.UBAfiuba
FACULTAD DE INGENIERÍA

Pronóstico de ventas para toma de decisiones en comercio electrónico usando machine learning

Ing. Jonathan Matias Borda

Carrera de Especialización en Inteligencia Artificial

Director: Dr. Diego Onna (Universidad de Buenos Aires)

Jurados:

Jurado 1 (pertenencia)
Jurado 2 (pertenencia)
Jurado 3 (pertenencia)

Ciudad de Buenos Aires, junio de 2026

Resumen

En la presente memoria se describe el desarrollo de un modelo de inteligencia artificial para el pronóstico de ventas en la tienda en línea de la empresa Latech [1], dedicada a la producción y comercialización de barritas alimenticias. El trabajo tiene como finalidad mejorar la planificación de la producción y facilitar la gestión del inventario y toma de decisiones estratégicas en la empresa.

Para su realización se aplicaron conocimientos de análisis de datos, modelado de series temporales, aprendizaje automático, integración de APIs y desarrollo de software.

Índice general

Resumen I

1. Introducción general 1
1.1. Contexto y motivación . . . 1
1.2. Estado del arte . . . 2
1.3. Objetivo y alcance . . . 3
1.3.1. Objetivo del trabajo . . . 4
1.3.2. Alcance del trabajo . . . 4

2. Introducción específica 7
2.1. Requerimientos . . . 7
2.2. Modelos de inteligencia artificial utilizados . . . 8
2.3. Tratamiento de datos (herramientas) . . . 9
2.3.1. Bibliotecas para procesamiento y análisis de datos . . . 9
2.3.2. Bibliotecas para visualización . . . 9
2.3.3. Plataformas y APIs externas . . . 9
2.3.4. Exposición de servicios . . . 10
2.3.5. Contenerización . . . 10
2.3.6. Orquestación y pipelines . . . 10

3. Diseño e implementación 13
3.1. Arquitectura del sistema . . . 13
3.1.1. Capa de recolección de datos . . . 14
3.1.2. Capa de procesamiento y transformación . . . 15
3.1.3. Capa de almacenamiento estructurado . . . 16
3.1.4. Capa de modelado predictivo . . . 17
3.1.5. Capa de integración operativa con microservicios . . . 18
3.1.6. Capa de visualización y consumo dentro de Inventory Tracker 18
3.1.7. Justificación global de la arquitectura . . . 19
3.2. Automatización mediante tareas programadas . . . 19
3.3. Condiciones del trabajo . . . 20
3.4. Análisis de datos . . . 21
3.5. Desarrollo de modelos . . . 21
3.6. Desarrollo de un framework modular . . . 21
3.7. Despliegue con microservicios e integración con bases de datos . . . 21

4. Ensayos y resultados 23
4.1. Ensayos de modelos . . . 23
4.2. Desempeño del modelo . . . 23
4.3. Desempeño de la predicción . . . 23
4.4. Comparación de algoritmos . . . 23
4.5. Validación de cumplimiento de requerimientos . . . 23
Requerimiento funcional 1 . . . 23

Requerimiento funcional 2 24
Requerimiento funcional 3 24
Requerimiento funcional 4 25
Requerimiento funcional 5 25
Requerimiento funcional 6 26
Requerimiento funcional 7 26

5. Conclusiones **29**
5.1. Resultados obtenidos 29
5.2. Trabajo futuro 29

Bibliografía **31**

Índice de figuras

3.1. Diagrama Entidad-Relación. Muestra las dos tablas principales. . . 17
3.2. Flujo general del sistema para generar el pronóstico de ventas. . . . 21

Índice de tablas

Capítulo 1

Introducción general

En este capítulo se presenta el contexto general del trabajo y la problemática que motivó su desarrollo. Se exponen las razones y necesidades que impulsaron su realización, junto con una descripción de las soluciones existentes y los enfoques actuales relacionados con la temática abordada. Asimismo, se detallan los propósitos principales del trabajo y se delimitan los alcances y límites de su implementación.

