ISSN 1870-4069

Análisis de patrones comerciales mediante técnicas avanzadas: Redes neuronales y análisis estadístico aplicado a datos públicos

Manuel Torres-Vásquez[1,2], Estefanía de-la-Cruz-Bautista[1], Cesar Alejandro Lara-Ramirez[1], Susana Chávez-Cruz[1]

[1] Universidad Mundo Maya, campus Villahermosa, Jefatura Académica de Arquitectura, Ingenierías y Computación, México

[2] Tecnológico Nacional de México campus Centla, División Sistemas Computacionales, México

{vlicd2231002@universidadmundomaya, vlicd2231001@universidadmundomaya}.edu.mx, susanachavez@umma.com.mx, manuel.torres@centla.tecnm.mx

Resumen. Este estudio analiza los patrones comerciales del Brazilian E-Commerce Public Dataset, un conjunto de datos público proporcionado por la empresa Olist. Para la predicción de ventas diarias, se aplicaron los modelos estadísticos ARIMA y Prophet, así como redes neuronales recurrentes con neuronas LSTM. Se llevó a cabo un preprocesamiento del conjunto de datos original, seguido del ajuste de los modelos, la evaluación de su desempeño y el análisis de errores. Los resultados indican que los modelos estadísticos presentan limitaciones, reflejadas en errores elevados en sus predicciones. En contraste, el uso de redes neuronales recurrentes mejoró significativamente la precisión de las predicciones. Esta diferencia se atribuye a la alta variabilidad de los datos, la posible presencia de patrones no estacionarios y la falta de componentes estructurales adecuados en los modelos estadísticos empleados.

Palabras clave: Series temporales, redes neuronales, predicción, comercio electrónico.

Analysis of Business Patterns Using Advanced Techniques: Neural Networks and Statistical Analysis Applied to Public Data

Abstract. This study analyzes the commercial patterns of the Brazilian E-Commerce Public Dataset, a public dataset provided by the company Olist. For the prediction of daily sales, ARIMA and Prophet statistical models were applied, as well as recurrent neural networks with LSTM neurons. Preprocessing of the original data set was carried out, followed by model fitting, performance evaluation and error analysis. The results indicate that the statistical models have

limitations, reflected in high errors in their predictions. In contrast, the use of recurrent neural networks significantly improved prediction accuracy. This difference is attributed to the high variability of the data, the possible presence of non-stationary patterns and the lack of adequate structural components in the statistical models used.

Keywords: Time series, neural networks, prediction, e-commerce.

1 Introducción

Anticipar la demanda de un producto puede marcar la diferencia entre el éxito y la ineficiencia operativa. En un mundo cada vez más impulsado por datos, el uso de Machine Learning para predecir ventas permite a las empresas optimizar la planificación de inventarios y mejorar la gestión de sus operaciones. A través de modelos avanzados de pronóstico, es posible analizar patrones de consumo y ajustar la disponibilidad de productos para satisfacer mejor las necesidades de los clientes. En este estudio, se analizan los patrones de distintos modelos aplicados a un conjunto de datos de ventas, evaluando sus limitaciones y determinando los factores que influyen en su rendimiento. Para ello, se emplean, un conjunto de datos públicos proporcionado por la empresa Olist del Brazilian E-Commerce Public Dataset [1]. Las series temporales son secuencias de datos organizadas de manera cronológica y equidistante en el tiempo, utilizadas en diversas áreas como el análisis del turismo en una región, el impacto del ruido urbano en la salud, el estudio de imágenes satelitales y la predicción de ventas para optimizar la logística de productos. Uno de los modelos más utilizados en el análisis de series temporales es ARIMA (Autoregressive Integrated Moving Average), el cual combina tres componentes principales: Autorregresivo (AR), Integrado (I) y Media Móvil (MA). Sus parámetros incluyen: p, que representa los retardos autorregresivos; q, que indica el componente de media móvil; y d, que define el orden de diferenciación necesario para estabilizar la serie [2].

Por otro lado, Prophet es un algoritmo de regresión aditiva desarrollado por Facebook, diseñado para modelar tendencias de crecimiento lineal o logístico por partes. Su enfoque se basa en la descomposición de los datos en componentes de tendencia, estacionalidad y efectos de días festivos. Prophet incluye un componente estacional anual, modelado mediante series de Fourier, y un componente estacional semanal, representado con variables ficticias. Sin embargo, su eficacia depende de la presencia de patrones bien definidos en los datos, lo que puede afectar su rendimiento en conjuntos de datos con alta variabilidad o ruido.

Finalmente, las redes neuronales artificiales (RNA) se inspiran en la estructura de las redes neuronales biológicas del cerebro humano y están compuestas por capas de nodos interconectados de manera jerárquica. Son especialmente útiles en problemas donde la formulación explícita de restricciones lógicas es compleja, como el reconocimiento de patrones y el análisis predictivo. Dentro de este campo, las Long Short-Term Memory (LSTM) representan una variante avanzada de las redes neuronales recurrentes, diseñada específicamente para mitigar el problema de la dependencia a largo plazo. Gracias a su arquitectura especializada, las LSTM pueden retener información durante períodos extensos, lo que las hace particularmente eficaces

