

# Motor de recomendación para escenarios de comercio electrónico



**Enric Rovira Meléndez**

Master Datascience -

Kschool

16-6-2019



# INDÍCE

## Contenido

<b>INDÍCE</b> .....	2
<b>Introducción:</b> .....	3
Objetivos del Proyecto: .....	3
<b>Descripción de los datos:</b> .....	4
Limpieza de los datos: .....	5
<b>Metodología</b> .....	5
Técnicas utilizadas:.....	6
Descripción del proceso: .....	7
Modelo generador de candidatos:.....	9
Modelo de ranqueo:.....	10
<b>Resultados y Conclusiones</b> .....	12
Resultados:.....	12
Limitaciones de los resultados: .....	13
Próximos pasos:.....	14
Conclusiones: .....	14

## Introducción:

### Objetivos del Proyecto:

El objetivo del proyecto es presentar un MVP (Mínimo Producto Viable) sobre un recomendador a medida para escenarios de comercio electrónico (*ecommerce*).

#### **¿Por qué a medida?**

Porque de esta forma cada *DataScientist* puede aportar su conocimiento de negocio y aplicar los diferentes ajustes que se requieran sobre el flujo base que yo aporte.

La idea inicial es combinar la información de los clientes junto con la de los artículos pudiendo capturar y aprovechar todo de interacción entre el cliente y el producto.

Mi objetivo no es crear el mejor recomendador posible para mi conjunto de datos sino crear un flujo de proceso estándar para un recomendador de *ecommerce*.

#### **¿Por qué es relevante?**

Está basado en Redes Neuronales lo que permite aprovechar en mayor medida los datos de los que disponemos y mezcla las dos técnicas más populares, como son el filtrado colaborativo y el basado en contenido.

#### **¿Estado del arte o proyectos relacionados?**

Este proyecto surgió porque no encontré ningún estándar ni proyecto que abarcara la personalización a un nivel tan detallado por tanto me propuse aprovechar ese vacío para crear un estándar.

## Descripción de los datos:

Los datos de entrada se han obtenido mediante WebScraping de la página:

<http://www.elcorteingles.es/>

- Una vez realizado el **WebScraping** los datos finales son los reflejados en la siguiente tabla:

Variable	Descripción
ID	Identificador del Producto 1
Brand	Marca del Producto
Store_Id	Identificador de la unidad de negocio del producto
Badges	Servicios de entrega
Price	Precio del Producto en EUR
Discount	Descuento actual sobre el producto en caso de tenerlo
Media	Cantidad de archivos multimedia
Name	Descripción del producto
Variant	[Desconocido]
Category	Categoría/s del producto
Alternative_Id	Identificador del Producto 2
Eci_Provider	[Desconocido]
Gtin	[Desconocido]
Status	Estado del Producto
Quantity	Cantidades incluidas en el producto
Image	Link a la imagen del producto (JPEG)

- Los **datos de los Clientes** al no ser públicos he tenido que generarlos aleatoriamente simulando una distribución parecida a la real donde hay pocos artículos muy populares y por tanto muy comprados y muchos artículos con pocas compras comparados con los más populares. Como se observa en la imagen inferior.



Los datos finales de las compras de los Clientes son los reflejados en la siguiente tabla:

