Predicción de Compra de Productos con Clustering y modelos RNN LSTM

Nelson Alfonso Beas Ham
Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticas

23 de noviembre de 2024

Palabras clave: predicción de compra, agrupamiento de productos, secuencias de compras, clasificación binaria, análisis de similitud, codificación binaria, regularización, reducción de dimensionalidad, aprendizaje supervisado, reducción de ruido, probabilidad de compra.

1 Resumen

El objetivo de este trabajo es desarrollar un modelo de predicción para la compra de productos en función de los artículos previamente adquiridos en un mismo ticket de compra. Para ello, se emplea un enfoque basado en el análisis de secuencias de productos y el uso de técnicas de agrupamiento y redes neuronales recurrentes (RNN) con Long Short-Term Memory (LSTM).

El proceso comienza con la preparación de los datos, donde se realiza un análisis de los productos y sus características utilizando técnicas como la vectorización TF-IDF y el agrupamiento KMeans para reducir la dimensionalidad de los datos. Posteriormente, se emplea un modelo de RNN LSTM para predecir la probabilidad de que un producto sea retirado de un ticket de compra dado un conjunto de artículos previamente comprados.

El modelo es entrenado con un conjunto de datos que incluye información sobre los productos y su aparición en distintos tickets de compra. Al final del proceso de entrenamiento, el modelo es capaz de identificar patrones en las compras y predecir qué productos probablemente serán comprados en función de los productos en un ticket.

Este modelo no solo predice qué productos fueron retirados del ticket de compra, sino que también puede utilizarse para predecir qué productos serán comprados dada una lista de artículos, proporcionando una herramienta útil para la recomendación y optimización de inventarios en el ámbito comercial.

2 Aplicaciones e Impacto en la Industria

La capacidad de predecir qué productos serán comprados en función de las compras previas tiene un gran potencial para la industria del retail y comercio electrónico. Las empresas que implementen este tipo de modelos pueden mejorar la personalización de las recomendaciones, ofreciendo productos adicionales que los clientes probablemente comprarán, lo que incrementa las ventas y la satisfacción del cliente.

Además, la predicción precisa de comportamientos de compra ayuda a optimizar la gestión de inventarios, permitiendo a las empresas ajustar sus existencias y evitar tanto el desabastecimiento como el exceso de productos en inventario. Los modelos de predicción también permiten anticipar la demanda de productos, facilitando la planificación de estrategias de marketing y promociones específicas.

En un mundo donde los comportamientos de los consumidores son cada vez más volátiles y complejos, las empresas necesitan herramientas avanzadas de análisis predictivo para mantenerse competitivas. El uso de modelos como el propuesto no solo mejora la eficiencia operativa, sino que también ofrece una ventaja estratégica al comprender mejor las decisiones de compra de los consumidores, lo que se traduce en un mejor servicio, reducción de costos y aumento de ingresos.

3 El Rol de las Técnicas de Machine Learning y datos masivos en la Construcción del Modelo

Las técnicas de *machine learning* (aprendizaje automático) son fundamentales para construir modelos capaces de realizar predicciones complejas sobre los comportamientos de los consumidores, como en el caso de la predicción de productos que serán comprados en función de artículos previos adquiridos. Sin el uso de estas técnicas avanzadas, sería extremadamente difícil abordar problemas de esta magnitud de manera eficiente y precisa.

En particular, las técnicas de clustering y redes neuronales recurrentes (RNN) como el LSTM (Long Short-Term Memory) son clave para manejar los datos secuenciales y no lineales típicos en problemas de recomendación y predicción de compras. El uso de algoritmos de agrupamiento, como KMeans, permite identificar patrones de comportamiento entre los productos de manera no supervisada, reduciendo la dimensionalidad y permitiendo encontrar relaciones complejas entre los productos que no son evidentes a simple vista. Esta capacidad de segmentar datos en grupos similares mejora significativamente la calidad de las predicciones al reducir el ruido y mejorar la relevancia de los productos recomendados.

Por otro lado, las redes neuronales recurrentes, y en particular el LSTM, son altamente efectivas en el procesamiento de secuencias de datos. Los LSTM son capaces de aprender y recordar patrones en secuencias de compras pasadas, capturando dependencias a largo plazo entre los productos adquiridos. Esto es

crucial en la predicción de compras, ya que los comportamientos de los consumidores no son simplemente independientes; las compras anteriores influyen significativamente en las decisiones futuras.