	order_id	customer_id	order_status	order_purchase_timestamp	order_approved_at
0	5f79b5b0931d63f1a42989eb65b9da6e	00012a2ce6f8dcda20d059ce98491703	delivered	2017-11-14 16:08:26	2017-11-14 16:35:32
1	a44895d095d7e0702b6a162fa2dbeced	000161a058600d5901f007fab4c27140	delivered	2017-07-16 09:40:32	2017-07-16 09:55:12
2	316a104623542e4d75189bb372bc5f8d	0001fd6190edaaf884bcaf3d49edf079	delivered	2017-02-28 11:06:43	2017-02-28 11:15:20
3	5825ce2e88d5346438686b0bba99e5ee	0002414f95344307404f0ace7a26f1d5	delivered	2017-08-16 13:09:20	2017-08-17 03:10:27
4	0ab7fb08086d4af9141453c91878ed7a	000379cdec625522490c315e70c7a9fb	delivered	2018-04-02 13:42:17	2018-04-04 03:10:19

order_delivered_carrier_date	order_delivered_customer_date	order_estimated_delivery_date	order_item_id	product_id
2017-11-17 15:32:08	2017-11-28 15:41:30	2017-12-04 00:00:00	1.0	64315bd8c0c47303179dd2e25b579d00
2017-07-19 19:09:37	2017-07-25 18:57:33	2017-08-04 00:00:00	1.0	84183944dc7cddca87a5d384452c1d3c
2017-03-01 15:24:20	2017-03-06 08:57:49	2017-03-22 00:00:00	1.0	9df2b21ec85378d71df4404712e17478
2017-08-19 11:34:29	2017-09-13 20:06:02	2017-09-14 00:00:00	1.0	af3ec22cce878225aae6d9eb6c7a78eb
2018-04-04 18:11:09	2018-04-13 20:21:08	2018-04-18 00:00:00	1.0	868b3136c5b206f91b8208fbfdf2cb7c

Fig. 1. Fragmento del conjunto de datos unificado de los datos proporcionados por la empresa Olist, uniendo las tablas originales usando los campos clave.

en el procesamiento de datos secuenciales, como series temporales y conjuntos de datos discretos.

2 Trabajos relacionados

La literatura especializada confirma que los LSTM superan a los enfoques clásicos de pronóstico al capturar patrones estacionales, promociones y efectos de calendario en conjuntos de ventas reales [3]. Ellos demostraron que una red LSTM con funciones de costo personalizadas y optimización genética redujo los costos de inventario de una maderería mexicana.

En [4] validó el modelo en un contexto de comercio electrónico ruso, mejorando el MAPE frente a un ARIMA base en más del 20 %. Estos trabajos coinciden en que, al combinar normalización adecuada, dropout y búsqueda sistemática de hiperparámetros, las arquitecturas LSTM generalmente de una a tres capas con 32 a 128 unidades ofrecen un equilibrio sólido entre precisión y costo computacional para la planificación de ventas en distintos sectores como la manufactura, retail y el comercio electrónico [5].

3 Materiales y métodos

El presente estudio se basa en el conjunto de datos públicos Brazilian E-Commerce proporcionado por la empresa Olist. El dataset contiene información de 100,000 pedidos del 2016 a 2018 realizados en varios mercados de Brasil. Los datos incluyen múltiples tablas interrelacionadas con detalles de pedidos, Items vendidos, Pagos, Reseñas de clientes, Datos de clientes y Datos geoespaciales.

Para la red neuronal

- **Agregación temporal:** Se transformaron los datos a una representación de serie de tiempo diaria por categoría de producto. Específicamente, a partir de la fecha y hora de compra de cada pedido (order_purchase_timestamp), se extrajo la fecha y se contabilizó el número de productos vendidos por categoría en cada día. Agrupamos las ventas por fecha y por nombre de categoría de producto, y se calculó el total de ventas diarias por categoría (campo ventas_totales). De este modo, para cada categoría de producto obtuvimos una secuencia temporal que refleja cuántos productos de esa categoría se vendieron cada día. Cabe destacar que, si una categoría no tuvo ventas en cierto día, dicha fecha no aparecía en la agrupación inicial; para asegurar continuidad temporal, se podrían imputar valores cero en días sin ventas, garantizando series cronológicas completas. Cada serie resultante quedó identificada por su categoría de producto[6].

Características de temporalidad (estacionalidad): Con el fin de ayudar al modelo a captar patrones estacionales o repetitivos en las series (ciclos semanales, tendencias anuales), se agregaron variables sintéticas periódicas derivadas de la fecha. Para cada registro diario se calculó:

- una representación cíclica del año en curso mediante $\sin(2\pi t/\text{año})$ y $\cos(2\pi t/\text{año})$,
- una representación del mes dentro del año (ciclo anual subdividido en 12) mediante $\sin(2\pi t/\text{mes})$ y $\cos(2\pi t/\text{mes})$, y
- una representación de la semana del año mediante $\sin(2\pi t/\text{semana})$ y $\cos(2\pi t/\text{semana})$. Aquí t representa la fecha en formato numérico (p. ej., timestamp Unix) y los denominadores corresponden a la duración de un año, mes o semana en la misma unidad temporal. Codifican de manera continua la posición del día dentro de cada ciclo temporal, permitiendo al modelo discernir, por ejemplo, qué tan avanzado está el año o si un dato corresponde a inicios o fines de semana. Se eliminó la columna original de order_date del conjunto de datos, ya que en su forma de entero no aporta significado directo y sus efectos de temporalidad ahora están capturados por las nuevas variables.

