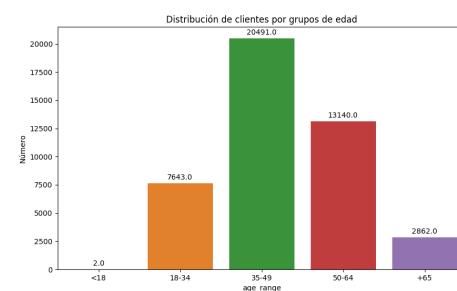


Figura 2.2: Distribución de los clientes por género

En cuanto a la distribución de edad observada en la imagen 2.3a se aprecia su simetría, y se corrobora con el hecho de que la media y la mediana coinciden. También se destaca la presencia de valores anómalos en edades superiores a los 75 años. Adicionalmente, en la imagen 2.3b se han agrupado en los siguientes grupos de edad: Menor de 18, 18-34, 35-49, 50-64, 65+ (Kotler y col., 2004); donde se observa que el grueso de los clientes se sitúa en el rango de 35 a 49 años.



(a) Distribución de clientes según edad



(b) Distribución de clientes según grupos de edad

2.4.3 Productos

En este conjunto de datos se proporciona información relevante de distintos productos, en total 7917 con 17 campos. Todos los campos son categóricos y son los siguientes:

agelimit Indica si se trata de un producto cuya venta solo está permitida para una determinada edad. En este caso los únicos valores son "Keines", del alemán ninguna y 18, e indica si se trata de un producto con contenido alcohólico.

article Descripción del producto

sectionname Jerarquía de sección de producto

familyname Jerarquía de familia de producto

categoryname Jerarquía de categoría de producto

subcategoryname Jerarquía de subcategoría de producto

hwgdescription Algún tipo de categorización de producto

internationalarticlenumber Número internacional del artículo

isocountrycode Código ISO del país del producto

merchandisegroup Jerarquía mercantil de grupo de producto

merchandise subgroup Jerarquía mercantil de subgrupo de producto

merchandisefamily Jerarquía mercantil de familia de producto

productid Identificador único de producto

scaleproduct y weightproduct Tras un análisis se ha deducido que un producto con las dos variables positivas indica que se trata de un producto a granel, mientras que un producto que solo tenga la variable weightproduct positiva indica un producto fresco previamente envasado. Las dos combinaciones restantes no arrojan una conclusión evidente.

vatdescription y vatrate Descripción y porcentaje respectivamente del tipo de impuesto aplicado sobre el producto

En el cuadro 2.4 se proporciona una estadística para la que se contabilizan los valores únicos, el número de registros y el número de valores ausentes. De la misma se puede deducir que no hay valores duplicados y que las columnas correspondientes a sectionname, familyname, categoryname, subcategoryname e internationalarticlenumber son prescindibles debido a su alto porcentaje de valores ausentes. Por lo tanto, los valores que se emplearán para la categorización de productos serán merchandisegroup, merchandise subgroup y merchandisefamily.

	Registros	Ausentes (%)	Únicos
agelimit	7917	0.000	2
article	7917	0.000	7287
categoryname	4473	43.501	238
familyname	4473	43.501	100
hwgdescription	7917	0.000	9
internationalarticlenumber	4374	44.752	4374
isocountrycode	7917	0.000	1
merchandisefamily	7917	0.000	162
merchandisegroup	7917	0.000	7
merchandisesubcategory	7917	0.000	42
productid	7917	0.000	7917
scaleproduct	7917	0.000	2
sectionname	4473	43.501	18
subcategoryname	4473	43.501	348
vatdescription	7917	0.000	5
vatrate	7917	0.000	5
weightproduct	7917	0.000	2

Cuadro 2.4.: Estadística descriptiva del conjunto de datos products

En la imagen 2.4 se presenta la distribución de productos según su pertenencia a la subcategoría.

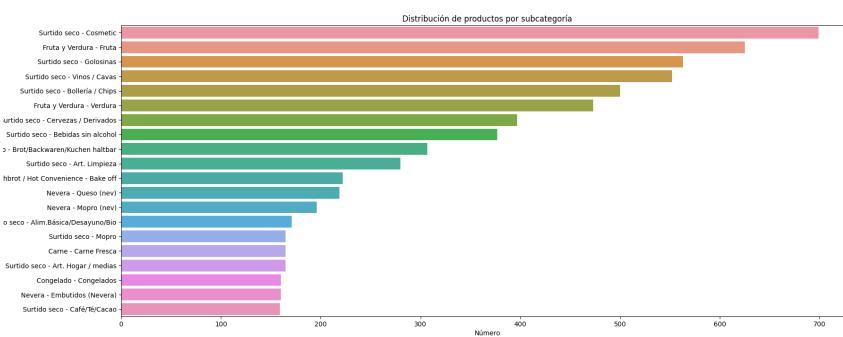


Figura 2.4.: Distribución de los productos por subcategoría

2.4.4 Tickets

El conjunto de datos de tickets se corresponde con las transacciones realizadas por cada cliente. Se compone de: 13,238,241 registros, de los cuales 117,122 son duplicados y se eliminan para realizar la estadística descriptiva; y los siguientes 11 campos:

cardtype Forma de pago

customerid Identificador del cliente que realiza la compra

datekey Fecha de la compra

originalamount Importe de la compra sin descuentos aplicados

totaldiscount Importe del descuento aplicado

extendedamount Importe de la compra con los descuentos aplicados

linenumber Número de linea en el ticket de compra

productid Identificador del producto comprado

quantity Cantidad o peso de productos comprados

storeid Identificador de la tienda donde se ha realizado la compra

ticketid Identificador del ticket de compra

Debido a que la información está particionada por lineas. Se van a realizar una serie de agregaciones para extraer conclusiones más relevantes. Para cada grupo formado por: customerid, storeid, cardtype, datekey, ticketid y productid; se va a sumar la cantidad total de un mismo producto, junto con su coste, original y extendido y el descuento, y adicionalmente se va a eliminar el número de línea pues no aporta información relevante. Así, en el cuadro 2.5 se obtiene información de la que se pueden extraer las siguientes conclusiones:

- Que los registros tienen lugar en 4 meses, de Mayo a Septiembre de 2019.
- Que se han comprado productos cuyo valor es 0.
- Que la cantidad es una distribución totalmente sesgada a la izquierda y el valor máximo queda muy alejado del 3 cuartil. Por lo tanto se tendrá que evaluar su tratamiento.

	Registros	Media	SD	Min	25%	50%	75%	Max	Ausentes (%)	Únicos
customerid	10592796	-	-	-	-	-	-	-	0.000	49729
storeid	10592786	-	-	-	-	-	-	-	0.000	556
cardtype	10592796	-	-	-	-	-	-	-	0.000	5
datekey	10592796	-	-	2019-05-01	-	-	-	2019-09-05	0.000	128
ticketid	10592796	-	-	-	-	-	-	-	0.000	922071
productid	10592796	-	-	-	-	-	-	-	0.000	6561
extendedamount	10592796	2.120	3.549	0.0	0.950	1.470	2.390	1481.75	0.000	7280
originalamount	10592796	2.171	3.611	0.0	0.950	1.490	2.490	1980.0	0.000	5964
totaldiscount	10592796	0.053	0.316	0.0	0.000	0.000	0.000	498.25	0.000	1924
quantity	10592796	1.538	1.986	0.004	1.000	1.000	2.000	1200.0	0.000	12865

Cuadro 2.5.: Estadística descriptiva de tickets agrupados

Tras analizar las líneas de ticket con extendedamount igual a 0, se puede verificar que tan solo suman 2995 registros y que siempre se trata de una unidad correspondiente a dos productos con identificadores 3839 y 3840. Ni la existencia ni la ausencia de estos registros alterará la estadística general del conjunto de tickets de forma significativa.

A continuación se elaborará una estadística descriptiva para conocer mejor que tipo de información se puede extraer este conjunto de datos.

Sobre la forma de pago

De la imagen 2.5 se puede extraer que prácticamente la totalidad de las compras se realizan con Tarjeta Digital(DigitalCard), el 88%, seguida por el pago móvil con un 11%.

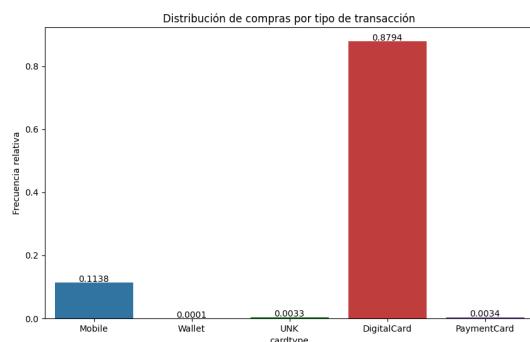
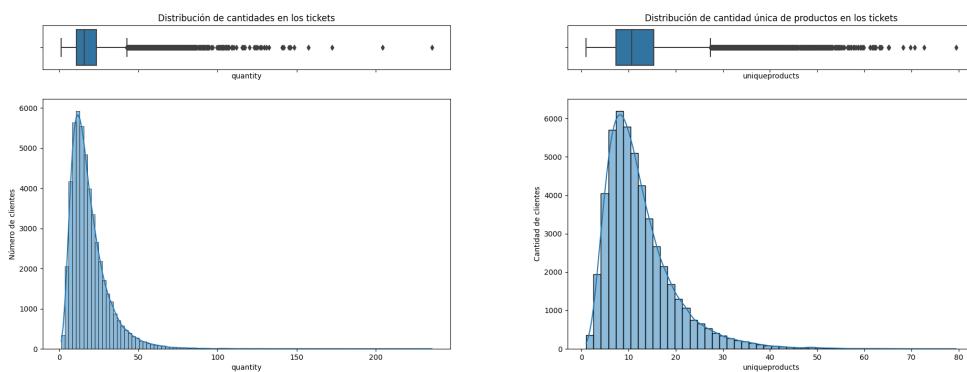


Figura 2.5.: Distribución de las formas de pago en las transacciones

Sobre el volumen de la cesta

En las imágenes 2.6a y 2.6b se observa la distribución por ticket para la cantidad de productos únicos en la cesta y el volumen total de la cesta. Se puede observar la existencia de valores anómalos que merecerían ser estudiados y determinar si deben eliminarse o no. Como es de esperar en ambos casos se trata de una distribución bastante dispersa y sesgada hacia la izquierda.



(a) Distribución de cantidad de productos en el carrito de la compra (b) Distribución del número de productos únicos en el carrito de la compra

La estadística sobre el volumen de la cesta se puede encontrar en la tabla 2.6. Como se puede ver la mayoría de clientes tiende a llenar sus cestas con una cantidad de entre 10 y 23 productos, y entre 7 y 15 productos distintos.

	count	mean	std	min	25%	50%	75%	max
quantity	49729.000	18.924	12.042	1.368	10.815	15.966	23.638	235.384
uniqueproducts	49729.000	12.384	7.291	1.000	7.375	10.647	15.393	79.467

Cuadro 2.6.

2.5 Análisis cruzado

A continuación se realiza un análisis exploratorio sobre la relación entre los distintos conjuntos de datos. Como el objetivo de este trabajo se centra en la segmentación de los hábitos de compra de los clientes, se va analizar la relación del conjunto de datos de tickets junto con la de clientes, productos y tiendas.

2.5.1 Relación entre clientes

De los 49729 clientes únicos existentes en el conjunto de tickets, y los 49998 existentes en el conjunto de clientes tan solo hay un 7.72% de coincidencia. Es decir, que solo se puede obtener la información de género y edad de un porcentaje muy pequeño. Por lo tanto dificultará la segmentación y dificultará el lanzamiento de campañas con base en estos indicadores. Se ha verificado además que la contribución de estos clientes al total de tickets es de tan solo 77400 de 922071, es decir, el 8%.

2.5.2 Relación con productos

De los 6561 productos únicos existentes en el conjunto de tickets, y los 7917 existentes en el conjunto de clientes tan solo hay un 32.92% de coincidencia, y su contribución al total de transacciones es del 34%. Esto dificulta realizar una segmentación basada en el contenido de la cesta de la compra, ya que se desconoce la categoría, el tipo impositivo aplicado al producto, etc., de gran parte de los productos existentes en el carro de la compra.

Segmentación

Como se ha comentado en el caso de uso, la segmentación propuesta se basa en el modelo RFM. Para ello se hará uso únicamente de la información contenida en el conjunto de datos de transacciones. Antes de realizar la segmentación lo primero será calcular los valores de RFM y su puntuación.

3.1 Análisis RFM

El análisis RFM consiste en analizar las siguientes variables calculadas de la siguiente forma:

Recencia Días desde la última vez que se realizó una compra con respecto a la fecha más reciente existente en el conjunto de datos.

Frecuencia Número de veces que se ha realizado una compra

Gasto Suma total de lo que el cliente ha gastado

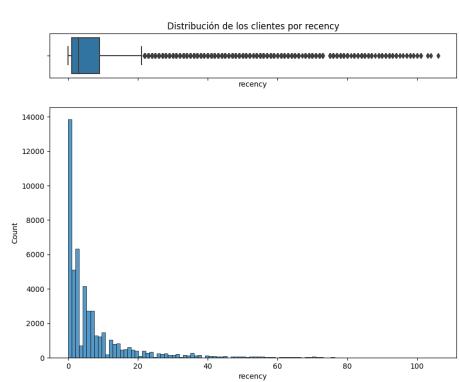
A continuación se realiza un análisis univariable de la recencia, frecuencia y gasto. En todos los casos se ha detectado la existencia de valores anómalos. A continuación se mostrará el resultado de las distribuciones de cada variable antes y después de la eliminación de valores anómalos, para la cual se ha utilizado el método de recorte para un determinado percentil.

3.1.1 Análisis descriptivo

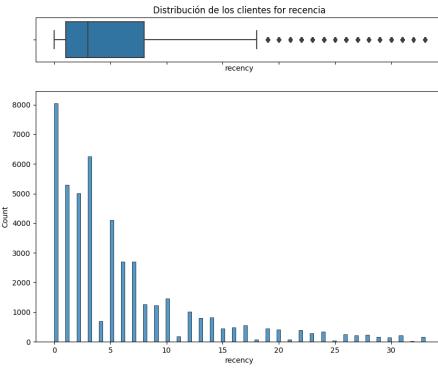
Compra más reciente

En las imágenes 3.3a y 3.3b, que muestran la distribución de clientes por recencia antes y después de la limpieza de valores anómalos, se observa que la muestra está muy sesgada a la derecha y con bastante dispersión. Tras recortar los valores superiores al percentil 95, se consigue menor dispersión aunque se mantiene la existencia de algunos valores atípicos. Gran parte de los consumidores han ido en

los últimos días al supermercado aunque una parte importante no han ido en la última semana.



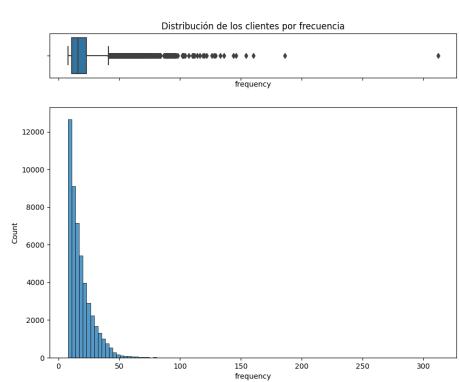
(a) Distribución de la compra más reciente



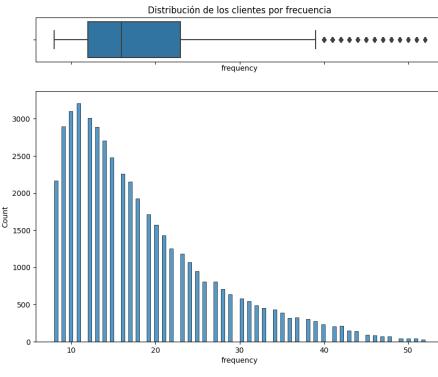
(b) Distribución de la compra más reciente recortada

Frecuencia de compra

Al igual que en el caso anterior, se aprecia con una distribución asimétrica, unimodal y sesgada a la derecha. Todos los clientes han ido repetidas veces al supermercado en los cuatro meses de muestra analizados. Como en el caso anterior tras recortar, en este caso el percentil 99 se consigue una distribución mucho menos dispersa.



