

## Resumen

En este informe se presenta un proyecto de *deep learning* aplicado al análisis del comportamiento de usuarios en un comercio electrónico. Todo el trabajo se desarrolló en Google Colab utilizando Python y TensorFlow, desde la carga de datos hasta el entrenamiento y evaluación del modelo. Se utilizó un dataset público de Kaggle llamado “E-commerce Behavior Data from Multi-Category Store”, el cual contiene más de un millón de registros de interacción de usuarios dentro de una tienda online. Entre estos eventos se encuentran visitas a productos, añadidos al carrito y compras, lo que permite analizar el proceso que sigue un cliente antes de decidir comprar.

El objetivo principal del proyecto fue predecir si un usuario realizará una compra basándose en su comportamiento dentro de una sesión. Para ello se entrenó una red neuronal profunda y luego se comparó su desempeño con modelos más simples, como la regresión logística. Antes del entrenamiento se realizó limpieza de datos, análisis exploratorio y generación de variables relevantes. Además, se tuvieron que aplicar técnicas para tratar el desbalance de clases, ya que la mayoría de usuarios solo navega y muy pocos compran. Los resultados mostraron una mejora clara frente al modelo base. El modelo logístico solo lograba identificar alrededor del 6% de las compras, mientras que la red neuronal logró detectar aproximadamente el 80%. Esto significa que el modelo profundo reconoce mucho mejor cuándo un usuario realmente va a comprar, aunque pierde un poco de precisión general. Aun así, la exactitud total se mantuvo alta (alrededor del 95%), cercana al baseline (97-99%), pero este último prácticamente ignoraba la clase de compra.

En conclusión, se logró construir un pipeline completo de aprendizaje profundo capaz de trabajar con datos reales a gran escala y generar un modelo predictivo útil para negocio. En las siguientes secciones se describen con más detalle la introducción, metodología, resultados y conclusiones del proyecto.

## Metodología

En esta sección se explica el proceso seguido para construir el pipeline de aprendizaje profundo, desde el tratamiento inicial de los datos hasta el entrenamiento y evaluación del modelo. La metodología se dividió en varias etapas: preprocesamiento de datos, análisis exploratorio, diseño del modelo de deep learning, entrenamiento y evaluación. A continuación, se detalla cada etapa.

### Preprocesamiento de datos

Los datos representan la secuencia de acciones que realizan los usuarios dentro de una tienda online durante un periodo de tiempo (correspondiente a octubre de 2019). Cada registro incluye información como:

- **Fecha y hora del evento**
- **Tipo de acción del usuario** (ver producto, añadir al carrito, quitar del carrito o comprar)

- **Identificador del producto**
- **Categoría del producto**
- **Marca**
- **Precio**
- **Usuario que realizó la acción**
- **Sesión del usuario (grupo de acciones continuas)**

Antes de entrenar el modelo, fue necesario limpiar y preparar los datos para que realmente sirvieran en el análisis.

Primero se eliminaron registros inválidos o incompletos, por ejemplo aquellos sin identificador de usuario, producto o fecha. También se descartaron valores sin sentido, como precios negativos o iguales a cero.

Luego se transformaron los tipos de datos. La fecha se convirtió a formato de tiempo para poder extraer variables nuevas como hora del día y día de la semana. Esto permite identificar patrones de comportamiento según el momento en que navega el usuario. Además, el precio se aseguró como valor numérico consistente.

Después se ordenaron los eventos por sesión y tiempo, de manera que cada interacción del usuario quedara en orden cronológico. Esto es importante porque el comportamiento del cliente depende de la secuencia de acciones que realiza.

Posteriormente se creó la variable objetivo del modelo. La idea fue predecir si una sesión termina en compra o no. Por ello, cada sesión se etiquetó como:

1 → si el usuario compró al menos una vez

0 → si solo navegó sin comprar

También se generaron variables resumen por sesión, por ejemplo:

- cantidad de productos vistos
- cantidad añadidos al carrito
- número de categorías visitadas
- duración total de la sesión

### **Análisis Exploratorio de Datos (EDA)**

Antes de entrenar el modelo se realizó un análisis exploratorio (EDA) para entender mejor los datos y validar algunas ideas iniciales, como el fuerte desbalance entre usuarios que navegan y los que compran.

Lo primero que se observó fue la distribución de eventos. La gran mayoría son solo vistas de productos (más del 95%), unos pocos añadidos al carrito (2-3%) y menos del 1% son compras. Esto confirma el típico comportamiento en e-commerce: muchos usuarios miran, pocos intentan comprar y muy pocos finalmente compran. Por ello, el modelo no podía evaluarse solo con exactitud, sino también con métricas como recall.

También se analizaron patrones de tiempo. La actividad es baja en la madrugada, sube durante el día y alcanza su punto máximo en la tarde-noche. Las compras ocurren en las mismas horas de mayor tráfico, lo que indica que existen “momentos de alta intención” donde los usuarios están más dispuestos a comprar.

A nivel de sesiones, se encontró que las sesiones con compra duran más tiempo y tienen muchas más interacciones. Es decir, quienes compran suelen navegar más productos y permanecer más tiempo en la página. Por eso variables como duración de sesión y número de vistas resultan importantes para el modelo.