Codificación de la categoría: A fin de que el modelo de redes neuronales distinga cuál es la categoría que está procesando, se añadió una codificación one-hot de la categoría de producto. En la representación tabular agregada, esto implicó crear columnas binarias para cada categoría única, inicializándolas en 0. Para cada serie de categoría, se marcó con valor 1 la columna correspondiente a esa categoría y 0 las de las demás categorías. Por ejemplo, la serie de la categoría toys tendrá un campo toys=1 en todas sus filas, mientras que health_beauty=0, baby=0, etc., y la serie de health_beauty tendrá health_beauty=1 y las demás 0, y así sucesivamente. Esta codificación sirve como entrada al modelo para indicarle explícitamente de qué categoría son los datos de esa secuencia.

('toys',

	order_date	ventas_totales	toys	health_beauty	baby	cool_stuff	\
0	1475452800	0.000000	1	0	0	0	
1	1475539200	0.067568	1	0	0	0	
2	1475625600	0.081081	1	0	0	0	
3	1475712000	0.027027	1	0	0	0	
4	1475798400	0.054054	1	0	0	0	

	bed_bath_table	sports_leisure	fashion_bags_accessories	pet_shop	\
0	0	0	0	0	
1	0	0	0	0	
2	0	0	0	0	
3	0	0	0	0	
4	0	0	0	0	

	garden_tools	furniture_decor	telephony	housewares	consoles_games	'
0	0	0	0	0	0	
1	0	0	0	0	0	
2	0	0	0	0	0	
3	0	0	0	0	0	
4	0	0	0	0	0	

Fig. 2. Fragmento del conjunto de datos de entrenamiento previo a la inclusión de las características de temporalidad y su segmentación en ventanas temporales

Normalización de valores: Dada las variaciones significativas en el volumen de ventas entre categorías, donde algunas presentan valores máximos diarios sustancialmente mayores, se aplicó una normalización min-max a cada serie de ventas de forma independiente. Utilizando la función MinMaxScaler de la biblioteca scikit-learn, el campo ventas_totales de cada serie fue escalado al rango [0,1]. Este procedimiento permite expresar cada serie temporal en términos de la proporción respecto a su propio valor máximo, evitando que las categorías con mayores volúmenes absolutos sesguen el proceso de aprendizaje del modelo. Finalmente, en la tabla procesada, el campo ventas_totales normalizado reemplazó el conteo original de ventas diarias, asegurando una representación homogénea entre categorías.

Segmentación en ventanas temporales: Los datos fueron preparados en formato de series de tiempo, con ventas diarias normalizadas, indicadores de categoría y atributos estacionales, completando así el conjunto de datos. Se procedió a construir los ejemplos de entrenamiento para la red neuronal en forma de secuencias. Se definió una ventana de entrada de 90 días y una ventana de predicción (salida) de 30 días, valores escogidos con base en la granularidad del negocio (un horizonte de un mes para previsión) y experimentación preliminar. A partir de cada serie diaria por categoría, se extrajeron pares de secuencia de entrada. Cada secuencia de entrada es una matriz de 90 filas (días) por N columnas de características. En este caso, las columnas incluyen:

- la venta diaria normalizada (ventas_totales escalada) de ese día,
- el indicador one-hot de cada categoría (71 columnas, de las cuales solo una tendrá valor 1 y las demás 0, determinado por la categoría de la serie), y
- las 6 columnas de seno/coseno anuales, mensuales y semanales para ese día. En total, la dimensión de características N es 1 (venta) + 71 (categorías) + 6 (estacionalidad) = 78 características por día. La etiqueta o secuencia de salida correspondiente es un vector de 30 valores que representa las ventas normalizadas de los 30 días siguientes (misma serie y categoría). Este procedimiento de ventaneo deslizante se repitió para todas las series de todas las categorías, generando un amplio conjunto de pares entrada-salida. Como resultado final del preprocesamiento, se obtuvo un dataset de aprendizaje compuesto por x

(secuencias de 90×78) y y (vectores de 30×1) que reúne información de todas las categorías de producto.

Para la construcción y entrenamiento del modelo de red neuronal, se utilizó Python 3 como lenguaje de programación. La manipulación de datos, la combinación de tablas y agregaciones por fecha se realizó con Pandas, mientras que NumPy se aplicó para las operaciones numéricas de bajo nivel. El preprocesamiento incluyó la normalización Min-Max y la división del conjunto de datos en entrenamiento y prueba, utilizando la librería sk-learn.

La implementación de la red neuronal recurrente se llevó a cabo con TensorFlow v2 y su API de alto nivel Keras, utilizando capas LSTM y rutinas de entrenamiento optimizadas para GPU. El entrenamiento se ejecutó en un entorno Kaggle con aceleración mediante una GPU NVIDIA Tesla T4, aplicando la estrategia de distribución tf.distribute.MirroredStrategy para la paralelización del cálculo. Este enforque permitió entrenar eficientemente el modelo a pesar del gran volumen de secuencias generadas[7].

Se empleó una red neuronal recurrente LSTM de una sola capa. La arquitectura del modelo consta de una capa LSTM con 32 unidades de memoria tal como Deng en [3] señala que "cuando el número de neuronas ocultas de un LSTM es pequeño, existen ventajas como una alta eficiencia computacional y la prevención del sobreajuste". Estas capas reciben secuencias de entrada de 90 pasos temporales con 78 características por paso. Esta capa procesa la secuencia completa y genera un vector de estado interno de tamaño 32 al final (return_sequences=False), extrayendo únicamente la salida del último paso.