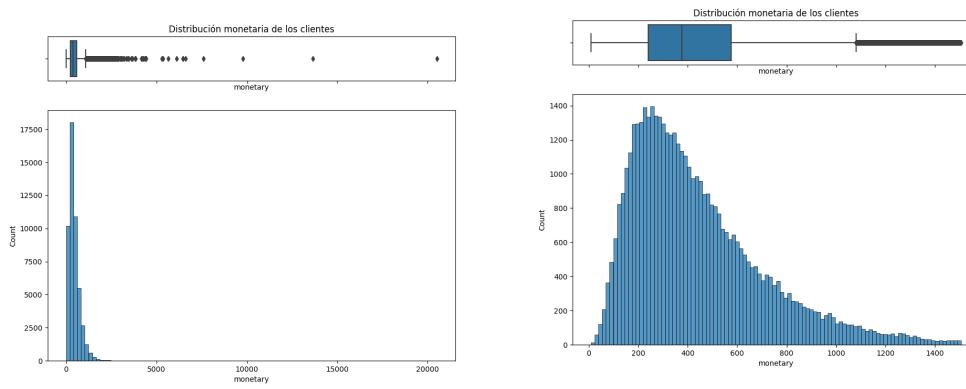
(a) Distribución de la frecuencia de compra



(b) Distribución de la frecuencia de compra recortada

Gasto

En el gasto nuevamente aparece una distribución muy sesgada a la derecha y con dispersión. Se perciben pocos valores atípicos en la cola de la distribución muy separados entre sí. Tras aplicar un recorte en el percentil 99 de la disitribución, se consigue reducir dicha dispersión. La mayor parte de los clientes han gastado entre 250 y 600 euros en cuatro meses aunque también se aprecia un alto porcentaje que ha gastado bastante por encima de la mediana.



(a) Distribución de la frecuencia de compra

(b) Distribución de la frecuencia de compra recortada

3.1.2 Conclusión RFM

La eliminación de valores anómalos en los percentiles elegidos no tiene un gran efecto sobre la distribución de las variables (Ver cuadro 3.1). Sin embargo se pueden elegir valores más agresivos que modifiquen de forma significativa los umbrales para los cuales se apliquen las distintas puntuaciones.

	En bruto			Recortada		
	Recencia	Frecuencia	Gasto	Recencia	Frecuencia	Gasto
25%	1	11	237.20	1	12	240.41
50%	3	16	374.43	3	16	376.77
75%	9	23	579.81	7	23	576.32

Cuadro 3.1.: Cuartiles de la distribución RFM con y sin valores anómalos

Los valores de cuartiles obtenidos de la distribución RFM recortada se aplicarán como umbrales al total del conjunto de transacciones. Tras aplicar estas transformaciones al conjunto de datos se obtienen las puntuaciones RFM. Como se ha decidido utilizar cuartiles en lugar de quintiles, la puntuación RFM estará en el rango de 111 a 444 (Ver imagen 3.4). Esto hace un total de 64 grupos distintos. Aunque este valor puede resultar útil y se puede tratar de establecer una agrupación de forma manual en base a estos valores, se ha decidido hacer uso del algoritmo de K-Means para conseguir las agrupaciones.

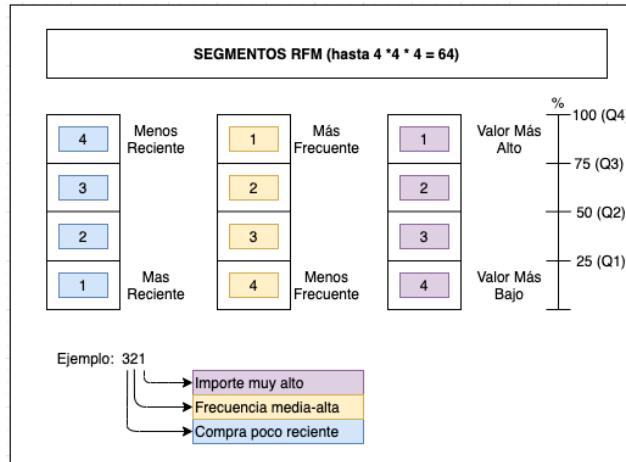


Figura 3.4.: Asignación de puntos RFM según cuartil

En la imagen 3.5 se observan los distintos grupos RFM según su puntuación ordenados de mayor a menor según el número de clientes con dicha puntuación. Se observa que los grupos más numerosos son los extremos 111 y 444.

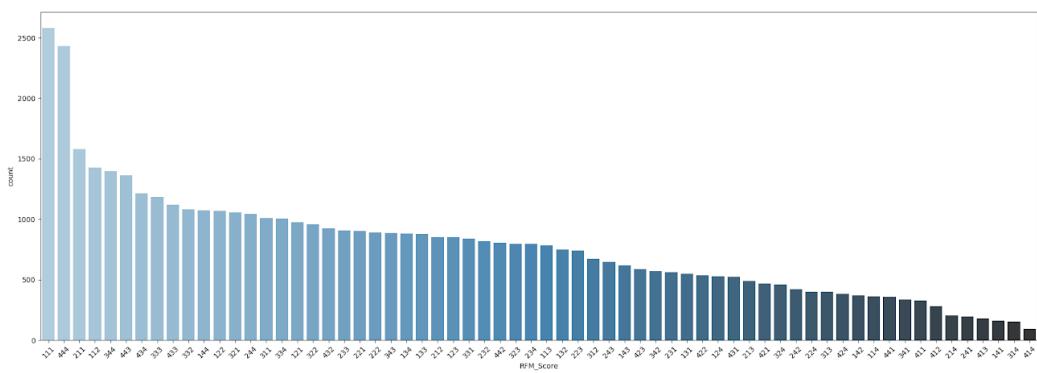


Figura 3.5.: Distribución de los clientes según puntuación RFM

3.2 Agrupación con K-Means

El algoritmo de K-Means es un algoritmo de clasificación no supervisada (clusterización) que toma como entradas parámetros y el número de grupos, y agrupa los datos en el número definido de cúmulos tal que la similitud intra-grupo sea alta. K-Means es una aproximación iterativa que computa el valor de los centroides cada iteración. Los puntos de los datos se mueven entre los diferentes grupos dependiendo del centroide calculado en cada iteración. El proceso se repite hasta que la suma ya no decrece. Los valores de las variables recencia, frecuencia y gasto se estandarizan, es decir se sustrae la media y se divide por la varianza quedando distribuciones con media 0 y varianza 1. Esto es necesario pues K-Means es isotrópico en todas las direcciones del espacio y por tanto tiende a producir grupos más circulares que ovalados. Dejar las varianzas distintas, equivale a dar más peso a las variables con menor varianza.

Una de las desventajas de K-Means es que inicializa los centroides de forma aleatoria. El problema de elegirlos de esta manera, es que los centroides pueden agruparse cerca entre sí lo que hace que los grupos sean menos significativos. Elegir los centroides iniciales determina cómo de buena sea la agrupación así como reducir el número de iteraciones, las soluciones óptimas globales y cómo de compacto sea el grupo (Christy, Umamakeswari, Priyatharsini & Neyaa, 2018, p. 4).

Es por esto que se decide utilizar la aproximación propuesta por Christy y col., 2018 en lo que denomina Repetitive Median K-Means. Este método consiste en ordenar cada uno de los vectores R, F y M de forma ascendente, particionarlos en K segmentos de igual tamaño y calcular la mediana de cada segmento. Con ello quedan determinados los K centroides iniciales con el que inicializar el algoritmo de K-Means. Según Christy y col., 2018, las agrupaciones obtenidas por este método son más significativas respecto a elegir los centroides de forma aleatoria.

Otra de las bondades de este método, es que la asignación del índice para las agrupaciones se mantiene, por lo que no es necesario reinterpretar los segmentos.

3.2.1 Elección del número de segmentos

Antes de incorporar la inicialización de centroides al algoritmo de K-Means, se había entrenado el algoritmo con una inicialización aleatoria de los mismos para diferente número de agrupaciones. De esta manera se puede obtener el WSSSE (Within Set

Sum of Squared Errors) que cuantifica cómo de pequeño es el error de la minimización de distancia entre los elementos de un grupo, que es el objetivo de K-Means. Cuanto menor es el error, mejor es la agrupación. En la figura 3.6 se observa que para una agrupación de 6 aparece el primer codo. Se podrían elegir agrupaciones con un menor WSSSE pero con ello aumentaría la dificultad para interpretar los segmentos.

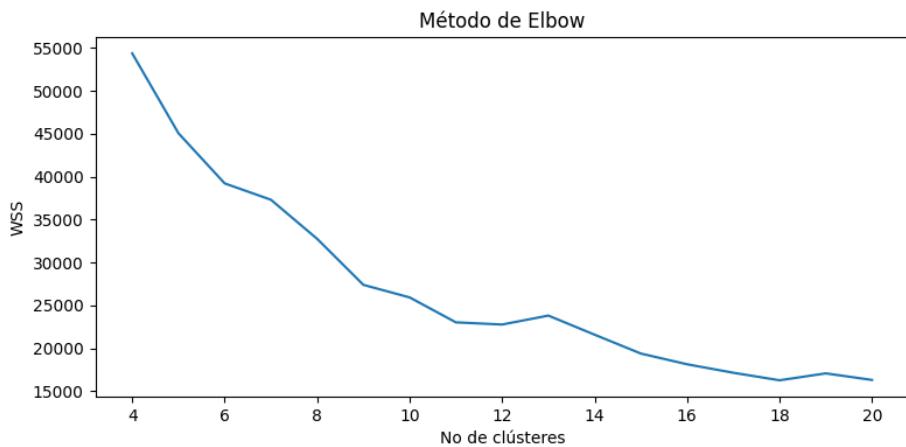


Figura 3.6.: Método de Elbow para centroides aleatorios

Tras detectar que utilizar centroides aleatorios era problemático, ya que aunque los grupos son similares, el orden asignado y la etiqueta asignada son distintos en cada ejecución. Cada vez que se reentrenase el algoritmo, habría que reinterpretar las agrupaciones. Para solucionar este problema se ha implementado la inicialización de centroides en base a las medianas propuesta por Christy y col., 2018. Tras entrenar el algoritmo incializando los centroides para distinto número de agrupaciones, se observó que el codo en 6 desaparecía y aparecía en 4 y 8. Ante este hecho, se decidió igualmente continuar con una segmentación de 6 grupos, principalmente por que ya se habían identificado y porque con la inicialización, la segmentación resultante que se verá a continuación era menos ambigua que la obtenida sin inicialización de centroides.

3.2.2 Interpretación de los segmentos

A continuación se presentan los resultados de la segmentación. En la imagen 3.7 se observa un gráfico de dispersión con los clientes donde se visualizan el gasto

y la frecuencia de compra en los ejes X e Y respectivamente. El tamaño del punto representa la recencia, cuanto más grande, mayor es el tiempo que ha pasado desde la última vez que compraron. Y el color y la forma representan el segmento asignado.

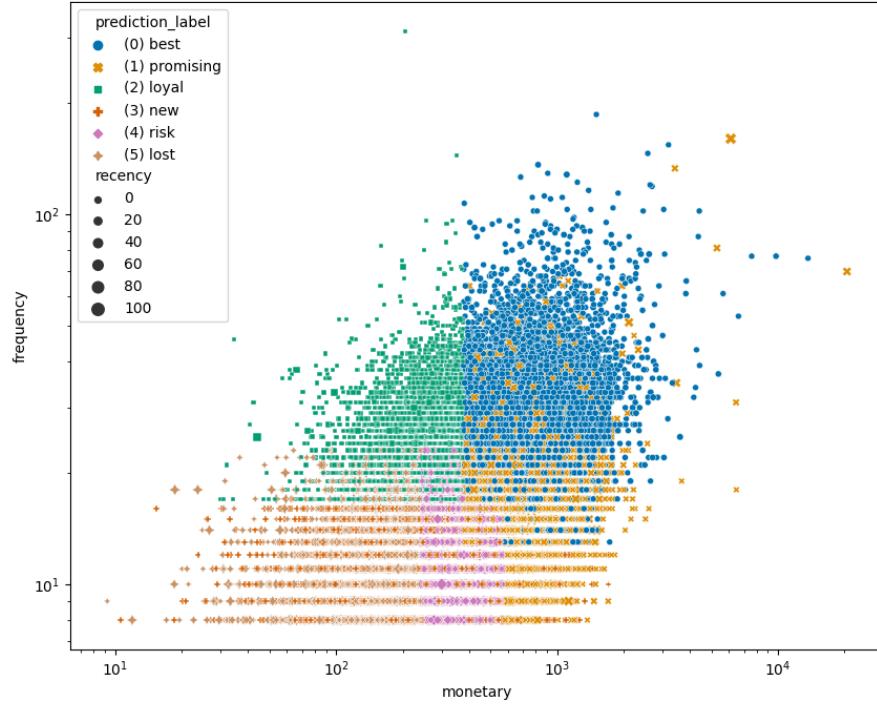


Figura 3.7: Gráfico de dispersión con segmentos

Para entender mejor cuáles son los criterios de la agrupación, en la imagen 3.8 se muestra la importancia relativa de los atributos. En ella se puede ver como se valora cada atributo respecto a la media (0), es decir, un valor positivo y alejado de cero significa que en ese grupo se encontrarán clientes con valores por encima de la media global de ese atributo, y viceversa.

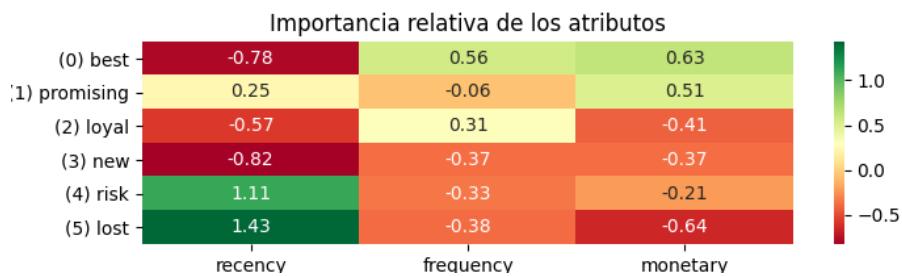


Figura 3.8.: Gráfico con la importancia relativa de las variables para cada grupo

En el cuadro 3.2 se realiza una explicación de cada uno de los grupos identificados así como su denominación.

0	24.38	39.8	$\downarrow R \uparrow F \uparrow M$	Best	Han venido hace muy poco, vienen mucho y gastan mucho
1	16.16	24.38	$\uparrow R F \uparrow M$	Promising	Llevan tiempo sin venir, venian poco aunque alguno solia venir bastante y gastan por encima de la media
2	12.63	7.44	$\downarrow R \uparrow F \downarrow M$	Loyal	Han venido hace poco, vienen por encima de la media y gastan por debajo de la media
3	17.66	11.11	$\downarrow R \downarrow F \downarrow M$	New	Han venido hace muy poco, vienen por debajo de la media y gastan por debajo de la media.
4	15.72	12.40	$\uparrow R \downarrow F M$	Risk	Llevan tiempo sin venir, vienen por debajo de la media y gastan por debajo de la media.
5	13.45	4.85	$\uparrow R \downarrow F \downarrow M$	Lost	Llevan mucho tiempo sin venir, venian poco y gastaban poco.