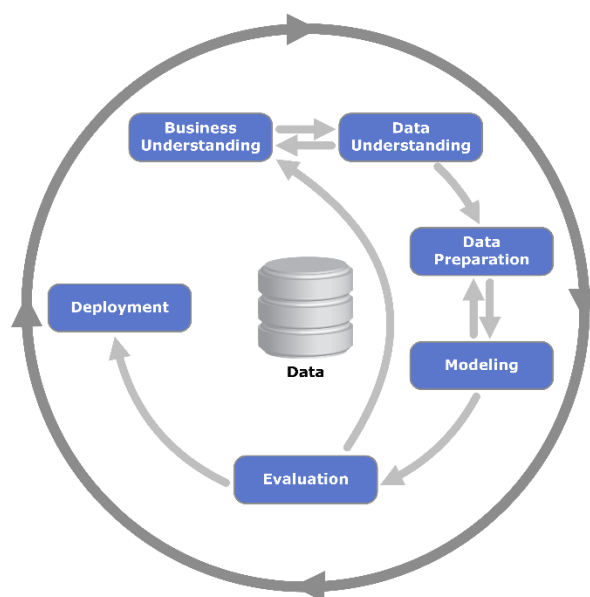
Variable	Descripción
date	Fecha de la transacción
item_id	Identificador del artículo
brand	Marca del artículo
PRICE	Precio del artículo
customer_id	Identificador del cliente
text	Descripción del articulo

### Limpieza de los datos:

- Se han descartado algunas secciones como alimentación, venta de entradas y restauración con el objetivo de reducir volumen y sobre todo porque supermercado tiene un comportamiento tan diferente que debería ser tratado por separado.
- Se han descargado el 60% de las imágenes, pero finalmente no se han incorporado al modelo final por rendimiento, sin embargo, he dejado detallado algunos beneficios de incorporarlas y que sin duda habría que hacer en un proyecto más duradero.

## Metodología


Para el desarrollo del proyecto se ha utilizado la metodología [CRISP-DM](#)




Utilizando un sistema de carpetas como el de la *Imagen2*, y una tipología de nombres y versionado estándar que seguía el siguiente patrón: **Indice\_Nombre\_FechaCreacion** (*Imagen3*)


 00\_EstudioDatoss\_20190308.ipynb


(Imagen3)

 00\_Datos

 01\_Compresion\_de\_Negocio

 02\_Compresion de Datos

 03\_Preparacion\_de\_Datos

 04\_Modelado\_y\_Evaluacion

 05\_Implementacion

(Imagen2)

### Técnicas utilizadas:

- Para el desarrollo total del Proyecto se han utilizado los siguientes lenguajes:

- Python
- Mark Down
- HTML
- CSS

-Y las siguientes herramientas:

- Jupyter Notebook
- Sublime
- Anaconda
- Github
- Bootstrap

-Las principales APIS utilizadas han sido **Pandas, Keras y Flask**.

A continuación, detallo los pasos del flujo del proyecto:

1. *WebScrapping*
2. Descarga de imágenes
3. Limpieza de Datos
4. EDA
5. Limpieza y Normalizacion del texto
6. Creación de un *WordEmbedding*
7. Creación de un *Autoencoder* de texto
8. Creación de un *Autoencoder* de imágenes
9. Creación del Modelo Generador de Candidatos
10. Creación del Modelo de Ranqueo
11. Función de Recomendación
12. Creación y Diseño del *WebService*

### Descripción del proceso:

El **WebScraping** descarga los datos de la fuente y los transforma en un CSV, que nos será más fácil de manejar, posteriormente descargamos todas las **imágenes** contenida en los enlaces que hemos guardado en dicho CSV.

Una vez tenemos descargada la información, procedemos a limpiar los datos quitando columnas que no utilizaremos y haciendo un estudio de las columnas que tenemos viendo valores distintos, valores más frecuentes, tratamiento de nulos etc. Nos guardaremos asimismo CSVs a modo de Dimensiones de Información como Marcas, Categorías, Descripciones etc.

Una vez finalizado el proceso de **estudio y limpieza de los datos** de forma genérica nos centramos en el tratamiento del texto contenido en las descripciones y las categorías. Decidí juntar el texto de las categorías con el de las descripciones ya que de esta forma nos aporta mucha más información del producto en cuestión y dejar el texto de la Marca en otra columna ya que la trataremos como una variable categórica.