La representación interna obtenida se transfiere a una capa Dense totalmente conectada con 30 neuronas de salida lineales, donde cada neurona predice las ventas normalizadas para un día en la ventana de salida de 30 días. Los pesos iniciales de esta capa se establecieron en cero para generar predicciones iniciales neutras, cercanas al promedio de la escala. No se aplicó función de activación en la capa de salida, ya que el modelo realiza una regresión sobre valores continuos en el rango [0,1].

El modelo se compiló utilizando el Error Cuadrático Medio (MSE) como función de perdida, adecuada para problemas de regresión continua, y el Error Absoluto Medio (MAE) como métrica de evaluación. Se empleo el optimizador Adam, un algoritmo de descenso de gradiente estocástico adaptativo ampliamente utilizado por su eficiente convergencia.

Para prevenir el sobreajuste, se implementó la técnica de early stopping, configurando un monitoreo de la pérdida en el conjunto de validación. El conjunto de datos se dividió aleatoriamente en 90% para entrenamiento y 10% para prueba mediante train_test_split (test_size=0.1). Este mismo 10% se usó como conjunto de prueba final y como validación durante el entrenamiento.

El modelo se entrenó por un máximo de 20 épocas, con un tamaño de lote predeterminado por TensorFlow. En cada época, el modelo procesó todas las secuencias de entrenamiento y ajustó sus pesos para minimizar la pérdida MSE. Aunque early stopping podía interrumpir el entrenamiento antes de las 20 épocas si la pérdida en validación dejaba de mejorar, en la práctica, el modelo completó todas las iteraciones, ya que la pérdida siguió disminuyendo gradualmente. Finalmente, el modelo ajustado se guardó para su posterior evaluación en el conjunto de prueba.

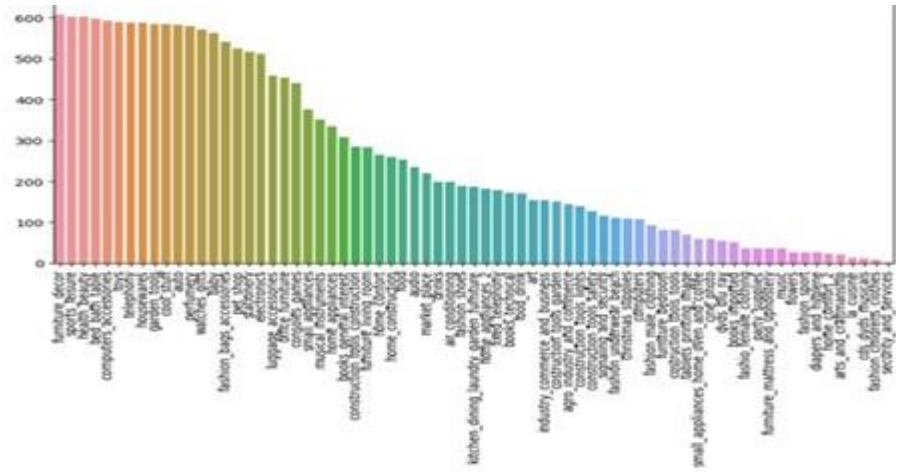

Fig. 3. Visualización por categorías la cantidad de ventas totales, ordenado por la categoría más vendida, a lo largo del periodo de tiempo que presenta el conjunto de datos.

El modelo ARIMA fue construido y ajustado utilizando Python 3, la manipulación de datos, la conversión de fechas de formato Unix a datatime y la agrupación de registros diarios se empleó Pandas. Las operaciones numéricas de bajo nivel aplicamos NumPy. La implementación del modelo se realizó con la biblioteca statsmodels, que facilita la definición, ajuste y evaluación de modelos de series temporales[2].

Inicialmente, se transformó la columna de fechas y se agregaron las ventas diarias para conformar la serie temporal. La estacionariedad de la serie fue evaluada mediante la prueba Dickey-Fuller Aumentada (ADF), cuyo p-valor inferior a 0.05 confirmó que la seria era estacionaria y adecuada para el ajuste directo del modelo. La selección de los parámetros óptimos (p, d, q) se realizó con base en los criterios de información AIC y BIC. Todos los coeficientes resultaron estadísticamente significativos ($p < 0.001$), como en los términos AR (1) = +0.8095 y AR(2) = –0.7992, que indican oscilaciones significativas en la serie, acompañadas por otros componentes MA. Una vez ajustado el modelo con los datos históricos, se generaron predicciones para un horizonte temporal definido[8].

La varianza estimada del error ($\sigma^2 \approx 3877$) fue alta, y los diagnósticos de residuos revelaron problemas importantes: el test de Jarque-Bera reportó un valor extremo (JB ≈ 947600, $p \approx 0.00$), evidenciando que los errores no siguen una distribución normal. Además, se detectó heterocedasticidad (H de White ≈ 3.22, $p \approx 0.00$), lo que implica que la varianza de los residuos cambia en el tiempo.

Los resultados muestran que el desempeño del modelo fue moderadamente bueno, arrojando los siguientes resultados:

- MAE ≈ 30.8,
- MSE ≈ 3715.1,
- RMSE ≈ 60.95,
- MAPE ≈ 31.7%.

El MAPE obtenido lo ubica en la categoría "razonable" (25–50%) según la

clasificación en [9]. Si bien este valor no alcanza el nivel “bueno” (<25%), se alinea con reportes previos sobre modelos de ventas con MAPE del 17–20% [8]. Concluimos que el modelo captura la tendencia general de la serie, aunque no predice con alta precisión los extremos.