Cuadro 3.2.: Descripción de los segmentos

Resultados

En este capítulo se exploran los diferentes segmentos obtenidos. Para ello además se ha elaborado una presentación para que un departamento de marketing sea capaz de interpretar los resultados obtenidos y planificar campañas para los diferentes segmentos de clientes. A continuación se describen las características más relevantes que se pueden extraer de la presentación Marketing Insights (Anexo A.2). Para este apartado se ha incorporado información externa generada manualmente para categorizar los productos según su categoría en saludables y no saludables. Se ha realizado dicha incorporación para suplir la imposibilidad de integración con fuentes externas que pudieran proveer este dato.

4.0.1 Segmento: BEST

Quiénes son

Clientes altamente comprometidos que han comprado lo más reciente, con mayor frecuencia, y han generado la mayor cantidad de ingresos.

Características Importantes

- Engloban el 24% de los consumidores y aportan el 40% del total de los ingresos.
- Compuesto por adultos de unos 47 años de media, en su mayoría mujeres.
- Hacen compras de tamaño medio.
- Apuestan por tendencias saludables, por encima del 60% de sus productos comprados se clasifican como saludables.
- Alrededor del 24% de sus productos tienen descuento, aunque demuestran tener una mayor disposición a pagar que el resto de los grupos.

Estrategia de Marketing

Nuestra estrategia de marketing se debería centrar en premiarlos, por ejemplo, con programas de lealtad para retenerlos, y que continúen con su actividad.

4.0.2 Segmento: PROMISING

Quiénes son

Clientes del pasado que solían venir con frecuencia y que gastaron cantidades importantes de dinero.

Características Importantes

- Agrupan al 16% de nuestros clientes y tienen un gran potencial debido a que aportan una cantidad considerable de nuestros ingresos, el 24%.
- Compuesto mayoritariamente por adultos de 46 años media, en su mayoría mujeres
- Hacen compras de tamaño medio.
- Se sitúan por debajo de la media en cuanto a compra de productos con descuento, por lo que los descuentos no son un factor significativo para los clientes de este grupo.

Estrategia de Marketing

Sería interesante aplicar una estrategia de marketing que se centre en construir una relación de fidelidad aumentando así su número de visitas y les permita evolucionar al grupo de BEST.

4.0.3 Segmento: LOYAL

Quiénes son

Clientes asiduos, pero que no gastan mucho.

Características Importantes

- Agrupan al 12% de nuestros clientes y aportan sólo el 7% de nuestros ingresos
- Compuesto por adultos de unos 47 años de media, en su mayor parte mujeres. Es el grupo con más clientes masculinos, aproximadamente el 40%.
- Hacen compras muy pequeñas.
- Es el grupo de clientes con hábitos de compra menos saludables, el 40% de sus productos son clasificados como no saludables.
- Los descuentos son un factor importante en la elección de productos para los consumidores de este grupo, el 25% de sus productos tienen descuento.

Estrategia de Marketing

Habría que centrarse en aumentar la monetización, por ejemplo, a través de recomendaciones de productos en compras pasadas e incentivos relacionados con los umbrales de gasto.

4.0.4 Segmento: NEW

Quiénes son

Compradores que han visitado el supermercado recientemente pero que no habían venido con anterioridad.

Características Importantes

- Engloban el 17% de nuestros clientes y suponen un 11% de nuestros ingresos.
- Compuesto por adultos con una edad promedio de 44 años, la edad media más baja de todos los segmentos, y con el mayor porcentaje de mujeres, el 67%.
- Hacen compras de gran tamaño.
- Es el grupo de clientes con hábitos de compra más saludables, casi el 65% de sus productos son saludables y con un menor porcentaje de productos no saludables, el 30%.
- En cuanto a la compra de productos con descuento, se sitúan en la media, con el 18% de sus productos con descuento.

Estrategia de Marketing

Es importante incentivar su número de visitas para que lleguen a convertirse en LOYAL, de lo contrario, podrían pasar al grupo de riesgo siendo más complicada su retención.

4.0.5 Segmento: RISK

Quiénes son

Antiguos compradores que llevan tiempo sin venir, gastan la media y es probable que se hayan decidido por algún otro competidor.

Características Importantes

- Agrupan el 15% de nuestros clientes, y dado que aportan el 12% de nuestros ingresos sería conveniente recuperarlos.
- Compuesto por adultos de unos 47 años de media, en su mayor parte mujeres.
- Hacen compras muy pequeñas.
- Se sitúan por debajo de la media en cuanto a la compra de productos con descuento, por lo que no es un factor significativo para este grupo.

Estrategia de Marketing

Campañas personalizadas para ayudar a reconectar.

4.0.6 Segmento: LOST

Quiénes son

Clientes del pasado que no han comprado en un periodo muy largo de tiempo.

Características Importantes

- Engloban el 13% de los consumidores y aportan menos del 5% del total de los ingresos.
- Compuesto por adultos de unos 45 años de media, en su mayoría mujeres
- Segundos consumidores de productos no saludables.
- Hacen compras pequeñas.
- Se sitúan en la media en cuanto a compra de productos con descuento, por lo que los descuentos no son un factor significativo para este grupo.

Estrategia de Marketing

No es prioritario apostar por una estrategia de marketing para intentar recuperarlos debido al poco valor monetario que aportan (< 5%).

Despliegue

En este capítulo se van a describir los pasos necesarios para realizar la segmentación, tanto entrenamiento como segmentación de forma desatendida. Primero se darán las instrucciones que se han utilizado por nuestra parte a modo de ejemplo, pero se deberán adaptar a las condiciones específicas del usuario interesado en ejecutar dichas tareas. Ambas tareas se han desarrollado en Python utilizando las librerías de PySpark para la ejecución de tareas en un clúster. El clúster utilizado será uno creado en Amazon EMR¹ y las tareas serán encoladas mediante comandos de AWS CLI².

5.1 Creación del clúster

Para la creación del cluster basta con ejecutar el comando `create-cluster` posteriormente al comando `create-default-roles` en caso de que los roles no hubieran sido creados. La salida del comando `create-cluster` proporcionará el identificador del clúster que deberá ser utilizado en los siguientes pasos 5.2.

List. 5.1: Comando AWS CLI para crear cluster

```
1 #!/bin/bash
aws emr create-default-roles

aws emr create-cluster \
--release-label emr-5.30.0 \
--instance-type m5.xlarge --instance-count 2 --applications Name=Spark Name=Hadoop \
--use-default-roles \
--scale-down-behavior TERMINATE_AT_TASK_COMPLETION \
--region us-east-1
```

List. 5.2: Comando AWS CLI para crear cluster

```
1 {
  "ClusterId": "j-XXXXXXXXXXXX",
  "ClusterArn": "arn:aws:elasticmapreduce:us-east-1:XXXXXXXXXXXX:cluster/j-XXXXXXXXXXXX"
}
```

¹<https://aws.amazon.com/es/emr>

²<https://aws.amazon.com/es/cli/>

5.2 Ejecución de tareas en el cluster

La ejecución de tareas en el clúster se realiza mediante la instrucción add-steps. En el listado 5.3 se muestra la instrucción en donde el parámetro -steps indica un fichero JSON con la definición de la tarea a ejecutar.

List. 5.3: Comando AWS CLI para añadir tareas

```
1#!/bin/bash
aws emr add-steps --cluster-id j-XXXXXXXXXXXXX --steps file://./steps.json --region us-
east-1
```

5.3 Entrenamiento del algoritmo

En el capítulo anterior se ha llevado a cabo toda la fase de análisis sobre los datos proporcionados y se han obtenidos los segmentos y los parámetros de interés para realizar la segmentación. En el anexo A.3.1 se encuentra el código para realizar la segmentación en base a 6 grupos. Dicho código debe estar alojado en algún cubo de Amazon S3, en este caso, se encuentra en:

s3a://mbde-tfm-grupo3/train-segmentation.py.

En el fragmento 5.4 se define la tarea a ejecutar así como los parámetros del programa. Para este programa se deben definir:

-input Cubo S3 dónde se encuentran las transacciones

-model-output Cubo S3 donde almacenar el modelo generado

-prediction-output Cubo de S3 donde almacenar los resultados, se crean: un directorio all con las transacciones segmentadas y 6 directorios adicionales con el nombre del segmento y la información de cada uno de los segmentos por separado.

-r-percentile Corte de recencia en percentil sobre el que descartar valores anómalos para el cálculo de los cuartiles.

-f-percentile Corte de frecuencia en percentil sobre el que descartar valores anómalos para el cálculo de los cuartiles.

-m-percentile Corte monetario en percentil sobre el que descartar valores anómalos para el cálculo de los cuartiles.

-quartile-output Cubo de S3 donde almacenar los valores que se utilizan para puntuar el RFM

Es necesario que al menos directorio de salida para el modelo no exista o si no la ejecución del script fallará. Por ello, es recomendable, previo a la ejecución del comando add-steps ejecutar la siguiente instrucción:

```
aws s3 rm s3://mbde-tfm-grupo3/kmeans_model -recursive
```

List. 5.4: Archivo train-step.json

```
1 [  
2   {  
3     "Name": "Train segmentation",  
4     "Args": ["--deploy-mode", "cluster",  
5               "--master", "yarn",  
6               "--conf", "spark.yarn.submit.waitAppCompletion=true",  
7               "s3a://mbde-tfm-grupo3/train-segmentation.py",  
8               "--input", "s3a://tfmbigdata/Files_Segunda_Entrega/tickets_*.json",  
9               "--model-output", "s3a://mbde-tfm-grupo3/kmeans_model",  
10              "--prediction-output", "s3a://mbde-tfm-grupo3/train_output",  
11              "--quartile-output", "s3a://mbde-tfm-grupo3/quartiles",  
12              "--r-percentile", "0.95",  
13              "--f-percentile", "0.99",  
14              "--m-percentile", "0.99"],  
15     "ActionOnFailure": "CONTINUE",  
16     "Type": "spark"  
17   }  
18 ]
```

Una vez finalizado los resultados se pueden encontrar en el directorio definido en el parámetro **-prediction-output** en formato JSON.

5.4 Ejecución a demanda del modelo entrenado

En el caso de que se quiera volver a ejecutar la segmentación sobre la totalidad, una parte o un nuevo conjunto de transacciones, no es necesario volver a ejecutar la fase de entrenamiento. El código del Anexo A.3.2 realiza la misma segmentación que la ejecución anterior y acepta los siguientes parámetros:

-input Cubo S3 dónde se encuentran las transacciones

-model-input Cubo S3 donde está almacenado el modelo del entrenamiento

-prediction-output Cubo de S3 donde almacenar los resultados, se crean: un directorio all con las transacciones segmentadas y 6 directorios adicionales con

el nombre del segmento y la información de cada uno de los segmentos por separado.

-quantile-input Cubo de S3 donde están almacenados los valores que se utilizan para puntuar el RFM

En el listado 5.5 se encuentra la definición de la tarea a ejecutar así como los parámetros del programa.

List. 5.5: Archivo run-step.json

```
1 [  
2 {  
3     "Name": "Run segmentation",  
4     "Args": ["--deploy-mode","cluster",  
5               "--master","yarn",  
6               "--conf","spark.yarn.submit.waitAppCompletion=true",  
7               "s3a://mbde-tfm-grupo3/run-segmentation.py",  
8               "--input", "s3a://tfmbigdata/Files_Segunda_Entrega/tickets_10.json",  
9               "--model-input", "s3a://mbde-tfm-grupo3/kmeans_model",  
10              "--prediction-output", "s3a://mbde-tfm-grupo3/run_output",  
11              "--quartile-input", "s3a://mbde-tfm-grupo3/quartiles"],  
12              "ActionOnFailure": "CONTINUE",  
13              "Type": "spark"  
14 }  
15 ]
```

Una vez finalizado los resultados se pueden encontrar en el directorio definido en el parámetro -prediction-output en formato JSON.

Conclusiones y líneas futuras

6.1 Conclusiones

En este trabajo se ha realizado un estudio de segmentación basado en el análisis RFM de los hábitos de compra. Debido a la cantidad de volumen de datos proporcionado, el análisis y segmentación se han realizado utilizando herramientas de Big Data como son Amazon S3, Amazon EMR y PySpark. Se han analizado completamente los datos proporcionados mediante Jupyter Notebooks y se ha tratado de determinar qué hacer con los valores anómalos. Se ha realizado el análisis RFM, y se ha creado la segmentación en 6 grupos utilizando el algoritmo RM K-Means (Christy y col., 2018). Con ello se han identificado diferentes perfiles de clientes para los que se ha analizado el comportamiento de cada uno de ellos con el fin de que el departamento de marketing pueda utilizar los resultados obtenidos. Por último se ha facilitado el código y las instrucciones de despliegue para realizar la segmentación de forma desatendida en un clúster de Amazon EMR mediante AWS CLI.

Aunque habría sido interesante poder realizar segmentaciones más allá del análisis RFM y basados en el contenido de la cesta de la compra o las condiciones demográficas y psicográficas, debido a la poca relación existente entre los conjuntos de datos de productos y clientes con el de tickets, los resultados obtenidos en estos términos no pueden tomarse como representativos.

6.2 Limitaciones y trabajo futuro

6.2.1 Limitaciones

Para la realización de este trabajo se ha dispuesto de datos proporcionados de la empresa en dos fases. La primera, de la que no se podían obtener resultados concluyentes sobre la descripción de los datos, puesto que representan un porcentaje muy pequeño del total ha servido para entender la tipología de los mismos y poder plantear estrategias de procesado y transformación de los mismos. También sirve para

detectar en una primera vuelta carencias o errores en los mismos. Con la segunda entrega de los datos, en la que se hacía necesaria la utilización de herramientas de Big Data, ya se pueden implementar y validar las estrategias planteadas en la primera entrega.

En relación a la detección de carencias y planteamiento de estrategias, hay que destacar que una de las premisas del trabajo era la incorporación de una fuente externa de datos para enriquecer los datos proporcionados. Aunque se comentaba la opción de Twitter, no se ha visto una integración clara de esa fuente de datos para los objetivos de este trabajo con los datos proporcionados. Se planteo al proveedor la incorporación al conjunto de datos del código de barras EAN-13 para cruzar los productos con la información de OpenFoodFacts¹. OpenFoodFacts es un sitio web colaborativo donde las personas pueden incorporar productos y describirlos de forma estructurada. Uno de los identificadores que usa el sitio es el EAN-13 (número de artículo europeo) que es un sistema de códigos de barras adoptado por más de cien países y cerca de un millón de empresas. Por tanto conociendo el EAN-13 de los productos proporcionados, un dato público que no compromete la identidad de los usuarios, podría obtenerse información muy relevante sobre el tipo de productos que compra un usuario no solo en base a la categoría de la empresa, si no a la categorización ofrecida por OpenFoodFacts, el Nutriscore², el grupo NOVA (grado de procesado del alimento)³ y su contenido, entre otros. Lamentablemente, dicha información no ha podido ser provista para este análisis aunque sería muy recomendable de cara a futuros trabajos.

Otra de las limitaciones detectadas para la obtención de resultados es la escasa coincidencia entre los datos de transacciones y los datos de clientes y productos. Aunque sí se ha podido obtener una visión global de los hábitos de compra de los clientes en cuanto a su recencia, frecuencia, gasto, y volumen de la cesta de la compra de cada segmento, los resultados obtenidos para los indicadores de las variables sociodemográficas, psicosociales o geográficas pueden no ser representativos.