En el proceso **de normalización del texto** procederemos a eliminar *Stopwords* estándar y *stopwords* que he definido en la exploración del texto, asimismo unifico entidades mediante ngramas de 2 y 3 palabras como Harry Potter que pasa a ser Harry\_Potter y El Corte Ingles que pasa a ser El\_Corte\_Ingles, convirtiéndose de esta manera en una única palabra ya que, aunque estén formadas por dos o tres palabras la entidad la forman el conjunto de sus palabras.

Asimismo, también limpiamos acentos, plurales y caracteres extraños haciendo que vestidos y vestido sean la misma palabra. Además de esto recurrimos al **filtrado de marcas** que pueden confundir ya que si aparece el texto Nike en una zapatilla no significa lo mismo que si aparece en una camiseta por eso decidí borrar las marcas del texto y dejarlas en una columna para que el modelo no use la marca contenida en el texto como punto de unión entre productos, es decir, no por el hecho de que dos productos contienen la palabra 'Nike' en el texto tienen por qué ser parecidos.

Por último, procederemos a **lematizar** las palabras quedándonos con el lexema de la palabra, para lo cual he utilizado un diccionario de lexemas de español base (el cual he editado con algunas palabras de mi corpus) para que dada una palabra de entrada me devuelva su correspondiente palabra lematizada. Esto sobre todo tiene efecto en los verbos y sus conjugaciones por ejemplo de transparentan pasamos a transparentar.

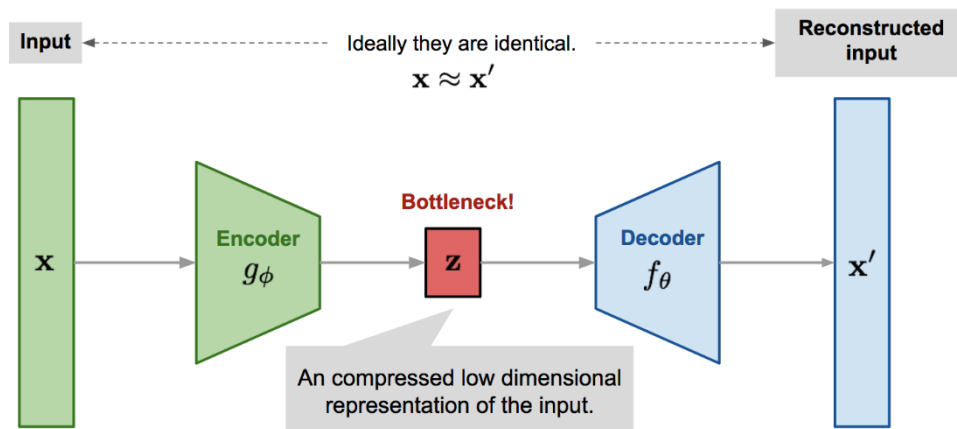


Una vez tenemos limpiado el texto, nos disponemos a crear un **WordEmbedding** de dimensión 100, mediante [Word2Vec](#) con el que crearemos una representación vectorial de cada palabra. Con este método, las palabras que suelen aparecer juntas tenderán a tener una representación vectorial similar entre sí que las que nunca salen juntas. De este modo creamos un contexto de nuestro corpus. Por ejemplo, dada la palabra de entrada “sandalia” las palabras que tienen una representación vectorial parecida a la de “sandalia” dentro de nuestro corpus serian:

“sandalia” 

```
[('alpargata', 0.6290709972381592),
 ('botin', 0.6135832667350769),
 ('salon', 0.5850276947021484),
 ('mocasin', 0.5569319725036621),
 ('chancla', 0.5555477142333984),
 ('bota', 0.5497039556503296),
 ('mercedita', 0.5435197353363037),
 ('nautico', 0.5411031246185303),
 ('pepito', 0.5364699363708496),
 ('deportivas', 0.5191572308540344)]
```

Usando este **WordEmbedding** creamos un [Autoencoder](#) de nuestro texto. El Autoencoder busca crear una representación de nuestro texto, pero con un vector mucho menor y por tanto mucho mas manejable como muestra la imagen inferior.