Sin el uso de técnicas de *machine learning*, como las mencionadas, los enfoques tradicionales para predecir comportamientos de compra se basarían en reglas simples o en modelos estadísticos básicos, que no podrían capturar la complejidad de las relaciones entre productos y las secuencias de compra. Los modelos basados en reglas o en métodos manuales requerirían una cantidad inmensa de trabajo para identificar patrones y serían incapaces de adaptarse de manera flexible a nuevas tendencias o comportamientos.

3.1 Uso de Apache Spark

En este proyecto se utilizó **Apache Spark**, una herramienta ampliamente reconocida por su capacidad para procesar grandes volúmenes de datos de manera eficiente. Spark es especialmente útil cuando el tamaño del conjunto de datos tiene un tamaño importante.

Spark opera mediante el uso de *datasets*, una estructura optimizada que combina las ventajas de las consultas tipo SQL con el poder de las transformaciones funcionales. Esto permite realizar operaciones complejas de preprocesamiento de datos de forma eficiente, incluso en sistemas con recursos limitados.

En este proyecto, Spark fue clave para manejar el conjunto de datos masivo de *Instacart Online Grocery Basket Analysis*. Las principales tareas de preprocesamiento realizadas incluyeron:

- **Limpieza de datos:** Eliminación de caracteres especiales, números y stopwords de los nombres de los productos.
- **Transformación de datos:** Creación de nuevas columnas, como versiones procesadas de los nombres, para facilitar la vectorización.
- **Escalabilidad:** Procesamiento eficiente de millones de registros mediante operaciones distribuidas.

El uso de Apache Spark no solo permitió superar las limitaciones de memoria al trabajar con datos de gran tamaño, sino que también simplificó significativamente las etapas de limpieza y transformación necesarias para el modelado posterior. Su capacidad para manejar tareas complejas de preprocesamiento con rapidez lo hace una herramienta indispensable en el análisis de datos a gran escala.

4 Datos y Preprocesamiento

En este proyecto, los datos utilizados consisten en una lista de productos y un conjunto de órdenes de compra. Cada producto tiene un identificador único (*product_id*) y un nombre (*product_name*). La siguiente tabla ilustra algunos ejemplos de productos en el dataset:

Product ID	Product Name
1	Chocolate Sandwich Cookies
2	All-Seasons Salt
3	Robust Golden Unsweetened Oolong Tea
4	Smart Ones Classic Favorites Mini Rigatoni With Vodka Cream Sauce
5	Green Chile Anytime Sauce
6	Dry Nose Oil
7	Pure Coconut Water With Orange

Table 1: Ejemplo de registros en la lista de productos.

En el contexto de este modelo, un **ticket de compra** se define como un conjunto de registros relacionados por un identificador único, el *order_id*. Cada fila representa un producto que fue añadido al carrito en un pedido específico. A continuación, se presenta un ejemplo de ticket, donde todos los productos están asociados al mismo *order_id*, lo que indica que forman parte de la misma compra:

Order ID	Order Product ID
1	49302
1	11109
1	10246
1	49683
1	43633
1	13176
1	47209
1	22035

Table 2: Ejemplo de ticket de compra sin columnas adicionales.

En este ejemplo, todos los productos están asociados al mismo *order_id*, indicando que pertenecen a la misma compra o ticket. Estos datos pueden ser utilizados para analizar qué productos fueron comprados juntos y cómo se relacionan entre sí en el contexto de un mismo pedido.

En este ejemplo, el campo *Reordered* indica si el producto ya había sido comprado previamente por el usuario (1) o si es un producto nuevo en la compra (0). Este tipo de información permite analizar patrones de consumo, como productos recurrentes o tendencias en el orden de adquisición dentro de un ticket.

El proceso de preprocesamiento de los datos involucró las siguientes etapas:

- **Conversión de formatos:** Los datos fueron cargados desde un archivo CSV, y las columnas clave (*product_id*) se convirtieron a tipos de datos apropiados para su manipulación.
- **Limpieza de nombres de productos:** Los nombres de productos fueron procesados para eliminar números, caracteres especiales y palabras irrel-

evantes (*stopwords*) como “with”, “and”, “of”, etc. Se normalizaron a minúsculas para una representación uniforme.