6.2.2 Trabajo futuro

En línea con las limitaciones comentadas en la sección anterior, explorar como integrar la información disponible de OpenFoodFacts podría arrojar resultados muy interesantes de cara a enfocar campañas de recomendación de productos en base a

¹<https://world.openfoodfacts.org/data>

²<https://world.openfoodfacts.org/nutriscore>

³<https://world.openfoodfacts.org/nova>

características subyacentes en la tipología y contenido de los productos de la cesta de la compra. Se pueden pensar diferentes estrategias como filtrado colaborativo o filtrado basado en contenido, para cada uno de los segmentos.

También, disponer de mayor información sociodemográfica de mayor cantidad de usuarios sería deseable, aunque hay que tener en cuenta que parte de esta información puede caducar y no representar la realidad para un largo periodo de tiempo.

De cara a una mejor segmentación basada en el contenido de la cesta de la compra, dejando al margen el enriquecimiento con fuentes externas, sería disponer de la información específica de cada artículo existente en los transacciones.

En este trabajo se ha analizado la información de 4 meses de transacciones para 50.000 clientes aproximadamente. De este análisis se ha obtenido una segmentación que abarca la totalidad del cuatrimestre. De cara a obtener unos indicadores que ayuden a comprender la evolución de los clientes, sería interesante obtener una evolución mensual, o del período de interés, de los segmentos. De esta manera se podría detectar cuando los clientes están modificando sus hábitos de compra y validar que determinadas campañas están teniendo un efecto positivo o negativo, o poder dirigir promociones específicas a usuarios que estén cambiando sus hábitos de compra ya sea recompensando su fidelidad o tratando de recuperar clientes en riesgo.

Por último, se propone trabajar estrechamente con el departamento de marketing una vez presentados los resultados para incorporar sus posibles necesidades o solventar posibles carencias del modelo propuesto. Y, además, tras plantear posibles campañas de marketing realizar la toma de requisitos para poder desplegar los sistemas adecuados de cara a su utilización de forma regular.

Bibliografía

- Christy, A. J., Umamakeswari, A., Priyatharsini, L. & Neyaa, A. (2018). RFM ranking An effective approach to customer segmentation. *Journal of King Saud University - Computer and Information Sciences*.
- Kotler, P., Armstrong, G., Rodríguez, C., Ibáñez, D. & Roche, I. (2004). *Marketing* (10a. ed.) Pearson, Prentice Hall.
- Miglautsch, J. R. (2000). Thoughts on RFM scoring. *Journal of Database Marketing & Customer Strategy Management*, 8(1), 67-72.
- Srivastava, R. (2016). Identification of Customer Clusters using RFM Model:A Case of Diverse Purchaser Classification. *International journal of business*, 4, 45-50.

Apéndice

A.1 Integración Notebook EMR con GIT

A continuación se detallan los pasos necesarios para poder vincular un repositorio de Git con un cuaderno Jupyter para la ejecución en vivo de instrucciones en un clúster de Amazon EMR.

Para conseguirlo es necesario crear y configurar un entorno apropiado consistente en una VPC con subredes pública y privada y salida a internet. En esta ocasión se realiza mediante una NAT Gateway cuyo precio es elevado pero no requiere de ningún tipo de configuración adicional. La otra opción es mediante una instancia NAT, que es mucho más barata pero requiere de configuración adicional. Si no se desea incurrir en gasto la opción más sencilla es subir los archivos al entorno de Jupyter creado de manera clásica y omitir este tutorial.

A.1.1 Configuración de una VPC

Lo primero es crear una IP elástica para configurar la salida a internet desde la subred privada.

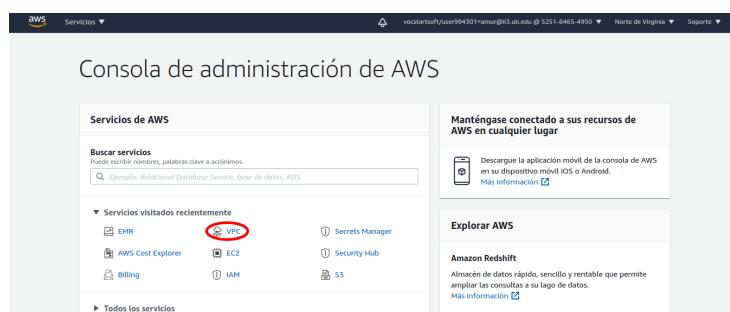


Figura A.1.: Ir a la sección VPC

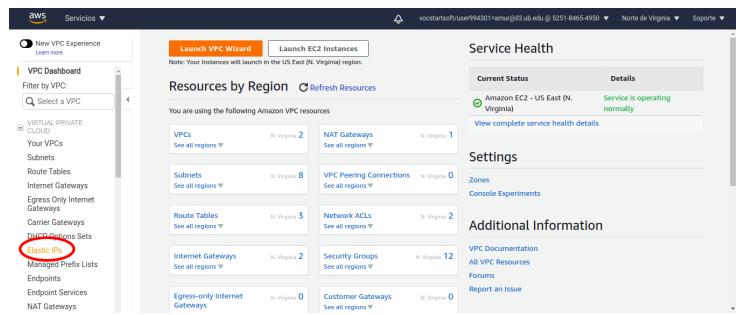


Figura A.2.: Ir a sección de IP elástica

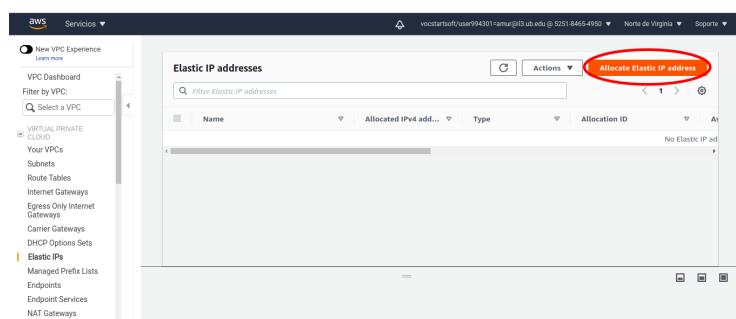


Figura A.3.: Crear IP elástica

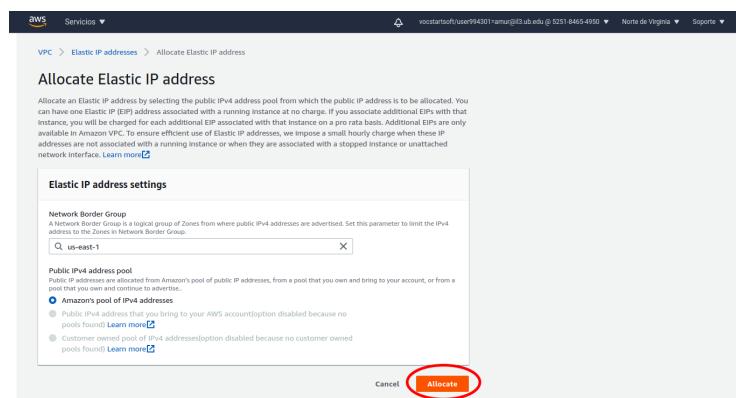


Figura A.4.: Confirmar creación de IP elástica

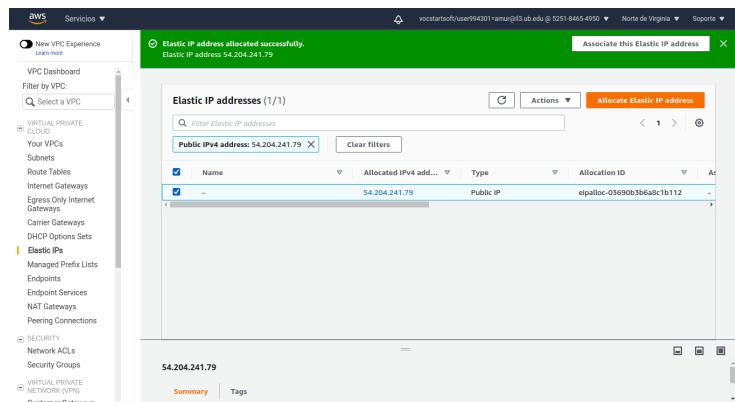


Figura A.5.: Confirmación de la creación de la IP elástica

Volver a la sección de VPC y lanzar el asistente.

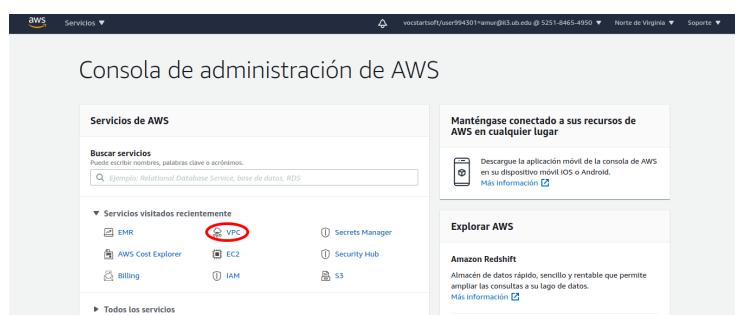


Figura A.6.: Ir a la sección VPC

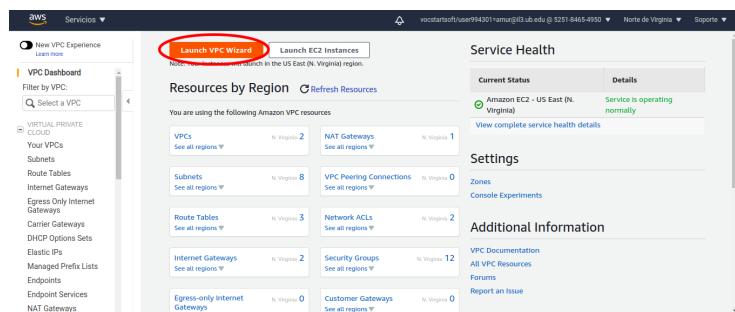


Figura A.7.: Lanzar asistente de creación de VPC

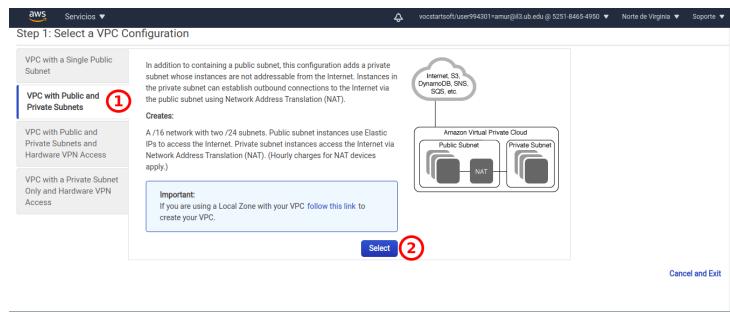


Figura A.8.: Configurar VPC

En la siguiente pantalla asegurarse de configurar las zonas de disponibilidad y seleccionar la IP elástica creada anteriormente.

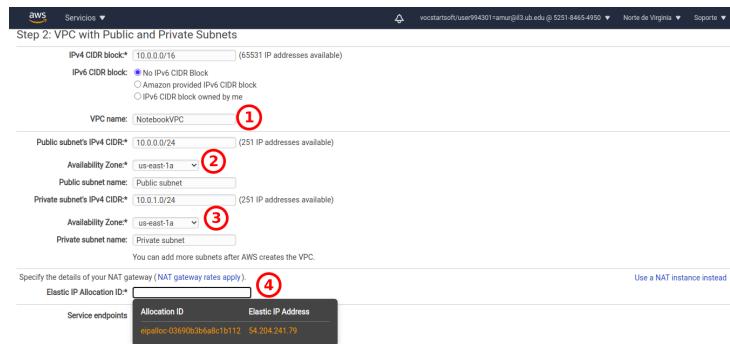


Figura A.9.: Editar la configuración de la VPC

A continuación añadir un punto de conexión con S3 con la configuración por defecto.

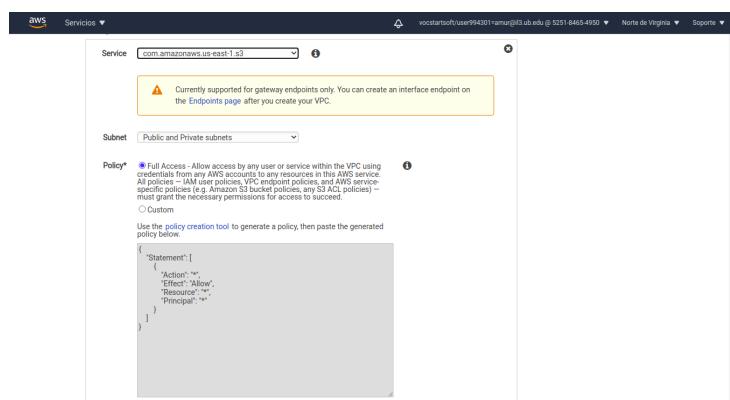


Figura A.10.: Configurar endpoint S3

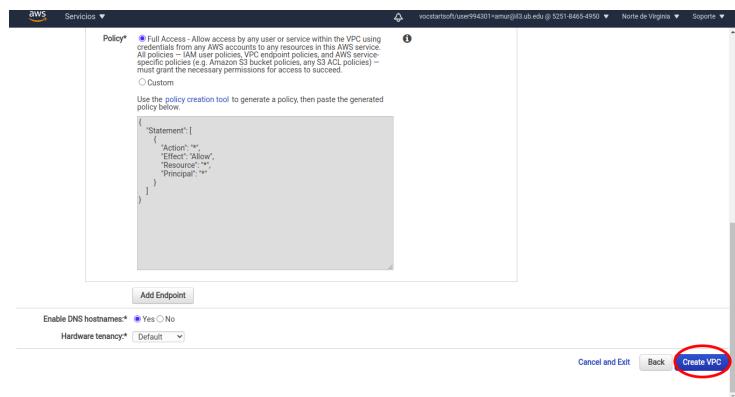


Figura A.11: Creación de VPC

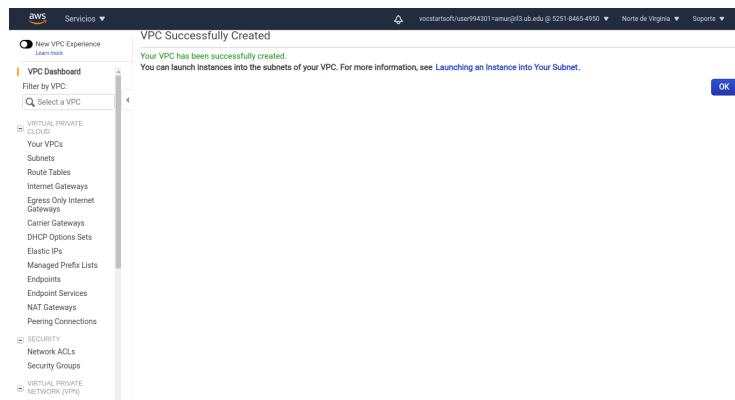


Figura A.12: Confirmación de la creación de la VPC

A.1.2 Creación del clúster EMR

Una vez configurada la red VPC, lo siguiente es crear el clúster EMR.