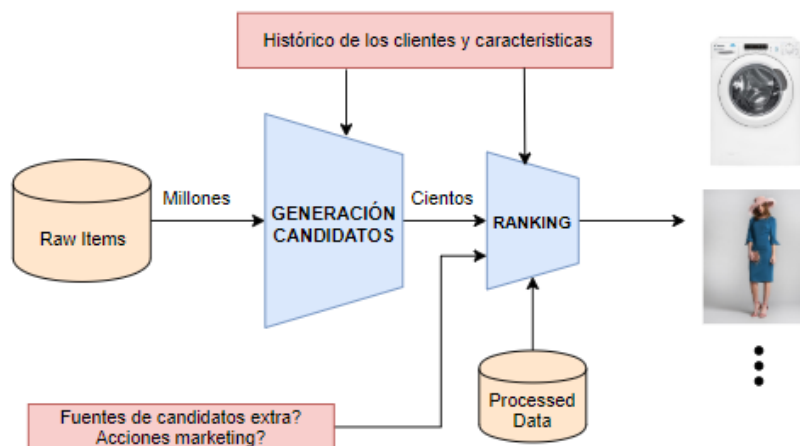


El **Autoencoder de las imágenes** tiene un aspecto muy parecido al del texto, pero en vez de recibir texto como entrada, recibe imágenes y de esta forma crear un vector de tamaño más reducido y manejable que nos permita representar nuestras imágenes lo mejor posible. Asimismo, al igual que en el texto utilizando el método de [KNearestNeighbours](#) podemos obtener un conjunto de imágenes parecidas a una imagen de entrada dada como se ve en la imagen inferior.



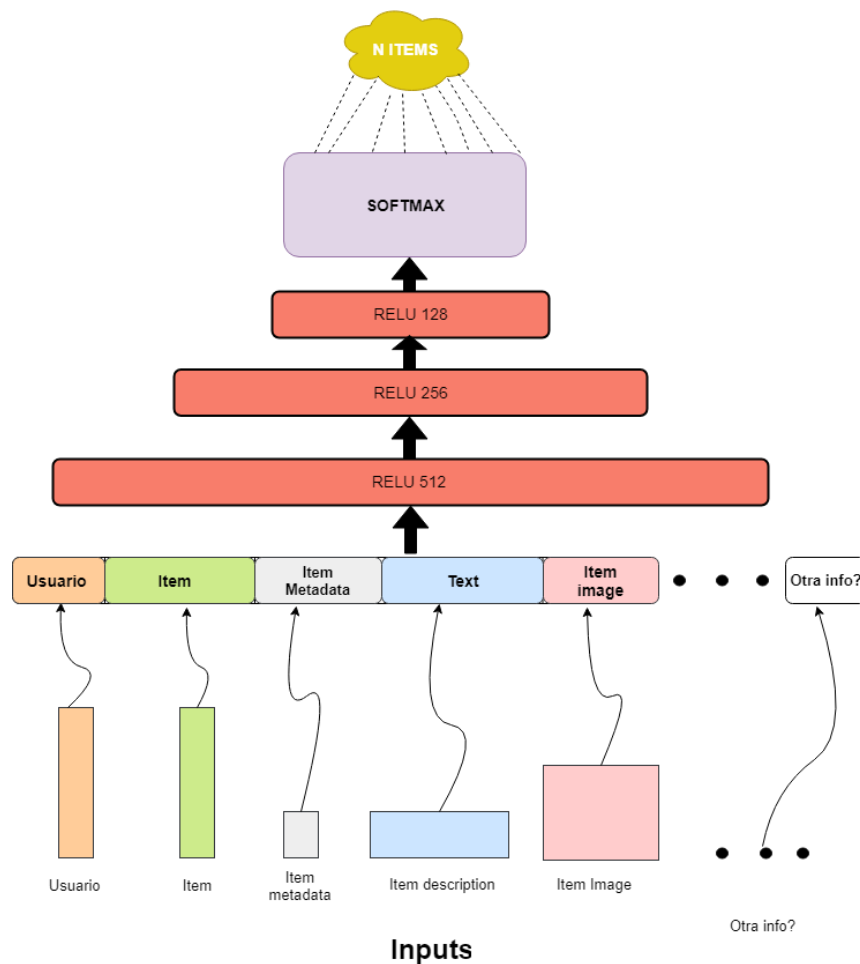
Una vez ya tenemos el WordEmbedding y los dos *Autoencoders* procedemos a crear el Modelo Recomendador.