- **Vectorización de textos:** Utilizando TF-IDF, se generó una representación numérica de los nombres procesados, capturando la importancia relativa de las palabras más significativas.
- **Clustering de productos:** A partir de la representación vectorial, se aplicó el algoritmo K-Means para agrupar productos similares en 300 clústeres. Este paso permitió identificar relaciones implícitas entre los productos basándose en sus descripciones.
- **Mapeo de productos:** Cada producto procesado se asignó a un identificador virtual correspondiente a su clúster, lo que facilitó la integración de esta información en el modelo.

Processed Product Name	**Product Name**	**Virtual Product ID**
allseasons salt	All-Seasons Salt	256
chocolate sandwich cookies	Chocolate Sandwich Cookies	171
dry nose oil	Dry Nose Oil	295
peach mango juice	Peach Mango Juice	292
pure coconut water with orange	Pure Coconut Water With Orange	68
cut russet potatoes	Cut Russet Potatoes	266
green chile anytime sauce	Green Chile Anytime Sauce	11
light strawberry yogurt	Light Strawberry Yogurt	129
robust golden unsweetened...	Robust Golden Unsweetened Oolong Tea	206
smart ones classic favorites...	Smart Ones Classic Favorites...	12

Table 3: Ejemplo de productos procesados y asignaciones de clústeres.

Este enfoque permite transformar datos textuales en una estructura adecuada para su uso en modelos de aprendizaje automático, conservando información semántica y reduciendo la dimensionalidad del problema. Gracias a estas técnicas, fue posible agrupar productos de manera efectiva, lo que representa un paso clave para predecir comportamientos de compra.

5 Estructura de los Registros de Entrada para el Modelo

Cada registro de entrada que se utiliza para entrenar el modelo tiene la siguiente estructura. En este formato, se encuentran los datos esenciales para predecir qué productos fueron eliminados del ticket de compra, basándose en el historial de compras previas. A continuación se describe cada columna:

Campo	Valor
Order ID	1572
Virtual Product IDs	`[148, 167, 4, 229...]`
Product IDs	`[8022, 34969, 294...]`
Processed Product Name	`[spinach pizza, r...]`
Remove Count	1
Removed Order Product IDs	`[160]`

Table 4: Ejemplo de registro de entrada para el modelo.

5.1 Descripción de las Columnas:

- **Order ID:** El identificador único de la orden de compra. Cada pedido tiene un *order_id* que agrupa a todos los productos adquiridos en esa compra.
- **Virtual Product IDs:** Una lista de los identificadores virtuales de los productos. Estos identificadores corresponden a los productos agrupados en clústeres durante el proceso de preprocesamiento, lo que permite al modelo tratar los productos de manera más generalizada.
- **Product IDs:** Una lista de los identificadores reales de los productos en el pedido. Estos son los identificadores específicos de cada producto que se adquirió en la compra.
- **Processed Product Name:** Los nombres de los productos procesados. Estos nombres son versionados para eliminar palabras irrelevantes y estandarizar las entradas, lo que ayuda al modelo a aprender patrones de productos de manera más efectiva.
- **Remove Count:** El número de productos que fueron eliminados del pedido, es decir, productos que ya no forman parte de la compra finalizada, pero que podrían haber sido parte del proceso de compra inicial.
- **Removed Order Product IDs:** Una lista de los identificadores de los productos que fueron eliminados del carrito de compra en este pedido. Esto ayuda a identificar qué productos fueron finalmente retirados y es esencial para el entrenamiento del modelo.

5.2 Relación de los Datos con el Modelo:

El modelo utiliza estos registros de entrada para predecir qué productos fueron eliminados del ticket de compra. Es importante destacar que el orden de los productos no tiene importancia, ya que el modelo está diseñado para reconocer patrones sin necesidad de que los productos sigan un orden específico. Sin embargo, la lista de productos puede contener productos que fueron retirados del carrito en un punto determinado del proceso de compra. En este caso, los productos eliminados corresponden a los *n últimos productos* de la lista,

pero este proceso es equivalente a seleccionar un producto aleatorio de la lista desordenada.

Es decir, si bien el modelo recibe los productos en un orden aleatorio y desordenado, lo que importa es que se identifiquen correctamente los productos eliminados (en este caso, los *n últimos*) en relación con el total de productos en el carrito. Esta flexibilidad en el orden facilita el entrenamiento del modelo, permitiendo que se adapte a cualquier secuencia de compra sin necesidad de que los datos estén ordenados cronológicamente o por algún otro criterio.