Para el modelo Prophet utilizamos Python 3, ocupamos las bibliotecas Pandas y NumPy para la manipulación y transformación de datos. La modelización se llevó a cabo con la biblioteca Prophet, diseñada para ajustar modelos de regresión aditiva que descomponen las series temporales en componentes de tendencia, estacionalidad y efectos de días festivos o eventos especiales[10]. Para la preparación de la serie temporal, las columnas fueron renombradas a ds (fecha) y y (ventas), conforme a los requisitos del modelo. Tras el ajuste del modelo con los datos históricos, se generó un conjunto de datos con fechas futuras para obtener predicciones a corto y mediano plazo. La capacidad del modelo para capturar patrones estacionales se evaluó mediante la visualización de sus componentes, incluyendo la tendencia general y la estacionalidad semanal y anual. Esto facilitó la interpretación de la dinámica de ventas y la cuantificación de la incertidumbre en las predicciones.

Sin embargo, el modelo mostró un desempeño muy deficiente. Como puede observarse en la figura, la línea azul de predicción no sigue adecuadamente los puntos reales, especialmente en los picos y valles. Prophet tiende a suavizar excesivamente la serie, lo que reduce su capacidad para capturar la alta variabilidad. Al igual que en el modelo ARIMA, se aplicaron las métricas; MAE, MSE, RMSE y MAPE para evaluar la precisión del modelo. Las métricas in-sample reflejan esta falta de precisión:

- MAE ≈ 39.98,
- MSE ≈ 5495.69,
- RMSE ≈ 74.13,
- MAPE ≈ 93.8%.

Este MAPE supera ampliamente el 50%, umbral como indicador de un modelo "poco confiable". Las posibles causas de este bajo rendimiento incluyen:

- Ausencia de regresores adicionales que expliquen mejor la variabilidad.
- Falta de ajuste de hiperparámetros.
- Comportamientos en la serie que no se explican bien por las estacionalidades estándar que Prophet incorpora.

Preprocesamiento de datos para los Modelos ARIMA y Prophet:

- Conversión de Fechas y Agrupación: La columna order_date, originalmente en formato Unix timestamp, fue convertida a datetime para facilitar su interpretación. Posteriormente, las ventas diarias fueron agregadas a nivel global y por categoría, generando series temporales estructuradas.
- Verificación de Estacionariedad: En el modelo ARIMA, la estacionariedad de la serie temporal fue evaluada mediante la prueba de Dickey-Fuller Aumentada (ADF), obteniendo un p-valor de 0.035. Este resultado indica que la serie es estacionaria y no requiere diferenciación adicional para su modelado [2].

4 Procedimiento experimental

El análisis de datos y la predicción de ventas se realizaron mediante dos enfoques principales: un modelo de redes neuronales LSTM y un enfoque estadístico utilizando los modelos ARIMA y Prophet. Pasos implementados para cada caso:

1. **Obtención de los datos:** El conjunto de datos utilizado corresponde al dataset público Olist. Se importaron al entorno de trabajo las tablas relevantes, incluyendo información sobre pedidos, ítems de pedidos, pagos, reseñas de clientes, productos, traducción de nombre de categoría y clientes.

2. **Integración y preparación de datos:** Para la construcción del conjunto de datos, se integraron múltiples tablas mediante operaciones de merge, consolidando la información en una única tabla unificada. Utilizando esta tabla, se realizaron diversas transformaciones para estructurar series temporales por categoría de producto:

- Se generó una columna de fecha diaria a partir del timestamp de compra, eliminando la información horaria.
- Los datos fueron agrupados por order_date y product_category_name_english, calculando el número total de ventas por día y categoría. Esto dio lugar a una serie temporal donde cada fila representa el total de ventas de una categoría en un día específico.
- La tabla agregada fue reestructurada para incluir una columna binaria por cada categoría única. En cada fila, solo la columna correspondiente a la categoría del registro se marcó con 1. Las demás se mantuvieron en 0.
- Se añadieron características estacionales mediante funciones seno y coseno para capturar ciclos anuales, mensuales y semanales. La columna de fecha original fue eliminada.
- Para la preparación de datos de entrada a la red neuronal, se generaron secuencias temporales de longitud fija. Se aplicó una ventana deslizante de 90 días sobre cada serie temporal normalizada, utilizando estos 90 días como entrada (X) y los 30 días siguientes como salida esperada (y).
- Finalmente, todas las secuencias de todas las categorías fueron recopiladas en dos arreglos denominados x_unificado e y_unificado, obteniendo aproximadamente 11,670 secuencias. Tras la división en conjuntos de entrenamiento y prueba (90/10), se dispuso de alrededor de 1,503 secuencias para entrenamiento y 1,167 para prueba.

3. **Modelo de Redes Neuronales:** Se diseño e implementó una red neuronal recurrente LSTM. La implementación se realizó con Keras, utilizando tf.keras.Sequential para definir el modelo, al cual se incorporaron dos capas: una LSTM con 32 unidades y una capa Dense con 30 neuronas. El entrenamiento se llevó a cabo con un tamaño de lote predeterminado y se aplicó la estrategia de distribución MirroredStrategy para optimizar el uso de la GPU. El modelo fue compilado con el optimizador Adam, la función de pérdida MSE y la métrica MAE para el monitoreo del desempeño.