Finalmente, se revisaron categorías y precios. Los productos electrónicos dominan el tráfico, pero algunas categorías con menos visitas presentan mayor proporción de compras. Además, los usuarios tienden a comprar productos ligeramente más baratos que los que solo observan.

### **Diseño del modelo de Deep Learning**

El problema se planteó como una clasificación binaria por sesión: usando la información resumida del comportamiento del usuario, el modelo debía predecir si esa sesión terminaría en compra (1) o no (0).

Aunque los datos originalmente eran secuenciales (una serie de acciones en el tiempo), se decidió trabajar con variables agregadas por sesión para simplificar el desarrollo. De esta forma se usó una red neuronal tipo *perceptrón multicapa (MLP)* en lugar de modelos más complejos como LSTM, ya que estos aumentarían mucho el tiempo de entrenamiento sin aportar demasiada ventaja inicial.

El modelo recibió entre 10 y 15 variables por sesión, por ejemplo: número de productos vistos, añadidos al carrito, categorías visitadas, duración de la sesión, hora del día y algunas características de marca o categoría. Para variables categóricas grandes se usaron embeddings, permitiendo que la red aprenda representaciones útiles sin generar miles de columnas.

### **Entrenamiento del modelo**

El entrenamiento del modelo se realizó en Google Colab utilizando GPU, lo que permitió acelerar bastante el proceso. El modelo se entrenó por lotes de 256 datos a la vez, buscando un equilibrio entre velocidad y estabilidad.

A medida que avanzaban las épocas, la pérdida iba bajando y la exactitud subía. En validación ocurrió algo parecido al inicio, pero después de unas 15 épocas dejó de mejorar, señal de que el modelo empezaba a memorizar los datos (sobreajuste). Por eso se aplicó early stopping, deteniendo el entrenamiento en el punto óptimo.

Las curvas de aprendizaje mostraron que el modelo sí aprendía correctamente: mejoró rápido al inicio y luego se estabilizó sin presentar problemas de falta de aprendizaje.

Debido al desbalance de clases, no solo se evaluó la exactitud sino también el recall y precision para la clase compra. El recall aumentó bastante (más de 0.75), lo que significa que

el modelo logró detectar la mayoría de compras. La precisión fue menor, algo esperado porque al intentar no perder compras aparecen más falsos positivos.

En este caso se priorizó el recall, ya que para el negocio es peor no detectar un posible comprador que marcar como comprador a alguien que finalmente no compra.

### **Evaluación y criterios de desempeño**

Una vez entrenado el modelo, se evaluó usando datos de prueba que el modelo nunca había visto. Esto permitió medir su desempeño real.

Se usaron varias métricas:

- Accuracy: qué porcentaje total de predicciones fueron correctas
- Recall: cuántas compras reales logró detectar el modelo
- Precision: de las compras predichas, cuántas sí ocurrieron
- F1-Score: equilibrio entre recall y precision
- Matriz de confusión: para ver errores y aciertos en detalle

Como comparación, la regresión logística (modelo base) tenía una exactitud muy alta (~99%) pero casi no detectaba compras (~5%).

Al balancearla mejoró a cerca de 45-50% de recall, aunque con más falsos positivos.

La red neuronal obtuvo resultados más útiles:

Accuracy  $\approx$  95%

Recall  $\approx$  80% (detecta 4 de cada 5 compras reales)

Precision  $\approx$  50-60%

Esto significa que el modelo detecta muchas más compras que el baseline, aunque marque como compradores a algunos usuarios que no lo son. En un contexto de negocio esto es aceptable, porque es peor perder un cliente potencial que ofrecerle algo a alguien que no comprará.

En general, el modelo profundo logró un mejor equilibrio entre métricas y superó claramente al modelo base.

En resumen, el modelo de deep learning logró mejorar la identificación de eventos de compra de manera notable en comparación con los baseline. Esto valida la hipótesis de que una arquitectura de red neuronal, combinada con las técnicas adecuadas (manejo de desbalance, features informativas, regularización), puede extraer patrones más sutiles del comportamiento de los usuarios que escapan a modelos lineales simples, permitiendo predecir con mayor acierto qué sesiones terminarán en conversión.

### **Resultados**

Discusión de resultados

Al comparar los modelos, la red neuronal funcionó mejor que los modelos simples. La regresión logística sin ajustes casi siempre predecía “no compra”, logrando mucha exactitud pero sin detectar compradores reales. Incluso al balancearla mejoró, pero no alcanzó el nivel del modelo profundo. Otros modelos como Random Forest tendieron a sobreajustar.

## **Conclusiones**

- Se pudo trabajar con datos masivos usando Python y Google Colab.
- El análisis exploratorio fue clave para entender el problema y elegir buenas variables.
- La red neuronal superó claramente a modelos tradicionales.
- El pipeline quedó reproducible y listo para usarse en escenarios reales.
- El modelo podría aplicarse para detectar clientes con alta intención de compra y realizar acciones como promociones o recomendaciones.
- Limitaciones y mejoras futuras
- Solo se utilizó una parte del dataset completo.
- No se probaron modelos secuenciales como LSTM que podrían mejorar resultados.
- En el futuro se podría predecir qué producto comprará el usuario o cuándo comprará.
- En resumen, el proyecto demuestra que el deep learning puede aportar valor real al negocio al anticipar compras y mejorar la toma de decisiones basada en datos.