Al analizar esta información, el modelo puede predecir las probabilidades de que un producto se elimine en futuras compras, en función de las características de los productos en el carrito y su historial de compras.

6 Clustering y Palabras clave

6.1 Similitudes dentro de un clúster

En el proceso de clustering, los productos son agrupados en función de sus similitudes, basadas en características textuales y contextuales. A continuación, se presenta una tabla que evidencia los productos agrupados dentro del clúster **141**, que incluye principalmente huevos y productos relacionados:

Producto
Jumbo grade eggs
Organic large grade eggs
Pastured eggs
Large brown grade AA eggs
Large pasteurized eggs
Large AA eggs
Eggs
Organic medium brown eggs
Brown extra large grade AA eggs
Large brown range eggs grade
Organic pasture-raised local eggs
Grade large white eggs
Whoppers Robin eggs
Brown eggs
Hard boiled eggs peeled ready to eat

Table 5: Productos agrupados en un mismo cluster.

El el ejemplo mostrado, el clúster agrupa productos relacionados principalmente con huevos en diversas presentaciones, incluyendo orgánicos, pasteurizados, cocidos y preparados. La creación de un macroproducto para este clúster facilita el entrenamiento del modelo al permitir una representación más general y robusta de los productos. Al unificar los productos bajo un macroproducto

común, se reducen la dimensionalidad y el ruido en los datos, lo que mejora la capacidad del modelo para identificar patrones relevantes. Este enfoque también simplifica la personalización de recomendaciones para los usuarios, ya que los productos del clúster cumplen necesidades similares dentro de una categoría alimenticia específica.

6.2 Importancia de las palabras en el modelo

En el contexto del clustering y la creación de productos virtuales, se identificaron las palabras más importantes. Estas palabras representan las características clave que definen a los productos en el corpus. La importancia de las palabras se calculó utilizando la métrica **TF-IDF** (Frecuencia de Término - Frecuencia Inversa de Documento). A continuación, se muestran las **10 palabras más importantes** y sus valores correspondientes:

Palabra	**Valor de Importancia**
Organic	1725.41
Chocolate	918.53
Cheese	844.72
Chicken	633.92
Sauce	578.21
Cream	552.75
Milk	500.96
Tea	488.37
Yogurt	485.58
Natural	468.87

Table 6: Palabras más importantes en el corpus y su relevancia.

6.3 Proceso de creación de la lista de palabras importantes

El cálculo de las palabras importantes en el modelo se realizó mediante los siguientes pasos:

- **Preprocesamiento de texto:** - Tokenización de los nombres de productos para dividirlos en palabras individuales. - Eliminación de palabras comunes o irrelevantes (*stopwords*). - Aplicación de *stemming* o *lemmatization* para unificar términos similares.

- **Cálculo de importancia:** Se utilizó la métrica **TF-IDF**:

$$\text{Importancia} = \text{TF} \times \text{IDF} \tag{1}$$

Donde: - **TF (Frecuencia del Término):** Indica cuántas veces aparece una palabra en un documento (producto). - **IDF (Frecuencia Inversa del Documento):** Mide la rareza de la palabra en el corpus completo.

- **Agregación por clúster:** - Se calculó la relevancia promedio de cada palabra dentro de cada clúster. - Las palabras con mayor relevancia fueron seleccionadas como representativas de los grupos.

6.4 Interpretación del valor de importancia

El valor calculado para cada palabra indica su **influencia relativa** en el corpus. Los valores altos están asociados a términos que son específicos de ciertos grupos y que diferencian claramente a los productos. Por ejemplo, la palabra *organic* tiene un valor de 1725.41, indicando su alta relevancia para describir productos relacionados con alimentos saludables.