El modelo LSTM fue entrenado utilizando la función model.fit, con un máximo de 20 épocas, procesando el conjunto de entrenamiento en cada iteración completa. Se

```
DATOS DE ENTRENAMIENTO
Valor de pérdida: 0.01188591867685318, Valor de métrica (MAE): 0.07283851504325867
DATOS DE PRUEBA
Valor de pérdida: 0.012535602785646915, Valor de métrica (MAE): 0.07460156828165054
37/37 ━━━━━━━━━━━━━━━━━━━━ 1s 12ms/step
```

Fig. 4. Resultados de la evaluación del modelo con los datos de prueba y entrenamiento a través del método model.evaluate().

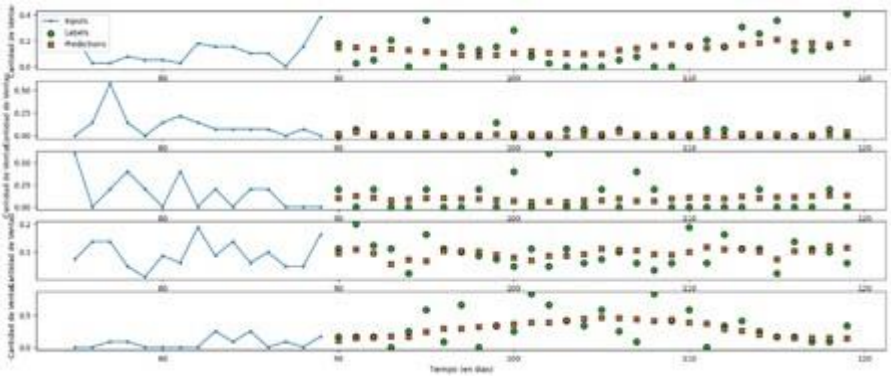

Fig. 5. Predicciones generadas con los datos de prueba. La línea azul punteada representa los últimos 20 datos de entrada, los puntos verdes representan las series temporales reales consecutivas, y las cruces naranjas son las predicciones generadas por el modelo.

empleo el subconjunto de prueba como validation_data para monitorear el desempeño en datos no utilizados en el ajuste. Para prevenir el sobreajuste, se implementó la estrategia de early stopping. Durante el proceso, los pesos del modelo fueron ajustados mediante backpropagation through time (BPTT) después de cada lote de secuencias. El entrenamiento se extendió hasta completar las 20 épocas, ya que la condición de parada anticipada no se activó, observándose una reducción progresiva de la pérdida de validación (val_loss), que alcanzó un valor aproximado de ~0.0125 al finalizar.

Una vez realizado el entrenamiento, se evaluó el desempeño del modelo utilizando tanto el conjunto de prueba como el de entrenamiento. Para ello, se suministraron al modelo las secuencias de entrada (x_pru) de 90 días, previamente no utilizadas en el ajuste, y se compararon las predicciones generadas para los 30 días siguientes (y_pred) con los valores reales correspondientes (y_pru). La evaluación cuantitativa se realizó mediante la función model.evaluate().

También, se generaron predicciones específicas utilizando la función model.predict sobre las secuencias de prueba, con el objetivo de visualizar y analizar los resultados. A partir de estas predicciones, se elaboraron gráficos comparativos entre las series de valores reales y los valores estimados para un subconjunto de categorías seleccionadas.

Para facilitar este análisis, se implementó la función auxiliar graficar_datos, la cual permite visualizar, para n casos de prueba, los últimos días de la ventana de entrada junto con la evaluación real futura y la estimada por el modelo. Este análisis visual complementa las métricas numéricas, proporcionando una mejor comprensión del desempeño del modelo en distintos escenarios.

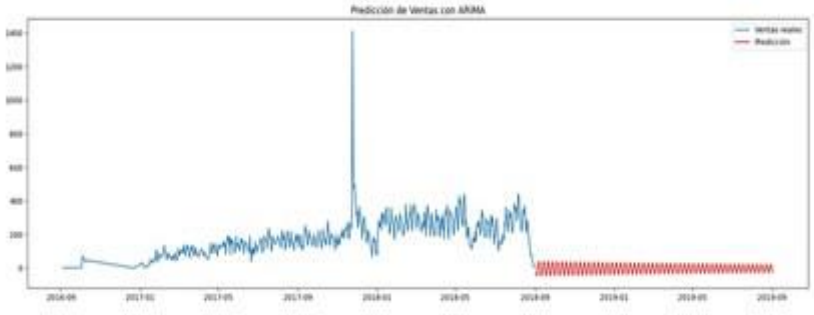

Fig. 6. Predicción de las ventas esperadas por Olist con el modelo estadístico ARIMA: La línea azul representa los valores reales de ventas, mientras que la línea roja muestra las predicciones generadas para un horizonte extendido.

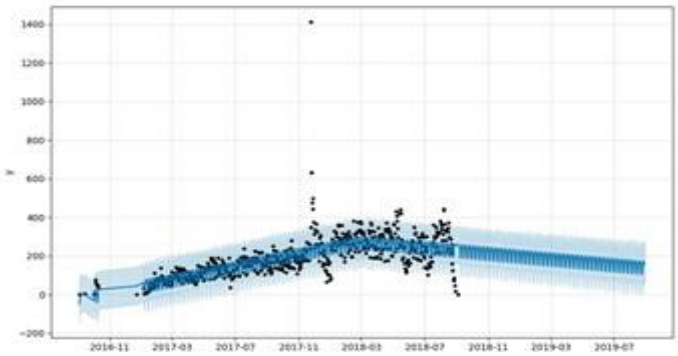

Fig. 7. Predicción de ventas diarias utilizando el modelo Prophet. Los puntos negros representan las observaciones reales, mientras que la línea azul indica la tendencia estimada y las bandas sombreadas corresponden a los intervalos de confianza.