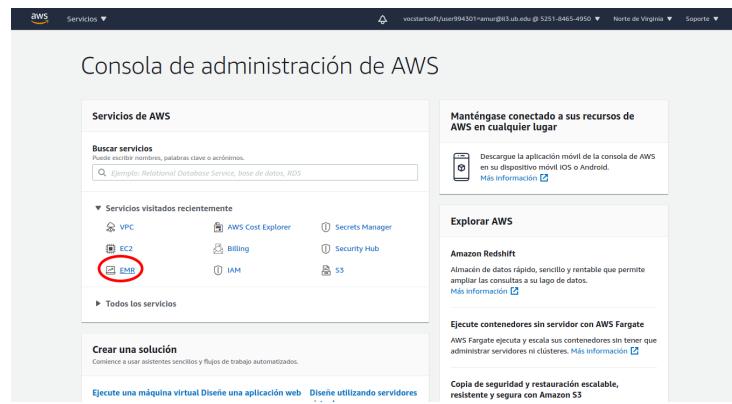


Figura A.13.: Ir la sección de EMR

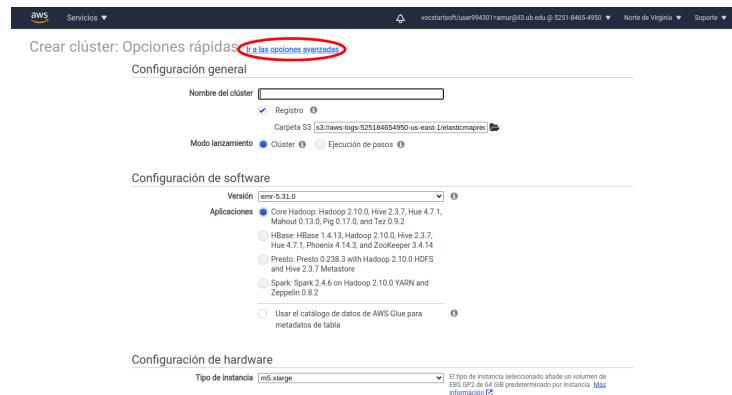


Figura A.14.: Ir a opciones avanzadas de creación EMR

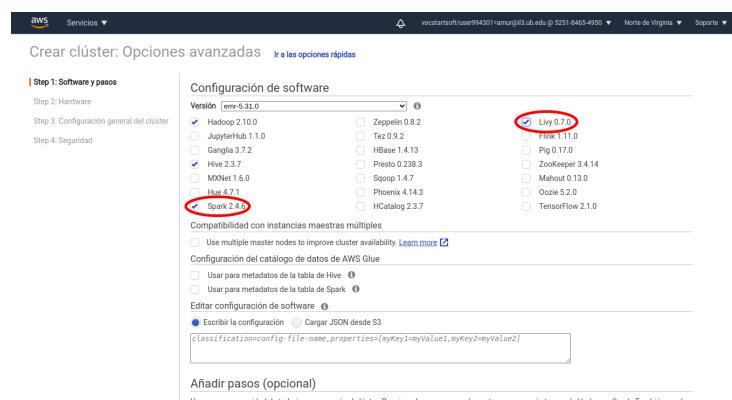


Figura A.15.: Configurar software EMR

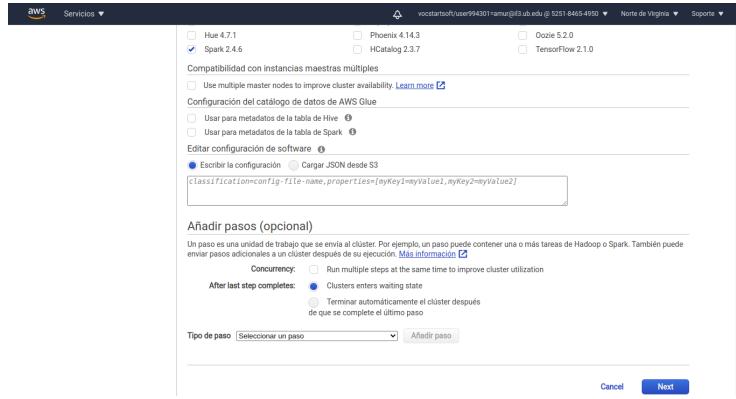


Figura A.16.: Configurar hardware EMR

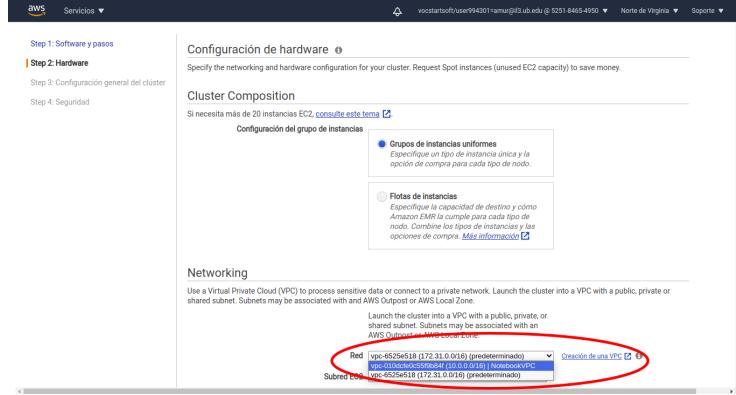


Figura A.17.: Selección de la VPC creada anteriormente

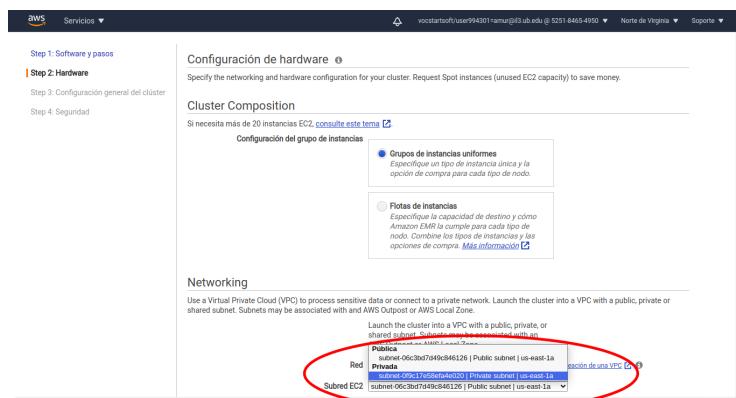


Figura A.18.: Seleccionar subred privada de la VPC

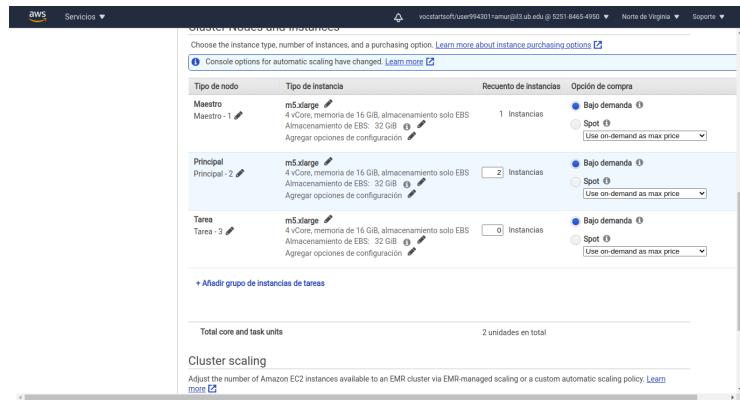


Figura A.19.: Configuración de hardware por defecto

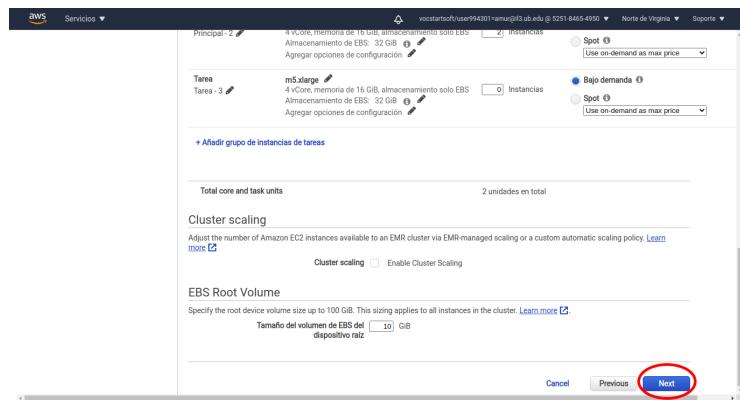


Figura A.20.: Siguiente paso de asistente de creación de EMR

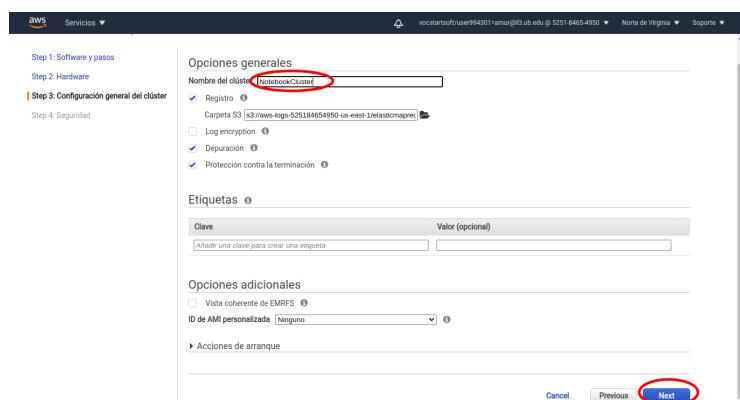


Figura A.21.: Configuración de nombre y opciones del clúster

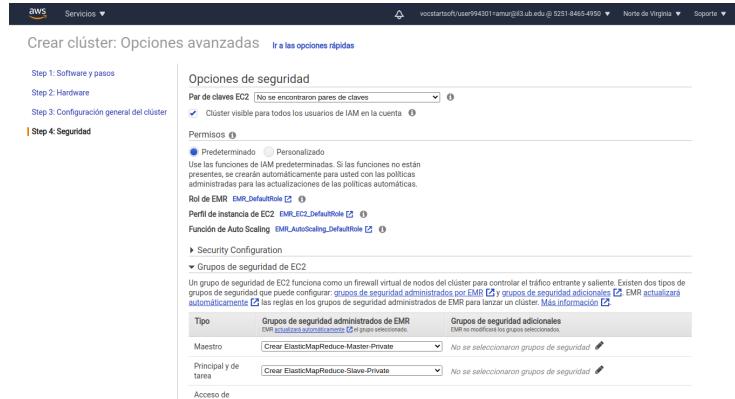


Figura A.22.: Opciones de seguridad del clúster por defecto

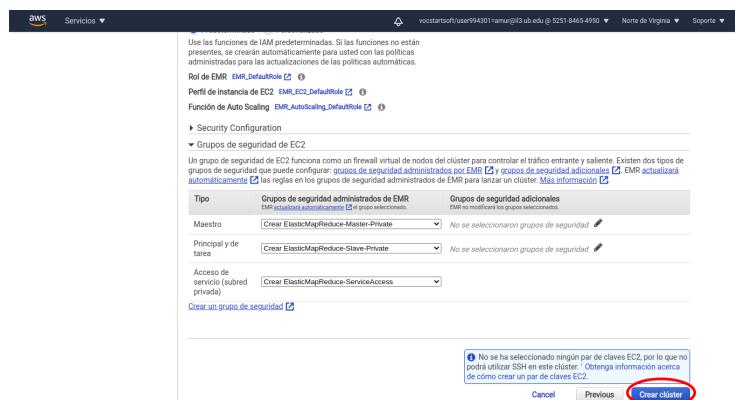


Figura A.23.: Siguiente paso de creación del clúster EMR

A.1.3 Creación del repositorio de Git

Creación del repositorio de git para vincularlo más tarde.