Inspirado en la arquitectura del [recomendador de Youtube](#), como entrada recibe el historial de compras de los clientes con sus correspondientes artículos, que son recibidos por un Modelo que genera artículos candidatos de ser recomendados a cada cliente. Con un tamaño de dimensión mucho menor y un volumen mucho menor nos fijamos en las características más destacadas a través del modelo de Ranqueo que recibe los candidatos y los ordena de mayor a menor afinidad para el Cliente. De esta forma podemos manejar mejor los tiempos de entrenamiento a la vez que nos fijamos en todas las características de negocio que son importantes de cara al proceso de recomendación.



#### Modelo generador de candidatos:

El modelo **generador de candidatos** recibe como entrada el historial de compras de los clientes con sus correspondientes artículos, estos son mapeados junto con los clientes asociados a sus compras, así como los metadatos de esta interacción (Marca, cantidad de artículos, precio etc.) y la descripción del producto representada a través de nuestra representación vectorial que posteriormente nuestro *WordEmbedding* mapeará. Es importante destacar que los *Embeddings* de clientes y productos se aprenden conjuntamente con todos los demás parámetros del modelo a través de las actualizaciones de retro propagación.

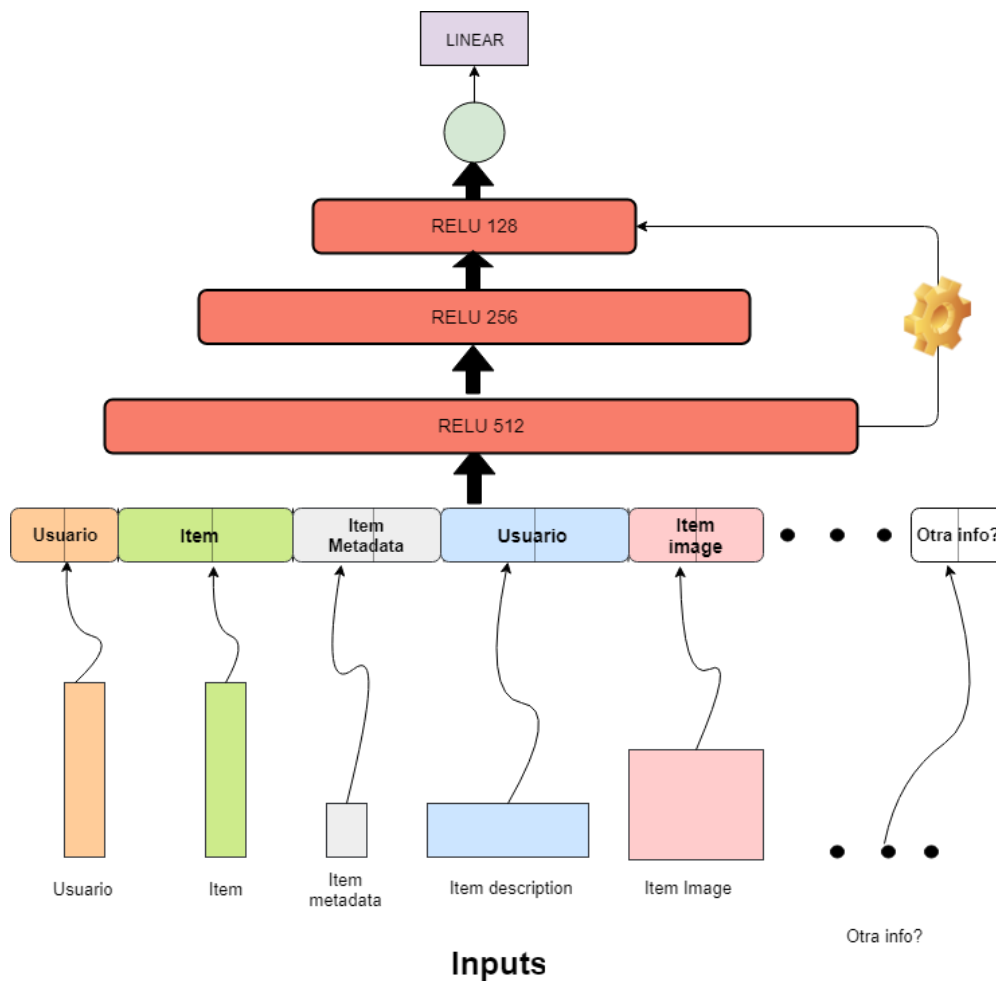


Todas estas características se concatenan en una amplia primera capa, seguida de varias capas densas, es decir, todas las neuronas de estas capas están conectadas con todas las neuronas de las capas adyacentes. Por último, encontramos una capa [Softmax](#) que recibe como entrada la última capa Densa y como salida cada uno de los productos que tenemos en 'Stock', el motivo por el cual uso una capa *Softmax* es porque necesitamos que la suma de todas las probabilidades de los productos sea 1, el valor de cada una de las salidas a su vez se encontrara en el rango [0,1].

#### Modelo de ranqueo:

Este modelo no siempre se debe utilizar dependiendo de la volumetría de clientes y artículos, bastaría con el generador de candidatos, pero dado que se ha probado en este proyecto se