4. **Modelos Estadísticos:** De forma paralela, se abordó la predicción de ventas utilizando dos métodos estadísticos:

ARIMA:

- Se transformó la columna order_date a formato datetime y se agrupó la serie de ventas diarias.
- Se aplicó la prueba de Dickey-Fuller Aumentada (ADF), la cual arrojó un p-valor inferior a 0.05, indicando estacionariedad y permitiendo el ajuste directo del modelo.
- Con criterios de información (AIC y BIC), se seleccionaron parámetros óptimos y se ajustó el modelo con la biblioteca statsmodels.
- Se generaron predicciones para un horizonte definido por 365 días y se evaluó el desempeño utilizando métricas: MAE, MSE, RMSE y MAPE, complementando el análisis mediante gráficos comparativos de las series reales y las predichas.

Prophet:

- Se preparó la serie temporal renombrando las columnas a 'ds' (fecha) y 'y' (ventas_totales) según los requerimientos del modelo.
- El modelo Prophet se ajustó a la serie histórica y se generó un conjunto de datos de fechas futuras para obtener predicciones a corto y mediano plazo (hasta 360 días).
- Se analizaron los componentes del modelo, lo que permitió interpretar los patrones subyacentes en las ventas y cuantificar la incertidumbre en las predicciones.
- Se calcularon métricas de error MAE, MSE, RMSE y MAPE para evaluar la precisión del modelo.

5 Resultados y discusión

Una vez realizado el entrenamiento del modelo de redes neuronales, se obtuvo en el conjunto de prueba, en la escala normalizada, un MAE de aproximadamente 0.0746, un MSE de aproximadamente 0.0125, un RMSE de 0.1125 y un SMAPE de 67.4727%. Estos resultados sugieren que, a pesar de la alta variabilidad en la serie temporal, el modelo logró capturar la dinámica de las ventas de manera efectiva. Además, la similitud entre los errores obtenidos en los conjuntos de entrenamiento y prueba indica una adecuada capacidad de generalización, sin evidencias de sobreajuste.

La figura 5 presenta ejemplos de predicción de ventas diarias para cinco casos del conjunto de prueba. En estos gráficos, la línea azul punteada representa los últimos 20 días de la ventana de entrada, los puntos verdes corresponden a las ventas reales de los 30 días posteriores y las cruces naranjas indican las predicciones generadas por el modelo. Los resultados muestran que la red neuronal logra capturar de manera efectiva la dirección de las tendencias y los patrones estacionales de las series temporales. No obstante, se identifican leves desviaciones en algunos casos, como la subestimación de picos de demanda o un ligero adelantamiento en la detección de caídas abruptas en las ventas.

En contraste, los métodos estadísticos presentaron mayores dificultades para modelar la compleja dinámica de la serie temporal. En particular, el modelo ARIMA, mostró un desempeño inferior en comparación con la red neuronal. Se obtuvieron errores estadísticamente significativos más altos, con un MAE de 30.792, un MSE DE 3714.466, un RMSE de 60.946 y un MAPE de 31.71%, de igual forma la varianza estimada del error ($\sigma^2 \approx 3877$) fue alta, y los diagnósticos de residuos revelaron problemas importantes: la prueba de Jarque-Bera reportó un valor extremo (JB ≈ 947600, $p \approx 0.00$), evidenciando que los errores no siguen una distribución normal. Además, se detectó heterocedasticidad (H de White ≈ 3.22, $p \approx 0.00$), lo que implica que la varianza de los residuos cambia en el tiempo.

La figura 6 muestra la predicción de ventas realizada con ARIMA, donde la línea roja, que representa las predicciones, presenta desviaciones notables respecto a la línea azul, que indica los valores reales. Estas diferencias reflejan las limitaciones del modelo para capturar cambios abruptos y adaptarse a la alta volatibilidad de la serie temporal.

El modelo Prophet también mostró un desempeño inferior, registrando un MAE de 39.984, un MSE de 5495.686, un RMSE de 74.133 y un MAPE de 93.80%. Como se observa en la figura 7, aunque el modelo capta la tendencia general (línea azul) y

muestra los intervalos de confianza, las predicciones presentan errores notables en la estimación de picos y valles, lo que indica dificultades para ajustarse a la alta variabilidad de las ventas diarias.

Comparando ambos enfoques, se observa que el modelo ARIMA entregó un desempeño significativamente mejor que Prophet. ARIMA alcanzó un MAPE ≈ 31.7%, lo cual, aunque no ideal, se considera razonablemente confiable. Por el contrario, Prophet obtuvo un MAPE cercano al 94%, lo que lo clasifica como un modelo altamente impreciso para este conjunto de datos.

Estos resultados reflejan que ARIMA logró capturar la tendencia general y comportamientos autorregresivos importantes de la serie, mientras que Prophet, con su configuración por defecto, fracasó en representar adecuadamente la variabilidad. De acuerdo con lo reportado por Krishna en [10] los modelos bien ajustados deberían presentar MAPEs inferiores al 20%, lo cual se aproxima ARIMA pero no Prophet.

En contraste, el enfoque basado en redes neuronales recurrentes, al entrenarse conjuntamente en todas la categorías, permite aprovechar patrones comunes y modelar relaciones no lineales, lo que se traduce en una mayor precisión en la predicción.