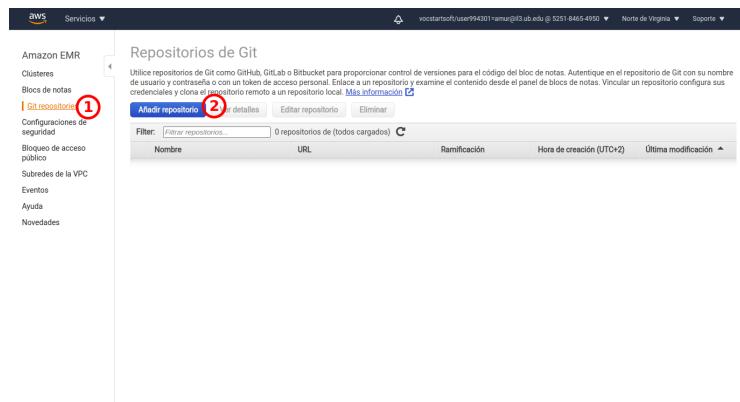


Figura A.24.: Creación de un repositorio de git

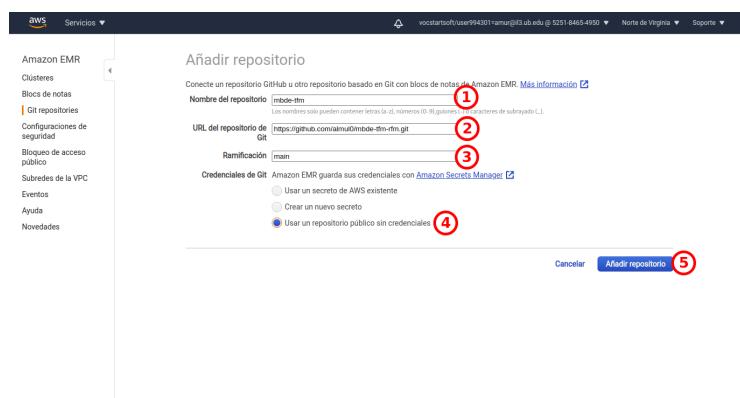


Figura A.25.: Configuración del acceso al repositorio

A.1.4 Creación del cuaderno Jupyter

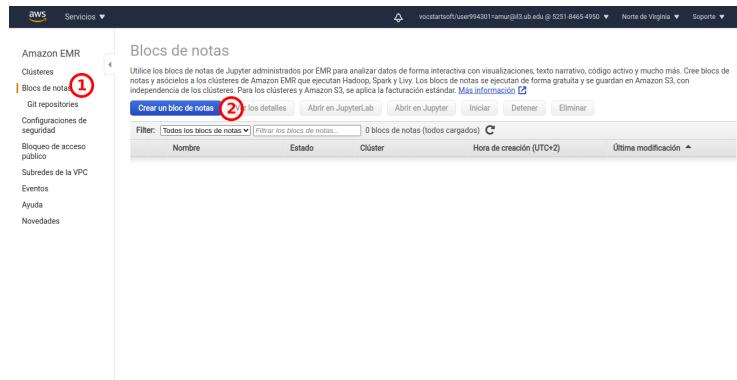


Figura A.26.: Creación del cuaderno Jupyter

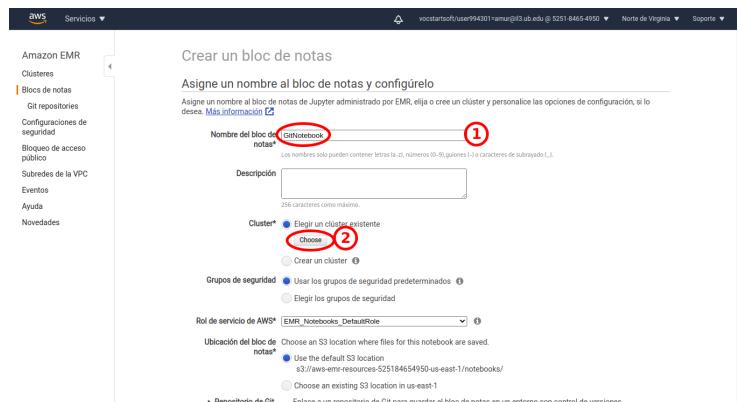


Figura A.27.: Nombre del cuaderno Jupyter

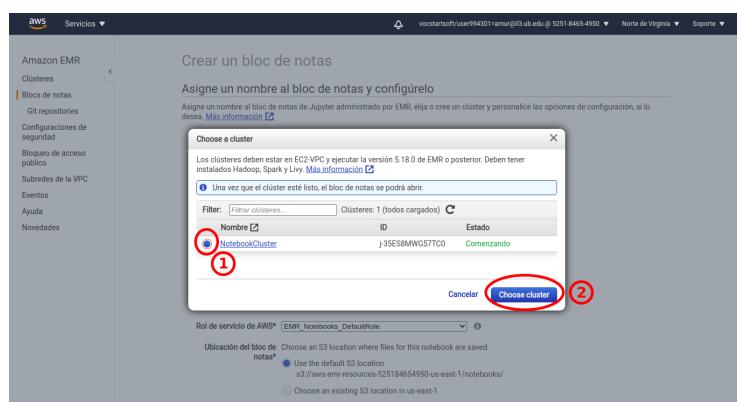


Figura A.28.: Selección del clúster para el cuaderno

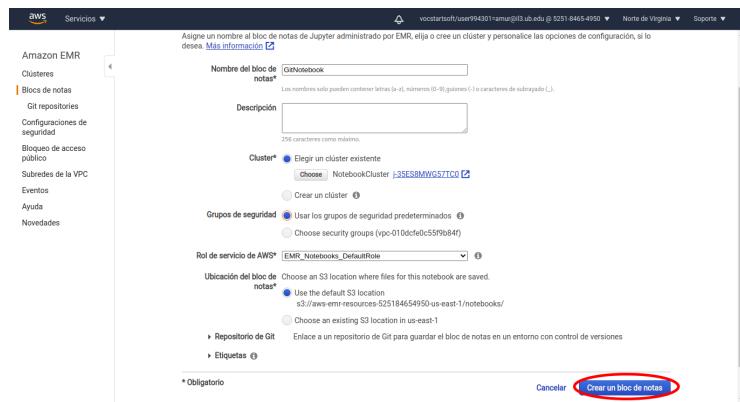


Figura A.29.: Creación del cuaderno cuaderno

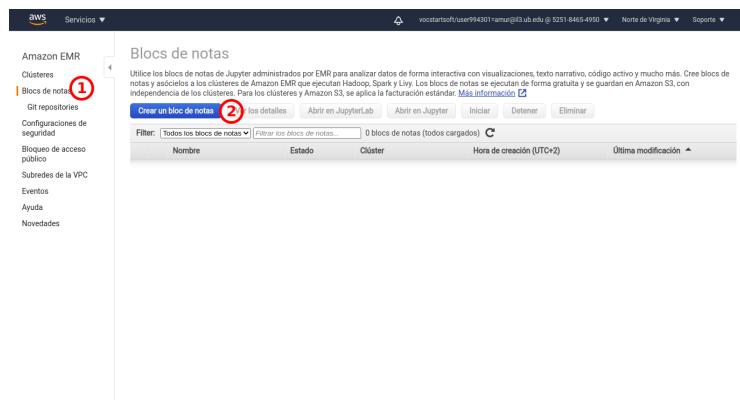


Figura A.30.: Creación del cuaderno Jupyter

Una vez estén listos tanto el clúster como el cuaderno aparecerán los grupos de seguridad y la configuración del cuaderno. El siguiente paso es modificar el grupo de seguridad asociado a la instancia del cuaderno.

Bloc de notas: GitNotebook [Lista](#) Workspace(notebook) is ready to run jobs on cluster j-3SESBMWG57TC0.

[Abrir en JupyterLab](#) [Abrir en Jupyter](#) [Detener](#) [Eliminar](#)

Bloc de notas

ID del bloc de notas: e-6GBMXNK41JSP6057C45M5ECZ
Descripción: -
Última modificación: Hace 12 minutos
Modificado por última ... asumido rol vocstarsoft/user994301-amur@i3.ub.edu
ver por:
Fecha de creación: 2020-10-13 10:52 UTC+2
Creado por: ... asumido rol vocstarsoft/user994301-amur@i3.ub.edu
Rol de IAM de servicio: [EMR_Notebooks_DefaultRole](#)
Grupos de seguridad: [sg-0334e033fc0d593a9](#) (✓)
para la instancia principal
Grupos de seguridad: [sg-00600234d14ff95](#) (✓)
para el bloc de notas
Etiquetas del bloc de: creadorUserId = ARAOXURR3TDEZUIWQMIJuser994301-amur@i3.ub.edu Ver todo / Editar
notas:
Ubicación del bloc de: s2://aws-emr-resources-52184654950-us-east-1/notebooks/
notas:

Cluster

Cluster: NotebookCluster
ID del cluster: j-3SESBMWG57TC0

Figura A.31.: Ir a modificar los grupos de seguridad del cuaderno Jupyter

Grupos de seguridad (2) [Información](#)

[Filtrar grupos de seguridad](#) [Quitar los filtros](#)

Name	ID del grupo de segu...	Nombre del grupo de seguridad	ID de la VPC	Des...
-	sg-00600234d14ff95	ElasticMapReduceEditors-Editor	vpc-010dcf0c55fb84f	Secu...
-	sg-0354e033fc0d593a9	ElasticMapReduceEditors-Livy	vpc-010dcf0c55fb84f	Secu...

Figura A.32.

Grupos de seguridad (2) [Información](#)

[Filtrar grupos de seguridad](#) [Quitar los filtros](#)

Name	ID del grupo de segu...	Nombre del grupo de seguridad	ID de la VPC	Des...
-	sg-00600234d14ff95	ElasticMapReduceEditors-Editor	vpc-010dcf0c55fb84f	Secu...
-	sg-0354e033fc0d593a9	ElasticMapReduceEditors-Livy	vpc-010dcf0c55fb84f	Secu...

Figura A.33.: Edición del grupo de seguridad Editor

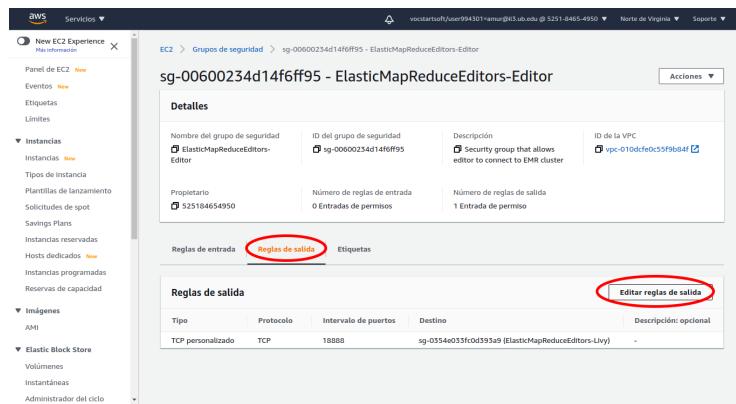


Figura A.34.: Editar las reglas de salida

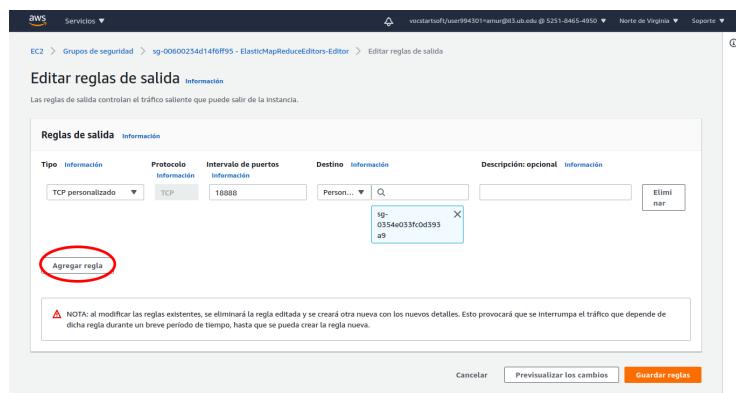


Figura A.35.: Añadir reglas de salida

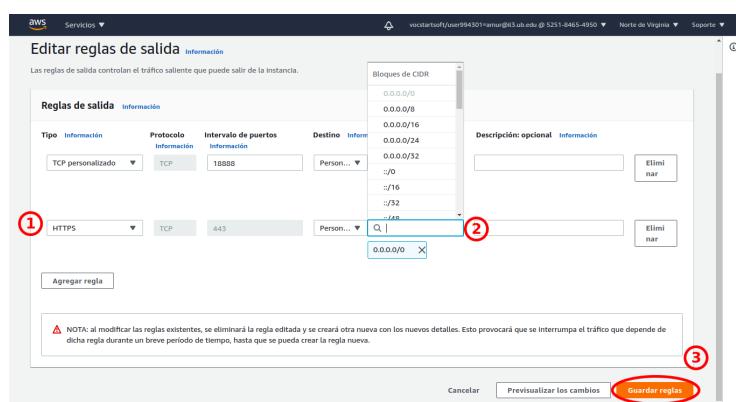


Figura A.36.: Añadir regla de acceso HTTPS

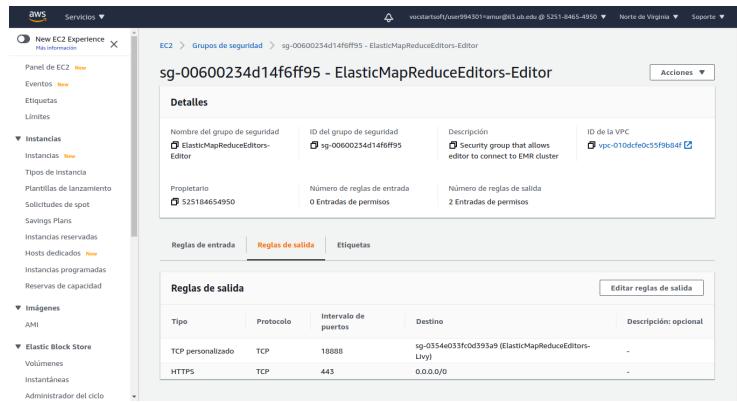


Figura A.37: Verificación de las reglas de salida

A.1.5 Vinculación del repositorio

Una vez modificado el grupo de seguridad, ya se puede vincular el repositorio creado anteriormente.

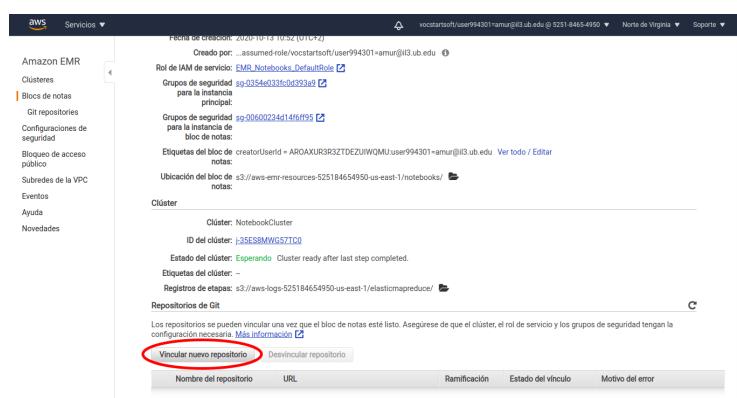


Figura A.38: Vincular el repositorio con el cuaderno

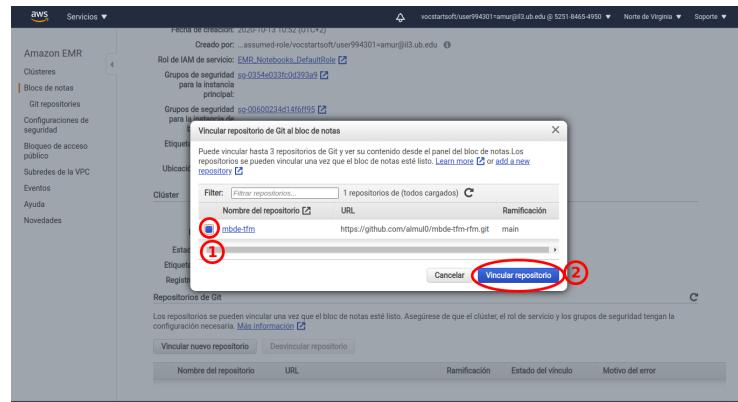


Figura A.39.: Seleccionar el respositorio

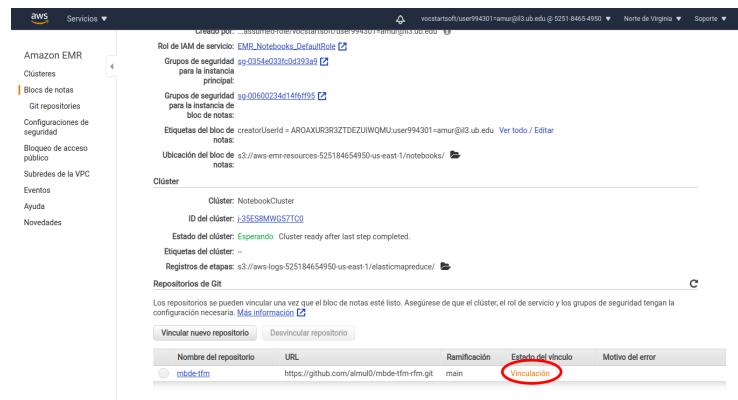


Figura A.40.: Repositorio en estado de vinculación

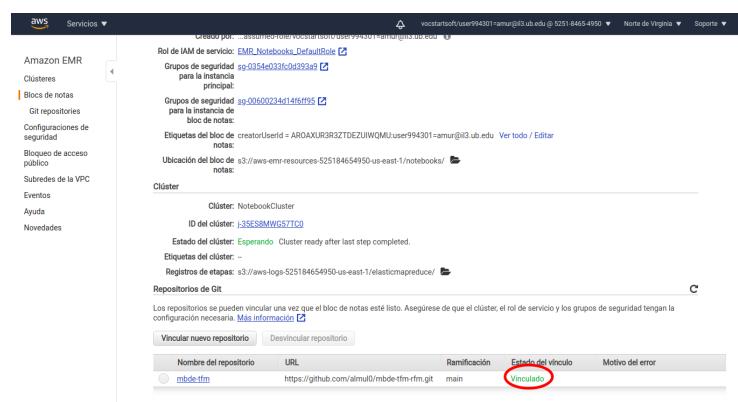


Figura A.41.: Repositorio correctamente vinculado

Una vez vinculado ya se puede abrir el la instancia del cuaderno Jupyter y ver los archivos del repositorio.

A.2 Presentación y resultados para marketing



Marketing Insights

Proyecto Final

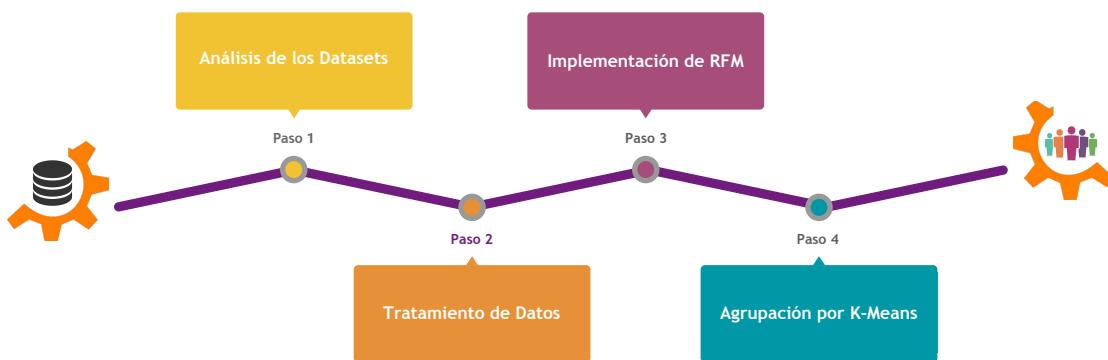
Alberto Mur López
Leonardo Elisei
Tamara Ruiz

Tutora: Cristina Giner Perez



1

Proceso de Segmentación I



Proceso de Segmentación II



En este paso, hemos analizado y comprendido la estructura de los datasets, y en base a este análisis, hemos diseñado la estrategia para la segmentación.

En este paso hemos abordado las tareas de detección, y corrección de valores anómalos, como la de los valores nulos, de aquellas variables necesarias para llevar a cabo nuestra predicción y segmentación.

Esta etapa consiste en la implementación de la técnica de segmentación a través del análisis RFM. Esta técnica ha clasificado los clientes en 64 categorías, por lo que hemos necesitado implementar una agrupación de las mismas.