detalla su arquitectura y funcionamiento. Al igual que el generador de candidatos, como input recibe las interacciones clientes-artículos y mapea mediante los embeddings todas las características.



Todas estas características son concatenadas en una primera capa, de manera más detallada de lo que lo hicimos en el modelo de candidatos. También aplicamos la técnica [Inception](#) que nos permite conectar capas superiores con inferiores para ganar musculo de aprendizaje dentro de nuestra red. A diferencia del modelo generador de candidatos, ahora disponemos de una única salida con una función de activación lineal que predice los días que faltan para que el cliente vuelva a comprar ese ítem. Como en este proyecto no disponemos de ese dato (Días transcurridos entre la compra y la fecha de referencia) hemos usado la cantidad de compras. (Usar las cantidades de artículos comprados crea un sesgo considerable hacia los artículos que se sustentan en un volumen mayor como puede ser la ropa, frente a otros

artículos de electrónica o electrodomésticos, por tanto, no se ha incluido en la demo final, aunque si se ha incorporado al código entregable, y por eso se deja constancia que en caso de disponer del dato de días transcurridos desde la compra habría que utilizarlo en este modelo). los outputs del modelo los usaremos para ranquear los ítems generados por el modelo de candidatos.

## Resultados y Conclusiones

Resultados:

Se han construido dos funciones de recomendación (una con el modelo de candidatos y otra con el de candidatos y el de ranqueo) y testado contra varios clientes observando cuanto se asemejaban las recomendaciones a su histórico de compras y se han resuelto varias cuestiones que más adelante voy a detallar.

La visualización de los resultados ha sido construida sobre un Webservice local usando la librería de Python **Flask**. El objetivo de este Webservice es poder presentar los resultados de una forma amigable y dinámica simulando recomendaciones reales.