7 Descripción del Modelo

7.1 Diseño del Modelo

El modelo utilizado es una red neuronal diseñada específicamente para predecir la probabilidad de que un producto sea retirado de una orden, basado en el historial de productos seleccionados. La arquitectura incluye los siguientes componentes:

- **Capa de Embedding**: Convierte los identificadores numéricos de los productos en representaciones vectoriales continuas. Esto permite que el modelo capture características latentes de cada producto.
- **LSTM (Long Short-Term Memory)**: Este componente procesó las secuencias de productos, capturando relaciones temporales y contextuales. La capacidad de LSTM para retener información a largo plazo lo hace ideal para este tipo de tarea secuencial.
- **Capa Completamente Conectada (Fully Connected)**: Genera una probabilidad para cada producto, indicando la posibilidad de que sea retirado de la orden.
- **Dropout**: Una técnica de regularización que reduce el riesgo de sobreajuste al desactivar de forma aleatoria una fracción de las neuronas durante el entrenamiento.

7.2 Entrenamiento del Modelo

El modelo fue entrenado utilizando un enfoque de aprendizaje supervisado. Para cada instancia de entrenamiento, el modelo recibió una secuencia de productos seleccionados y un vector binario que indicaba los productos retirados. Durante el proceso:

- Se calculó la pérdida entre las predicciones del modelo y las etiquetas reales utilizando una función diseñada para problemas de clasificación binaria.

- Se utilizó el optimizador Adam para ajustar los pesos del modelo de manera eficiente.
- Se implementaron varias épocas de entrenamiento para asegurar que el modelo pudiera generalizar bien sobre datos no vistos.

7.3 Evaluación del Modelo

La evaluación del modelo se realizó calculando la precisión en las predicciones, comparando los valores previstos con los reales. Para ello, las salidas del modelo se transformaron en valores binarios utilizando un umbral definido (por ejemplo, 0.5). La precisión se midió como el porcentaje de coincidencias entre las predicciones del modelo y las etiquetas reales.

7.4 Hiperparámetros

Se ajustaron varios hiperparámetros clave para optimizar el rendimiento del modelo, incluyendo:

- **Tamaño del Vocabulario**: Determinado por la cantidad total de productos únicos.
- **Dimensión de los Embeddings**: Representa la cantidad de características latentes asociadas a cada producto.
- **Dimensión de la Capa Oculta del LSTM**: Controla la capacidad del modelo para retener y procesar información contextual.
- **Tasa de Aprendizaje**: Regula la velocidad con la que el modelo actualiza sus pesos.
- **Número de Épocas**: Determina cuántas veces se procesó todo el conjunto de datos durante el entrenamiento.

8 Codificación Binaria de los Productos, optimizador y perdida

Para ilustrar cómo se codifican las salidas esperadas en este modelo, consideremos un ejemplo específico. Supongamos que tenemos un conjunto de productos representado por una lista de identificadores únicos de productos, y queremos codificar cuál de esos productos ha sido retirado.

8.1 Codificación Binaria de un Producto Específico

Ahora, supongamos que tenemos un ticket que contiene solo el *Producto 3* y este producto ha sido retirado. En este caso, la codificación binaria se realiza

asignando un valor de 1 al índice correspondiente al *Producto 3* y un valor de 0 a todos los demás productos.

La codificación binaria del ticket será entonces la siguiente:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & \dots \end{bmatrix}$$

Donde: - El valor de 1 en la tercera posición (y_2) indica que el *Producto 3* ha sido retirado. - Los valores de 0 en las demás posiciones indican que esos productos no han sido retirados.

En este caso, la longitud del vector binario será igual al tamaño del vocabulario de productos, es decir, al número total de productos posibles. Si, por ejemplo, hay 10 productos en total, la codificación binaria será un vector de 10 elementos, con solo el elemento correspondiente al *Producto 3* en 1, y los demás en 0.

8.2 Flexibilidad en la Longitud de la Secuencia

Este modelo es flexible en cuanto a la longitud del ticket, ya que puede manejar cualquier número de productos en la secuencia de entrada. Si el ticket contiene más productos, el vector binario se ajustará en consecuencia. Además, aunque la cantidad de productos varíe, el modelo puede procesar la secuencia sin necesidad de un tamaño fijo, gracias al uso de la arquitectura LSTM.

8.3 Función de pérdida y optimizador

Para entrenar el modelo, utilizamos la función de pérdida **BCEWithLogitsLoss** (Binary Cross-Entropy with Logits Loss). Esta función es adecuada porque el problema es de clasificación binaria, donde la salida deseada es un vector binario que indica qué productos fueron retirados. La función de pérdida se calcula como:

$$\mathcal{L}(y, \hat{y}) = -\frac{1}{N} \sum_{i=1}^{N} [y_i \log(\sigma(\hat{y}_i)) + (1 - y_i) \log(1 - \sigma(\hat{y}_i))]$$

donde:

- y_i es el valor objetivo (0 o 1) para el producto i.
- $\hat{y}_i$ es la salida del modelo antes de aplicar la función sigmoide.
- $\sigma(\cdot)$ representa la función sigmoide, que convierte las salidas en probabilidades entre 0 y 1.