Este paso dio solución a la enorme cantidad de categorías obtenidas en el paso previo.

La lógica detrás de este método, ha sido capaz de agrupar las categorías en 6 segmentos, cuyos pesos estaban aproximados.

Segmentos obtenidos



Best ↓R ↑F ↑M

Han venido hace poco y gastan mucho.



Promising ↑R <>F ↑M

Llevan tiempo sin venir, vienen poco, aunque algunos suelen venir bastante, y gastan por encima de la media.



New ↓R ↓F ↓M

Han venido hace muy poco, vienen poco, y gastan por debajo de la media.



Risk ↑R ↓F <>M

Vienen por debajo de la media, gastan la media, y hace mucho que no vienen.



Loyal ↓R ↑F ↓M

Han venido hace muy poco, vienen por encima de la media, y gastan por debajo de la media.



Lost ↑R ↓F ↓M

Hace mucho que no vienen, tienen frecuencia baja, y gastan poco.

R = Recency
F = Frequency
M = Monetary

↑ = Por encima de la media
↓ = Por debajo de la media
<> = Media

Métricas analizadas



* Todas las métricas son aplicadas a cada uno de los segmentos.

NOTA IMPORTANTE

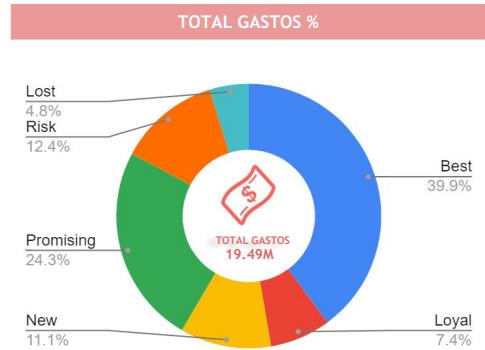
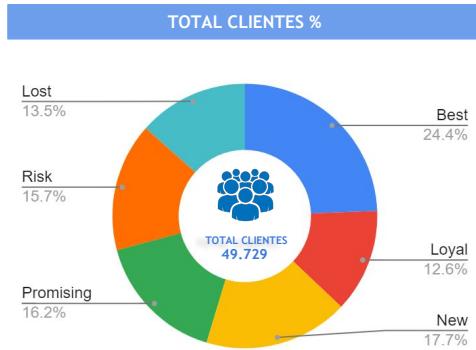


Lamentablemente, habiendo cruzado los datos entre las distintas bases de datos provistas, contamos con una enorme cantidad de datos desconocidos, ya que no existen en los datasets, y representan:

- 92% de los clientes que han comprado
- 65% de los productos vendidos

Por lo tanto, las métricas que mostraremos a continuación, son obtenidas en base a los porcentajes restantes, que son los datos conocidos que sí hemos podido cruzar.

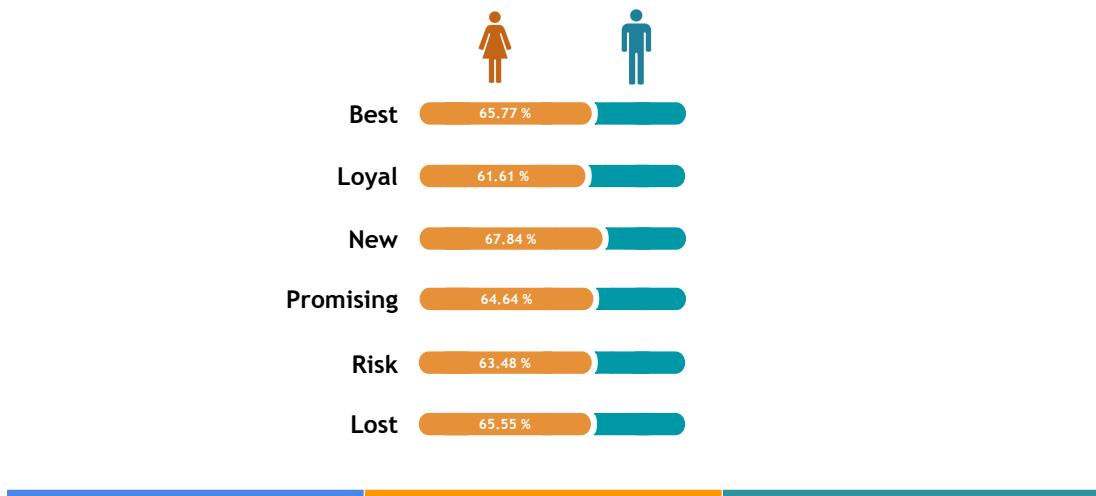
Distribución de segmentos



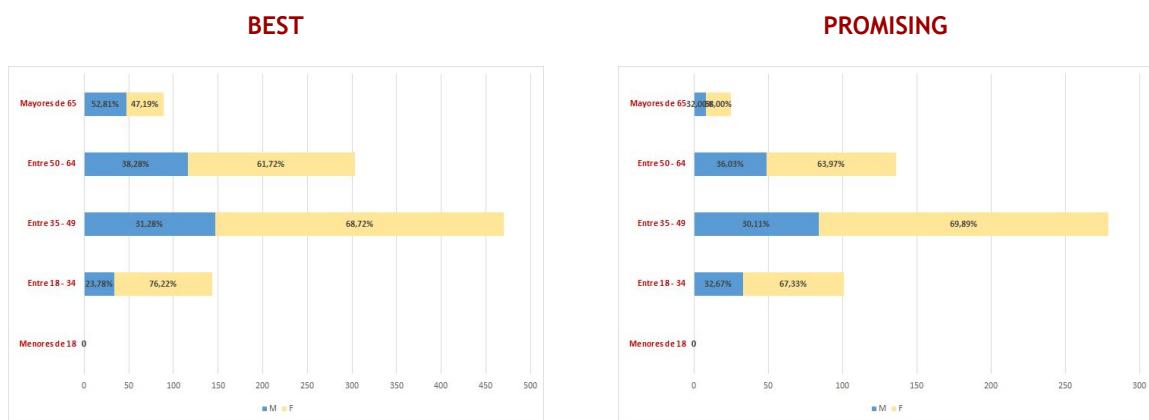
Promedios RFM



Distribución por género



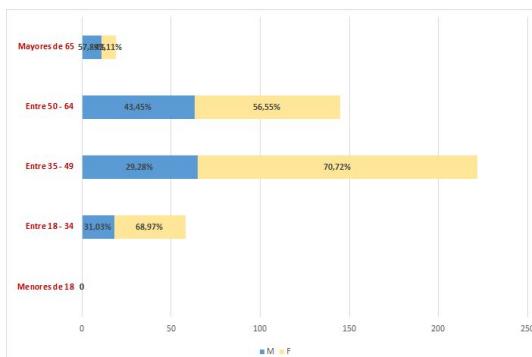
Distribución por género y edad



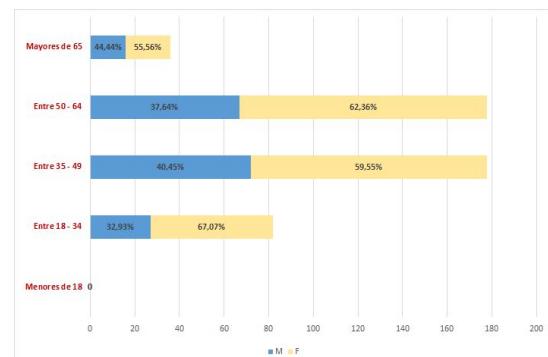
NOTA: Existían otras categorías además de M-F pero su porcentaje era inferior al 0.01%, por lo que para facilitar la visualización, no se muestran en las gráficas.

Distribución por género y edad

NEW



LOYAL

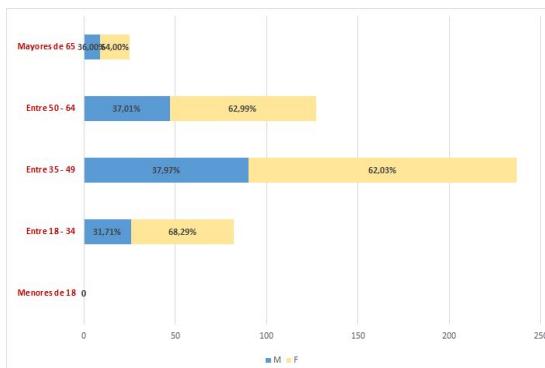


NOTA: Existían otras categorías además de M-F pero su porcentaje era inferior al 0.01%, por lo que para facilitar la visualización, no se muestran en las gráficas.

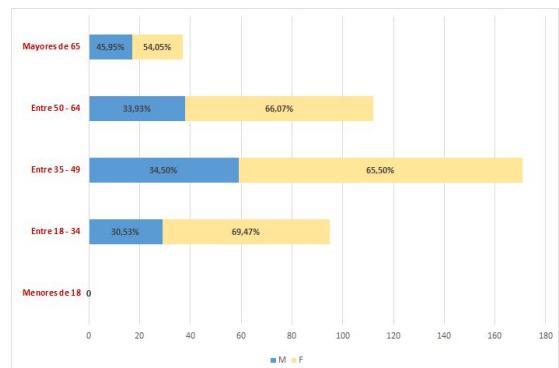


Distribución por género y edad

RISK



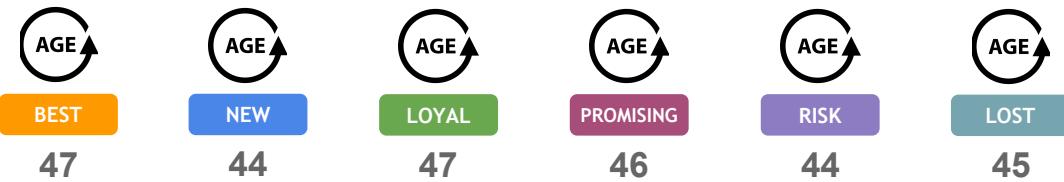
LOST



NOTA: Existían otras categorías además de M-F pero su porcentaje era inferior al 0.01%, por lo que para facilitar la visualización, no se muestran en las gráficas.



Promedio de Edad



Categorías de productos más elegidos



Categorías de productos comprados menos elegidos

► MENOS ELEGIDOS ►



Productos comprados más elegidos

◀ MAS ELEGIDOS ▶



Promedio de cantidad de productos por cesta

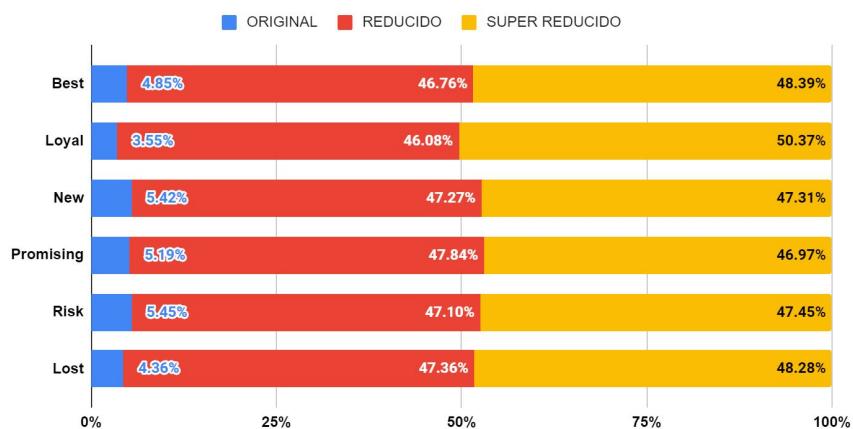


Distribución de productos comprados según los hábitos de consumo de los clientes / su saludabilidad

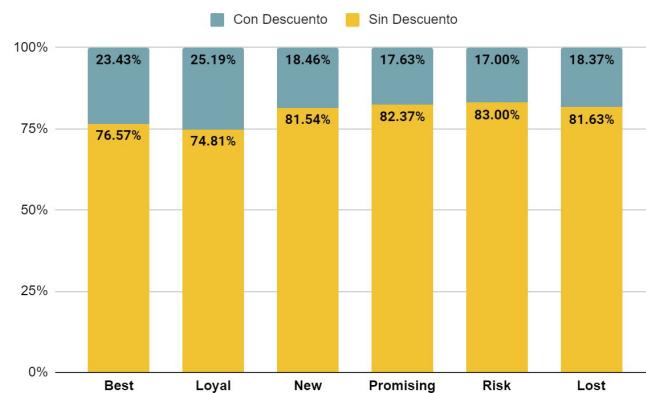
The figure shows the distribution of products bought by consumption habits, categorized into three groups: HEALTHY (green), NONE (blue), and NON HEALTHY (red). The categories are aligned with the six customer segments from the previous chart. The percentages represent the proportion of healthy, non-healthy, and none purchases for each segment.

	HEALTHY	NONE	NON HEALTHY
BEST	61.20%	5.38%	33.41%
LOYAL	53.86%	5.70%	40.43%
NEW	64.08%	5.50%	30.41%
PROMISING	59.92%	5.85%	34.21%
RISK	61.85%	5.66%	32.47%
LOST	56.13%	6.11%	37.74%