Sin embargo, se ha identificado oportunidades de mejora en el modelo de redes neuronales. La incorporación de variables externas (como promociones, feriados o tendencias de búsqueda) y la exploración de arquitecturas más profundas (por ejemplo, modelos secuencia-a-secuencia o con mecanismos de atención) podrían aumentar la capacidad predictiva y mejorar la robustez frente a cambios en la demanda.

Podemos mencionar que los hallazgos indican que la red neuronal presenta un desempeño sobresaliente en la predicción de ventas diarias, superando a los modelo ARIMA y Prophet en términos de precisión y adaptación a la alta variabilidad de la serie. Aunque los métodos estadísticos permiten una interpretación clara de los componentes de la serie temporal, su capacidad predictiva es limitada en escenarios con fluctuaciones abruptas. Estos hallazgos abren la posibilidad de futuras investigaciones que integren enfoques híbridos o incorporen información adicional para optimizar la toma de decisiones en el comercio electrónico. Los modelo estadísticos y la red neuronal presentada fueron publicados en el siguiente repositorio de GitHub para el estudio público: https://github.com/CesarLara08/COMIA_PAPER_155_2025.

6 Conclusiones

En este estudio se desarrolló un modelo predictivo de ventas basado en una red neuronal recurrente con arquitectura LSTM y se comparó con métodos estadísticos tradicionales (ARIMA y Prophet) usando el conjunto de datos Olist. La red neuronal mostró alta precisión en la predicción de ventas diarias, con errores mínimos en datos no vistos, lo que evidencia su capacidad para capturar la compleja dinámica y estacionalidad de las distintas categorías de productos. En contraste, aunque los modelos ARIMA y Prophet permitieron descomponer la seria en componentes de tendencia y estacionalidad, sus errores fueron significativamente mayores, subrayando las limitaciones de los enfoques lineales y aditivos ante la volatilidad y los picos irregulares del comercio electrónico.

La efectividad de la red neuronal sugiere que un enfoque basado en aprendizaje profundo puede apoyar la toma de decisiones empresariales en el sector minorista,

facilitando la planificación de inventarios, la gestión de la cadena de suministro y el diseño de estrategias de marketing proactivas. Además, entrenar un modelo global que abarque todas las categorías simplifica el mantenimiento y la escalabilidad, en comparación con la necesidad de ajustar modelos individuales con métodos estadísticos.

Como líneas de trabajo futuro, se propone incorporar variables exógenas para enriquecer el modelo y explotar arquitecturas de redes más avanzadas para capturar dependencias a largo plazo. También se sugiere comparar enfoques híbridos que combinen las fortalezas de los métodos estadísticos y el aprendizaje profundo, permitiendo un análisis más robusto y adaptable a las condiciones del mercado.

En conclusión, el estudio demuestra la viabilidad y efectividad del enfoque basado en LSTM para la predicción de ventas en comercio electrónico, superando las restricciones de los métodos estadísticos tradicionales y abriendo nuevas perspectivas para optimizar estrategias comerciales en la era del Big data.

Referencias

1. Olist, A. S.: Brazilian E-Commerce Public Dataset by Olist. https://www.kaggle.com/dsv/195341 (2024)
2. Ayala-Aldana, N., Monleon-Getino, A., Canela-Soler, J., Retamal-Contreras, E., Predicción con modelo ARIMA en series temporales de Salmonella spp en Chile entre 2014-2022. Ciencia Latina Revista Científica Multidisciplinar, 7(1), pp. 1337–1351 (2023) doi: 10.37811/cl_rcm.v7i1.4484.
3. Deng, M.: The Analysis of Hidden Units in LSTM Model for Accurate Stock Price Prediction. INSTICC, pp. 419–424 (2024) doi: 10.5220/0012799100003885.
4. Bajoudah, A., Alsaidi, M., Alhindi, A.: Time Series Forecasting Model for E-commerce Store Sales Using FB-Prophet. In: 14th International Conference on Information and Communication Systems (ICICS), pp. 1–6 (2023) doi: 10.1109/ICICS60529.2023.10330530.
5. Weytjens, H., Lohmann, E., Kleinsteuber, M.: Cash flow prediction: MLP and LSTM compared to ARIMA and Prophet. Electronic Commerce Research, 21(2), pp. 371–391 (2021). doi: 10.1007/s10660-019-09362-7.
6. Yu, Y., Si, X., Hu, C., Zhang, J.: A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. Neural Comput, 31(7), pp. 1235–1270 (2019) doi: 10.1162/neco_a_01199.
7. Zhang, X., Guo, F., Chen, T., Pan, L., Beliakov, G., Wu, J.: A Brief Survey of Machine Learning and Deep Learning Techniques for E-Commerce Research. Multidisciplinary Digital Publishing Institute (MDPI) (2023) doi: 10.3390/jtaer18040110.
8. Hendra Saputra, B.: Evaluation of ARIMA model performance in projecting future sales: Case study on electronic products. Jurnal Mantik, 16(5) (2024. doi: 10.35335/cit.Vol16.2024.993.pp329-337.
9. Khatibi, A., Da Silva, A.P.C., Almeida, J.M., Gonçalves, M.A.: A quantitative analysis of the impact of explicit incorporation of recency, seasonality and model specialization into fine-grained tourism demand prediction models. PLoS One, 17(12) (2022) doi: 10.1371/journal.pone.0278112.

10. Krishna, K., Samal, R., Sathya, K., Santosh, B., Das, K., Acharaya, A.: Time Series based Air Pollution Forecasting using SARIMA and Prophet Model (2019) doi: 10.1145/3355402.3355417.