Esto permite al modelo aprender una representación probabilística de los productos retirados.

Para optimizar los parámetros del modelo, utilizamos el optimizador **Adam**, que combina las ventajas de AdaGrad y RMSProp para un entrenamiento más eficiente. Su fórmula general para actualizar los parámetros es:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{v_t} + \epsilon} m_t$$

donde:

- η es la tasa de aprendizaje (*learning rate*).
- m_t es el promedio móvil de los gradientes.
- v_t es el promedio móvil de los cuadrados de los gradientes.
- ϵ es un pequeño valor para evitar divisiones por cero.

8.4 Proceso de validación de las predicciones

Durante la validación, el modelo genera un vector de probabilidades para cada producto. Para evaluar la precisión de las predicciones, seguimos el siguiente procedimiento:

1. Aplicamos la función sigmoide a la salida del modelo para obtener las probabilidades entre 0 y 1.
2. Identificamos el índice con la mayor probabilidad en las predicciones (*predicted*):
$$\text{predicted\_idx} = \arg\max(\sigma(\hat{y}))$$
3. Identificamos el índice correspondiente al producto retirado real (*target*):
$$\text{target\_idx} = \arg\max(y)$$
4. Comparamos si el índice predicho coincide con el índice real:
$$\text{Correcto} = \text{predicted\_idx} == \text{target\_idx}$$
5. Si los índices coinciden, contamos la predicción como correcta. En caso contrario, se considera incorrecta.

La precisión del modelo se calcula como el porcentaje de predicciones correctas sobre el total de predicciones:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}} \times 100$$

Este enfoque garantiza que solo se cuente como correcta la predicción donde el modelo acertó exactamente el producto retirado.

9 Resultados y Análisis

Los resultados obtenidos demuestran que el modelo propuesto logra identificar con precisión el macroproducto correspondiente en el 71% de los casos. Este porcentaje refleja un desempeño robusto considerando la complejidad del problema, que implica predecir el próximo producto a ser retirado en función de tickets de compra con longitudes y órdenes variables.

Además, los clústeres generados mediante la vectorización de los nombres de los productos y el posterior agrupamiento han demostrado ser efectivos. Los productos dentro de cada clúster presentan una alta similitud semántica, lo que confirma que los grupos representan categorías coherentes, como tipos de huevos, frutas o snacks, reduciendo significativamente el ruido presente en los datos iniciales. Este proceso de agrupación no solo mejora la calidad de los datos de entrada, sino que también permite optimizar el proceso de entrenamiento al enfocarse en patrones más consistentes.

El uso de macroproductos derivados de estos clústeres también fue clave para simplificar el problema, agrupando productos con nombres y características similares bajo una misma categoría. Esto redujo la dimensionalidad y facilitó el aprendizaje del modelo, permitiendo una representación más manejable de los datos sin perder información crítica para la tarea de predicción.

En conjunto, los resultados validan la capacidad del modelo y del enfoque basado en clústeres para abordar problemas de predicción en entornos con alta diversidad de productos. Sin embargo, se identifican oportunidades para seguir mejorando, como el análisis de relaciones más complejas entre los productos y su contexto de compra.

9.1 Visualización de Clusters

Para evaluar las agrupaciones generadas por el modelo, utilizamos la técnica de reducción dimensional **t-SNE** para visualizar los datos en dos dimensiones. La gráfica resultante muestra cómo los productos son distribuidos en función de sus características, coloreados por los clusters a los que pertenecen.