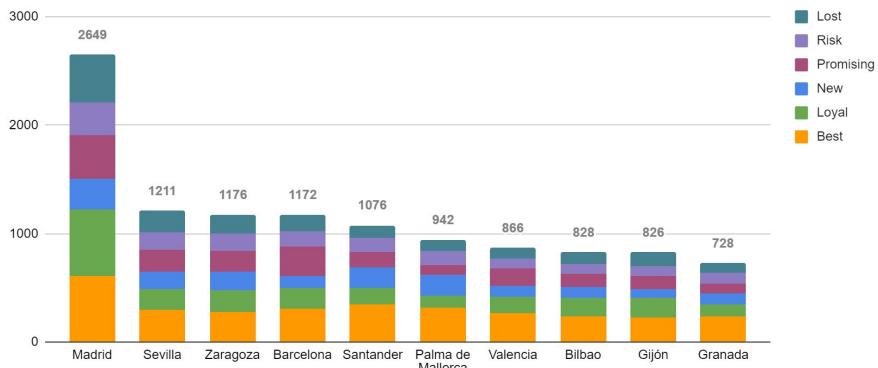
Distribución de compras según su VAT



Porcentaje de Productos comprados con y sin Descuento



Distribución de segmentos en las 10 ciudades con más clientes



A.3 Scripts para el despliegue

A.3.1 Script de entrenamiento

List. A.1: Código para la ejecución del entrenamiento train-segmentation.py

```
1 import argparse
  import logging

  from pyspark.sql import SparkSession
5
  import numpy as np
  from math import sqrt
  import pyspark.sql.functions as F
  from pyspark.sql.types import *
10
  from pyspark.mllib.clustering import KMeans,KMeansModel
  from pyspark.mllib.linalg import Vectors
  from pyspark.ml.evaluation import ClusteringEvaluator
  from pyspark.ml.feature import StandardScaler
15 from pyspark.ml.feature import VectorAssembler

N_CLUSTERS=6

PREDICTION_MAP = {
20    'best': 0,
    'promising': 1,
    'loyal': 2,
    'new': 3,
    'risk': 4,
25    'lost' :5
}

logger = logging.getLogger(__name__)
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
30 spark = None

def compute_rfm(spark, input, quartile_output, r_percentile=None,f_percentile=None,
               m_percentile=None):
    logger.info("Loading data from {}".format(input))
    tickets = spark.read.format("json").load(input)
35
    # Drop duplicate rows
    tickets = tickets.drop_duplicates()
    tickets = tickets.withColumn("date", F.to_date("datekey", "yyyy-MM-dd"))
    logger.info('Grouping tickets')
40    tickets = tickets.groupby(['customerid','storeid','cardtype','date','ticketid','
                               productid']).agg(
        F.sum('extendedamount').alias('extendedamount'),
        F.sum('originalamount').alias('originalamount'),
        F.sum('totaldiscount').alias('totaldiscount'),
        F.sum('quantity').alias('quantity')
45    )
}
```

```

logger.info('Calculating RFM')
max_df = tickets.select(F.max("date")).collect()
max_val = max_df[0][0]
tickets = tickets.withColumn('max_date', F.lit(max_val))
tickets = tickets.withColumn("recencydays", F.expr("datediff(max_date, date)"))
rfm_table = tickets.groupBy("customerid")\
    .agg(F.min("recencydays").alias("recency"), \
        F.countDistinct("ticketid").alias("frequency"), \
        F.sum("extendedamount").alias("monetary"))

55
logger.info('Calculating upper boundaries')
r_trim_cond = f_trim_cond = m_trim_cond = F.lit(True)
r_out_cond = f_out_cond = m_out_cond = F.lit(False)

60
if r_percentile:
    r_up_boundary = rfm_table.approxQuantile("recency", [r_percentile], 0)
    r_trim_cond = (F.col("recency") < r_up_boundary[0])
    r_out_cond = (F.col("recency") >= r_up_boundary[0])

65
if f_percentile:
    f_up_boundary = rfm_table.approxQuantile("frequency", [f_percentile], 0)
    f_trim_cond = (F.col("frequency") < f_up_boundary[0])
    f_out_cond = (F.col("frequency") >= f_up_boundary[0])

70
if m_percentile:
    m_up_boundary = rfm_table.approxQuantile("monetary", [m_percentile], 0)
    m_trim_cond = (F.col("monetary") < m_up_boundary[0])
    m_out_cond = (F.col("monetary") >= m_up_boundary[0])

75
logger.info('Trimming data to compute quartiles')
rfm_trimmed = rfm_table
rfm_trimmed = rfm_trimmed.filter(r_trim_cond & f_trim_cond & m_trim_cond)
rfm_outliers = rfm_table
rfm_outliers = rfm_outliers.filter(r_out_cond | f_out_cond | m_out_cond)

total_count = rfm_table.count()
trimmed_count = rfm_trimmed.count()
outliers_count = rfm_outliers.count()
logger.info(f"Total = {total_count}")
logger.info(f"Trimmed = {trimmed_count}")
logger.info(f"Outliers = {outliers_count}")

#Create quartiles for each metric
90
logger.info('Computing quartiles')
r_quartile = rfm_trimmed.approxQuantile("recency", [0.25, 0.5, 0.75], 0)
f_quartile = rfm_trimmed.approxQuantile("frequency", [0.25, 0.5, 0.75], 0)
m_quartile = rfm_trimmed.approxQuantile("monetary", [0.25, 0.5, 0.75], 0)

95
spark.createDataFrame(r_quartile, FloatType()).write.mode('overwrite').csv(
    quartile_output+"/r_quartile")
spark.createDataFrame(f_quartile, FloatType()).write.mode('overwrite').csv(
    quartile_output+"/f_quartile")
spark.createDataFrame(m_quartile, FloatType()).write.mode('overwrite').csv(
    quartile_output+"/m_quartile")

```

```

100     #assing score from 1 to 4 depending on quantile => 1 best mark - 4 worst one
100     logger.info('Assigning scores')
100     rfm_table = rfm_table.withColumn("r_quartile",F.when(F.col("recency") > r_quartile[2] ,
100                                         4).\
100                                         when(F.col("recency") >
100                                         r_quartile[1] , 3).\
100                                         when(F.col("recency") >
100                                         r_quartile[0] , 2).\
100                                         otherwise(1))

105     rfm_table = rfm_table.withColumn("f_quartile",F.when(F.col("frequency") > f_quartile[2]
105                                         ] , 1).\
105                                         when(F.col("frequency") >
105                                         f_quartile[1] , 2).\
105                                         when(F.col("frequency") >
105                                         f_quartile[0] , 3).\
105                                         otherwise(4))

110     rfm_table = rfm_table.withColumn("m_quartile",F.when(F.col("monetary") > m_quartile[2]
110                                         , 1).\
110                                         when(F.col("monetary") >
110                                         m_quartile[1] , 2).\
110                                         when(F.col("monetary") >
110                                         m_quartile[0] , 3).\
110                                         otherwise(4))

115     rfm_table = rfm_table.withColumn("rfm_score", F.concat(F.col("r_quartile"), F.col("f_quartile"),
115                                         F.col("m_quartile")))

120     return rfm_table

120 def standarize_features(rfm_table):
120     assembler = VectorAssembler(inputCols=['r_quartile','f_quartile','m_quartile'],
120                                 outputCol='features',handleInvalid = 'skip')
120     rfm_table = assembler.transform(rfm_table)

125     standardizer = StandardScaler(withMean=True, withStd=True).setInputCol("features").
125                                         setOutputCol("scaled_features")
125     std_model = standardizer.fit(rfm_table)
125     features = std_model.transform(rfm_table)
125     return features

130 def extract(row):
130     return tuple(row.scaled_features.toArray().tolist())

130 def calculate_initial_centroids(spark, features):
130     rm_features = features.select('scaled_features').rdd.map(extract).toDF()
135     rows_count = rm_features.count()
135     chunk_size=np.ceil(rows_count/N_CLUSTERS)
135     bc_chunk_size = spark.sparkContext.broadcast(chunk_size)

135     r_vector = rm_features.select('_1').rdd.sortBy(lambda x: x[0]).zipWithIndex().map(
135                                         lambda x: (x[0][0],int(x[1]/bc_chunk_size.
135                                         value))).toDF()

```

```

140     f_vector = rm_features.select('_2').rdd.sortBy(lambda x: x[0]).zipWithIndex().map(
141         lambda x: (x[0][0], int(x[1]/bc_chunk_size.
142             value))).toDF()
143     m_vector = rm_features.select('_3').rdd.sortBy(lambda x: x[0]).zipWithIndex().map(
144         lambda x: (x[0][0], int(x[1]/bc_chunk_size.
145             value))).toDF()
146
147     r_median = r_vector.groupby('_2').agg(F.expr('percentile_approx(_1, 0.5)').alias(
148         'r_median'))
149     f_median = f_vector.groupby('_2').agg(F.expr('percentile_approx(_1, 0.5)').alias(
150         'f_median'))
151     m_median = m_vector.groupby('_2').agg(F.expr('percentile_approx(_1, 0.5)').alias(
152         'm_median'))
153
154     rfm_medians = r_median.join(f_median, '_2').join(m_median, '_2').sort('_2').select([
155         'r_median', 'f_median', 'm_median'])
156     initial_centroids = rfm_medians.rdd.map(lambda x: np.array([x[0],x[1],x[2]])).collect
157
158     return initial_centroids
159
160 def train_kmeans(train_features, initial_model):
161     model = KMeans.train(train_features, N_CLUSTERS ,initialModel=initial_model)
162     return model
163
164 if __name__ == "__main__":
165     parser = argparse.ArgumentParser()
166     parser.add_argument(
167         '--input',
168         help="S3 location for transaction files")
169     parser.add_argument('--model-output', help="K-Means model output")
170     parser.add_argument('--quartile-output', help="K-Means model output")
171     parser.add_argument('--single-files', action="store_true", help="Single file output
172                         flag")
173     parser.add_argument('--r-percentile', type=float)
174     parser.add_argument('--f-percentile', type=float)
175     parser.add_argument('--m-percentile', type=float)
176     parser.add_argument('--prediction-output', help="S3 from output prediction")
177     args = parser.parse_args()
178
179     with SparkSession.builder.appName("Train segments").getOrCreate() as spark:
180         spark.sparkContext._jsc.hadoopConfiguration().set("fs.s3.canned.acl",
181                                         "BucketOwnerFullControl")
182
183         rfm_table = compute_rfm(spark, args.input, args.quartile_output,
184                                r_percentile= args.r_percentile,
185                                f_percentile= args.f_percentile,
186                                m_percentile= args.m_percentile)
187
188         # Standarize features
189         logger.info("Standarize features")
190         features = standarize_features(rfm_table)
191
192         # Centroid inicializacion
193         initial_centroids = calculate_initial_centroids(spark, features)
194         logger.info(f"Initial centroids: {initial_centroids}")

```

```

185     initial_model = KMeansModel(initial_centroids)

    train_features = features.select('scaled_features').rdd.map(lambda x: x[0].toArray
        ()).cache()
    logger.info("Training K-Means")
    model = train_kmeans(train_features, initial_model)
190

    logger.info("Saving model....")
    model.save(spark, args.model_output)

    logger.info("Prediction....")
195    labels = features.select(['customerid','scaled_features']).rdd.map(lambda x: (x[0],
        model.predict(x[1].toArray()))).toDF(['
        customerid','prediction'])

    prediction = features.join(labels,'customerid')

    prediction_map = dict((v,k) for k,v in PREDICTION_MAP.items())
    logger.info("Segment mapping")
200    map_func = F.udf(lambda row : prediction_map.get(row,row))
    prediction = prediction.withColumn("segmento",map_func(F.col('prediction')))
    prediction = prediction.withColumn("prediction_label",F.concat(F.lit('('),F.col('
        prediction'),F.lit(')'),F.col('segmento')))

205    logger.info("Saving")
    prediction.write.mode('overwrite').format('json').save(args.prediction_output + '/
        all')
    for k in PREDICTION_MAP:
        prediction.filter(F.col('segmento') == k).write.mode('overwrite').format('json'
            ).save(args.prediction_output + "/" + k)

```

A.3.2 Script de ejecución

List. A.2: Código para la ejecución de la segmentación train-segmentation.py

```

1 import argparse
    import logging

    from pyspark.sql import SparkSession
5
    import numpy as np
    from math import sqrt
    import pyspark.sql.functions as F
    from pyspark.sql.types import *
10
    from pyspark.mllib.clustering import KMeans,KMeansModel
    from pyspark.mllib.linalg import Vectors
    from pyspark.ml.evaluation import ClusteringEvaluator
    from pyspark.ml.feature import StandardScaler
15 # from pyspark.sql import SQLContext

```

```

# from pyspark.sql.functions import col, when, lit, expr, countDistinct, max, min, sum,
                                         concat
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.evaluation import ClusteringEvaluator
from pyspark.ml.feature import StandardScaler
20
N_CLUSTERS=6

PREDICTION_MAP = {
    'best': 0,
25    'promising': 1,
    'loyal': 2,
    'new': 3,
    'risk': 4,
    'lost' :5
30}

logger = logging.getLogger(__name__)
logging.basicConfig(level=logging.INFO, format='%(levelname)s: %(message)s')
spark = None
35

def compute_rfm(spark, input, quartile_input):
    logger.info("Loading data from {}".format(input))
    tickets = spark.read.format("json").load(input)

40    # Drop duplicate rows
    tickets = tickets.drop_duplicates()
    tickets = tickets.withColumn("date", F.to_date("datekey", "yyyy-MM-dd"))
    logger.info('Grouping tickets')
    tickets = tickets.groupby(['customerid','storeid','cardtype','date','ticketid',
                               'productid']).agg(
        F.sum('extendedamount').alias('extendedamount'),
        F.sum('originalamount').alias('originalamount'),
        F.sum('totaldiscount').alias('totaldiscount'),
        F.sum('quantity').alias('quantity')
    )
    logger.info('Calculating RFM')
50    max_df = tickets.select(F.max("date")).collect()
    max_val = max_df[0][0]
    tickets = tickets.withColumn('max_date',F.lit(max_val))
    tickets = tickets.withColumn("recencydays", F.expr("datediff(max_date, date)"))
    rfm_table = tickets.groupBy("customerid")\
        .agg(F.min("recencydays").alias("recency"), \
             F.countDistinct("ticketid").alias("frequency"), \
             F.sum("extendedamount").alias("monetary"))

60    r_quartile = sorted([row[0] for row in spark.read.csv(quartile_input+"/r_quartile").
                           collect()])
    f_quartile = sorted([row[0] for row in spark.read.csv(quartile_input+"/f_quartile").
                           collect()])
    m_quartile = sorted([row[0] for row in spark.read.csv(quartile_input+"/m_quartile").
                           collect()])

    #assing score from 1 to 4 depending on quantile => 1 best mark - 4 worst one
65    logger.info('Assigning scores')

```

```

rfm_table = rfm_table.withColumn("r_quartile", F.when(F.col("recency") > r_quartile[2] ,
4).\
when(F.col("recency") >
r_quartile[1] , 3).\
when(F.col("recency") >
r_quartile[0] , 2).\
otherwise(1))

70 rfm_table = rfm_table.withColumn("f_quartile", F.when(F.col("frequency") > f_quartile[2]
] , 1).\
when(F.col("frequency") >
f_quartile[1] , 2).\
when(F.col("frequency") >
f_quartile[0] , 3).\
otherwise(4))

75 rfm_table = rfm_table.withColumn("m_quartile", F.when(F.col("monetary") > m_quartile[2]
, 1).\
when(F.col("monetary") >
m_quartile[1] , 2).\
when(F.col("monetary") >
m_quartile[0] , 3).\
otherwise(4))

80 rfm_table = rfm_table.withColumn("rfm_score", F.concat(F.col("r_quartile"), F.col("f_quartile"),
F.col("m_quartile")))

return rfm_table

85 def standarize_features(rfm_table):
    assembler = VectorAssembler(inputCols=['r_quartile','f_quartile','m_quartile'],
                                outputCol='features',handleInvalid = 'skip')
    rfm_table = assembler.transform(rfm_table)

    90 standardizer = StandardScaler(withMean=True, withStd=True).setInputCol("features").
                                setOutputCol("scaled_features")
    std_model = standardizer.fit(rfm_table)
    features = std_model.transform(rfm_table)
    return features

    95 if __name__ == "__main__":
        parser = argparse.ArgumentParser()
        parser.add_argument(
            '--input',
            help="S3 location for transaction files")
        100 parser.add_argument('--model-input', help="K-Means model input")
        parser.add_argument('--quartile-input', help="Quartile input")
        parser.add_argument('--single-files', action="store_true", help="Single file output
                                    flag")
        parser.add_argument(
            '--prediction-output', help="S3 from output prediction")
        105 args = parser.parse_args()

        with SparkSession.builder.appName("Train segments").getOrCreate() as spark:

```

```

spark.sparkContext._jsc.hadoopConfiguration().set("fs.s3.canned.acl","BucketOwnerFullControl")

110    rfm_table = compute_rfm(spark, args.input, args.quartile_input)

    # Standarize features
    logger.info("Standarize features")
    features = standarize_features(rfm_table)

115    train_features = features.select('scaled_features').rdd.map(lambda x: x[0].toArray
                                                               ()).cache()
    logger.info("Training K-Means")

    model = KMeansModel.load(spark.sparkContext, args.model_input)

120    logger.info("Prediction....")
    labels = features.select(['customerid','scaled_features']).rdd.map(lambda x: (x[0],
                                                               model.predict(x[1].toArray()))).toDF(['
                                                               customerid','prediction'])

    prediction = features.join(labels,'customerid')

125    prediction_map = dict((v,k) for k,v in PREDICTION_MAP.items())
    logger.info("Segment mapping")
    map_func = F.udf(lambda row : prediction_map.get(row,row))
    prediction = prediction.withColumn("segmento",map_func(F.col('prediction')))
    prediction = prediction.withColumn("prediction_label",F.concat(F.lit('('),F.col('
                                                               prediction'),F.lit(')'),F.col('segmento')))

    logger.info(prediction.take(5))

    logger.info("Saving")
    prediction.write.mode('overwrite').format('json').save(args.prediction_output + '/'
                                                all")
135    for k in PREDICTION_MAP:
        prediction.filter(F.col('segmento') == k).write.mode('overwrite').format('json'
                                                               ).save(args.prediction_output + "/" + k)

```