## Customer 12 Recommendations

### Items bought by Customer 12:

- 1 - pantalon cropped mujer cintura alta
- 2 - vestido camisero print leopardo
- 3 - sombra gelcrema aqua xl color paint make ever exclusivo sephora
- 4 - perfilador labio pro sculpting make exclusivo sephora
- 5 - falda pantalon mujer raya
- 6 - abrigo hombre perimeter mte
- 7 - neceser mujer danielle nicole plateado parche minnie mouse
- 8 - vestido print leopardo cuello camisero
- 9 - vestido mujer verde jareta
- 10 - vestido mujer estampado sixties volante
- 11 - vaquero pitillo mujer bordado
- 12 - pantalon slim mujer cinturon
- 13 - pantalon recto mujer roberto verino bolsillo
- 14 - pantalon pitillo mujer pana
- 15 - pantalon pitillo mujer boutique moschino fucsia
- 16 - neceser mujer formula joven bordado oriental flor
- 17 - neceser mujer calvin klein plata cremallera
- 18 - abrigo hombre torrey hooded mte
- 19 - neceser mujer calvin klein negro cremallera
- 20 - jeggins mujer color liso strass

### Top 5 Items recommended to Customer 12:

- 1 - bufanda exclusiva spot navidad corte ingles franja bicolor rojo beige
- 2 - bota mujer piel color negro
- 3 - bailarinas mujer serraje color negro tacha
- 4 - bota unisex dr martens jadon black polished smooth
- 5 - vestido mujer verde jareta

Esta aplicación web recibe como parámetro de entrada un identificador de cliente y un número de recomendaciones comprendido entre 5 y 20, y produce como salida los 5 productos que nuestro modelo recomendaría para ese cliente, así como su histórico de compras para comprobar si las recomendaciones son adecuadas o no a ese cliente.

#### Limitaciones de los resultados:

La gran mayoría de los problemas surgidos en este proyecto que afectan a la calidad de los resultados viene por la indisponibilidad de datos de clientes. Generé las compras aleatoriamente pero no quise generar más información (Sexo, edad, fechas de compra...) aleatoria para no crear relaciones ficticias como que los hombres compren mucha ropa de mujer o que las personas jóvenes compren música antigua etc.

- Sesgo de los artículos más populares debido al filtrado colaborativo.
- Recomendaciones con sexo "dudoso" debido a que no se dispone del sexo del cliente, el modelo no puede aprender que las mujeres suelen comprar ropa de mujer y los hombres ropa de hombre.
- El modelo de ranqueo con las cantidades de compra se desestima debido a que crea un sesgo hacia los productos que se compran con más artículos, por ejemplo, ropa frente a electrónica o electrodomésticos.

- Se ha utilizado una muestra para el modelo compilado del 40% del conjunto final. Dados los tiempos de entrenamiento, lo consideré más adecuado.

#### Próximos pasos:

Obtener datos de clientes y hábitos de compra en caso de disponer de ellos y obtener datos de búsquedas de los clientes en caso de disponerlos.

Utilizar los días de compra como output en el modelo de ranqueo, estudiar funciones de coste customizadas que tengan en cuenta la recencia de los artículos, así como sus eficacias dentro de la compañía.

Crear un [generador batch](#) para procesar las imágenes dentro del modelo además de las otras características comentadas.

Realizar un EDA más amplio de las distribuciones de las medidas agregadas como el gasto de los clientes, la recencia la frecuencia etc.

#### Conclusiones:

En este proyecto se ha podido definir un flujo estándar para realizar un motor de recomendación en escenarios de comercio electrónico, tratando las diferentes etapas críticas del proceso como los orígenes de datos y la comprensión de los mismo, la definición de los casos de uso requeridos en función de las necesidades de negocio que defina el usuario.

Algunas etapas se han dejado abiertas debido a la naturaleza de este proyecto y la restricción de los datos de los clientes sin embargo se han dejado indicados los pasos a seguir y se ha hecho un estudio de las posibles mejoras o líneas de trabajo para desarrollarlas.