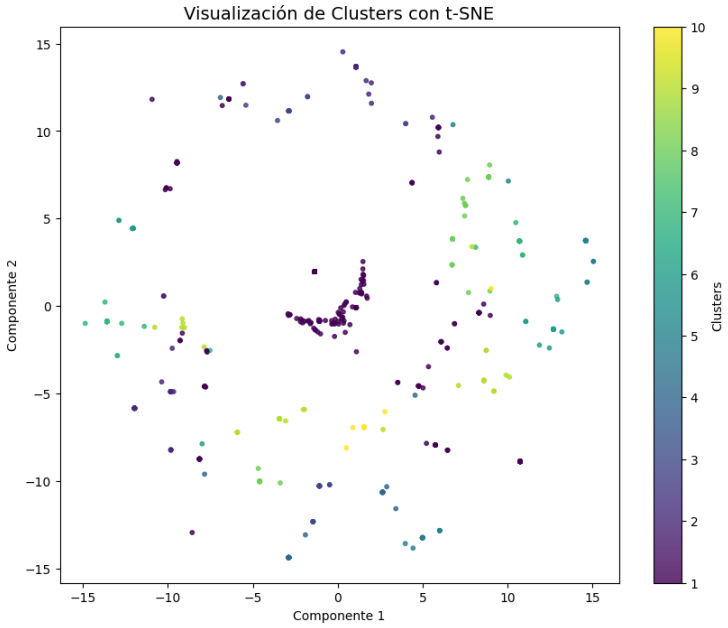

Figure 1: Visualización de clusters utilizando t-SNE. Cada punto representa un producto, y los colores indican el cluster al que pertenece.

Como se observa en la Figura 1, los productos similares tienden a agruparse, reflejando la eficacia del modelo para identificar patrones comunes en los datos.

9.2 Reducción de la Pérdida durante el Entrenamiento

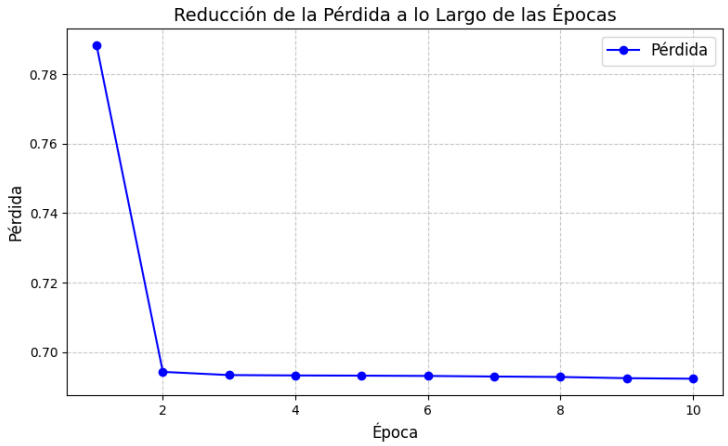

Figure 2: Gráfica de la pérdida durante el entrenamiento.

El comportamiento de la pérdida durante el entrenamiento se muestra en la Figura 2. Desde la primera época, se observa una disminución significativa en la pérdida. Esto se debe al tamaño considerable del conjunto de datos, lo que permite al modelo aprender patrones útiles rápidamente desde las primeras iteraciones.

A medida que avanza el entrenamiento, la pérdida se estabiliza, indicando que el modelo converge hacia una solución óptima. Este resultado evidencia que la arquitectura utilizada, junto con la función de pérdida *BCEWithLogitsLoss*, es adecuada para abordar la tarea de predicción de productos.

Aunque el descenso inicial de la pérdida es prometedor, podría evaluarse la incorporación de estrategias de optimización adicionales, como ajustes dinámicos de la tasa de aprendizaje, para mejorar aún más la convergencia en etapas posteriores.

10 Evolución del Rendimiento del Modelo

En la Figura 3 se muestra la evolución del *accuracy* del modelo a lo largo de 10 épocas de entrenamiento. Se observa un incremento inicial significativo en las primeras dos épocas, alcanzando un *accuracy* del 40%. Posteriormente, el crecimiento se estabiliza, mostrando mejoras más graduales hasta llegar a un valor final del 71% en la décima época.

Este comportamiento es esperado, ya que durante las primeras iteraciones el modelo ajusta rápidamente los pesos para aprender patrones básicos del conjunto de datos. Sin embargo, conforme avanza el entrenamiento, las mejoras se vuelven más lentas debido a la naturaleza compleja de los datos y a que el modelo comienza a ajustar detalles más finos para minimizar errores.

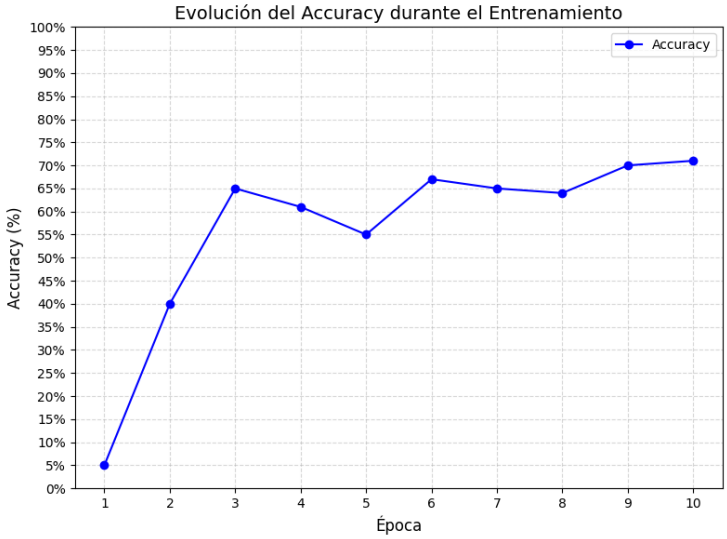

Figure 3: Evolución del *accuracy* durante el entrenamiento. Se observa un rápido incremento inicial seguido de una estabilización hacia el final.

11 Conclusiones y futuras mejoras

En este trabajo, hemos desarrollado un modelo de predicción para identificar los productos que serán retirados en un ticket de compra. A través de la utilización de un enfoque basado en redes neuronales recurrentes (RNN), específicamente una arquitectura LSTM, y una codificación binaria para representar la salida esperada, logramos entrenar un modelo que genera predicciones precisas de los productos retirados.

Los resultados obtenidos en el proceso de validación muestran un **alto porcentaje de precisión (accuracy)**, lo que indica que el modelo ha aprendido de manera efectiva a identificar los productos que serán retirados en función de las compras previas. Este buen desempeño es un indicativo de que el modelo es capaz de generalizar bien sobre los datos, lo que lo convierte en una herramienta útil para la predicción de productos retirados en escenarios de ventas reales.

Sin embargo, aunque los resultados son satisfactorios, siempre hay oportunidades para mejorar el rendimiento y la capacidad de generalización del modelo. Algunas posibles direcciones de mejora incluyen:

- **Ajuste de hiperparámetros:** Si bien se utilizaron valores estándar para algunos hiperparámetros, como la tasa de aprendizaje y la dimensión de la capa oculta, un ajuste más fino de estos valores mediante búsqueda en grid o técnicas de optimización bayesiana podría resultar en un modelo aún más preciso.

- **Mejora en la representación de productos:** En lugar de utilizar codificaciones binarias simples, se podrían explorar representaciones más sofisticadas para los productos, como embeddings preentrenados o características adicionales que capturen más información sobre los productos y sus relaciones con otros.
- **Ampliación de la base de datos:** El modelo podría beneficiarse de un mayor número de datos de entrenamiento, lo cual permitiría a la red aprender patrones más complejos. Además, una mayor diversidad en los datos, como productos de diferentes categorías o estacionales, podría ayudar a mejorar la robustez del modelo.
- **Incorporación de información temporal:** Dado que los productos retirados pueden tener una relación temporal con los tickets de compra, agregar características relacionadas con el tiempo, como la temporada o el comportamiento histórico del cliente, podría mejorar la capacidad del modelo para predecir retiros con más precisión.
- **Pruebas con otras arquitecturas:** Aunque las redes LSTM son útiles para procesar secuencias, probar con otras arquitecturas como Transformers o redes convolucionales recurrentes (CRNN) podría llevar a mejores resultados en términos de precisión y eficiencia computacional.

En resumen, el modelo propuesto ha demostrado un buen desempeño en la predicción de productos retirados, pero con ciertas mejoras en los aspectos mencionados anteriormente, es posible lograr una mayor precisión y robustez, lo que beneficiaría su aplicación en sistemas reales de recomendación y predicción de productos en el ámbito comercial.

12 Datos Utilizados

El conjunto de datos utilizado en este proyecto proviene de la empresa instacart y está disponible públicamente en el siguiente enlace: Instacart Online Grocery Basket Analysis Dataset. Este conjunto contiene información sobre productos, pedidos y clientes, proporcionando una base ideal para desarrollar sistemas de recomendación y análisis de patrones de consumo.