1.1. Contexto y motivación

El presente trabajo forma parte de la Carrera de Especialización en Inteligencia Artificial y tiene como propósito el desarrollo de un modelo capaz de pronosticar las ventas de una tienda en línea perteneciente a la empresa Latech. Esta compañía fabrica y comercializa barritas alimenticias en distintos sabores, lo que exige una planificación de producción precisa para evitar tanto faltantes como excedentes de stock. En este contexto, disponer de pronósticos confiables de ventas representa un factor estratégico para las decisiones operativas y comerciales.

La empresa ofrece siete sabores considerados productos base, disponibles de forma permanente, y además introduce ediciones limitadas que agregan entre dos y cuatro variantes nuevas por año. Estas ediciones dependen del nivel de aceptación del público y permanecen activas solo mientras alcanzan un volumen de ventas satisfactorio. El producto se vende en paquetes de 10, 20 y 30 unidades, con un precio por unidad que disminuye a medida que aumenta el tamaño del paquete. Asimismo, el sitio ofrece al usuario la opción de suscribirse, lo que permite recibir el producto mensualmente con un descuento aproximado del 10 %. Este sistema facilita la construcción de patrones de demanda asociados a consumos recurrentes y posibilita el análisis del ciclo de vida de las suscripciones, incluyendo altas, bajas y reactivaciones.

Latech lanza cada nueva variante de sabor mediante campañas simultáneas en sus canales de comunicación, como TikTok [2], Instagram [3], Facebook [4] y Google [5]. Estos lanzamientos producen un aumento significativo en la actividad comercial y suelen duplicar las ventas de la nueva variante en comparación con los productos restantes. Un comportamiento similar aparece durante eventos promocionales, como *Black Friday*, donde la aplicación de descuentos generales genera un incremento promedio cercano al 20 % en las ventas totales. Estas fluctuaciones introducen picos marcados en la demanda, cuya anticipación resulta crucial para coordinar producción, logística y abastecimiento.

Antes del desarrollo de este sistema, la empresa no contaba con un mecanismo de pronóstico capaz de anticipar estos cambios de manera sistemática. Las variaciones abruptas en la demanda, especialmente las derivadas de campañas exitosas, superaban en ocasiones la capacidad inmediata de producción. En consecuencia, la organización registraba episodios de quiebre de stock, retrasos en las entregas y deterioro en la experiencia del cliente. La falta de previsión reducía la capacidad de planificar compras de insumos, organizar turnos de producción y definir niveles óptimos de inventario.

La incorporación de datos provenientes de múltiples fuentes permite construir una visión integral del proceso comercial. La empresa utiliza la API de Triple Whale [6] para concentrar la información de inversión publicitaria de TikTok, Instagram, Facebook y Google. Esta información permite identificar qué canales influyen en la conversión, la proporción de usuarios nuevos y recurrentes, y las variaciones en el volumen de ventas según la estrategia de difusión. Por su parte, los datos históricos de ventas extraídos desde Shopify [7] permiten reconstruir la evolución temporal de la demanda, medir el impacto de las ediciones limitadas, cuantificar el efecto de las promociones y analizar patrones estacionales. La combinación de ambas fuentes constituye un insumo esencial para entrenar un modelo de predicción de ventas robusto y contextualizado.

En conjunto, este trabajo apunta a desarrollar un sistema basado en aprendizaje automático que proporcione estimaciones confiables de demanda y que funcione como herramienta de apoyo a la planificación comercial. Su objetivo final consiste en mejorar la toma de decisiones en áreas como producción, abastecimiento, definición de inventarios, activación de campañas y análisis del comportamiento del cliente, que favorezca una gestión más eficiente y orientada a datos.

1.2. Estado del arte

En la actualidad, el pronóstico de ventas mediante técnicas de inteligencia artificial constituye un área de creciente interés en el ámbito del comercio electrónico. Los avances en el análisis de datos y en el aprendizaje automático han permitido la aparición de modelos capaces de anticipar la demanda con altos niveles de precisión, lo que contribuye a una mejor planificación de la producción, la gestión de inventarios y la toma de decisiones estratégicas.

Entre los enfoques más utilizados se encuentran los modelos de series temporales clásicos, como ARIMA [8], SARIMA y *Exponential Smoothing*, que permiten capturar patrones estacionales y tendencias a lo largo del tiempo [9]. Sin embargo, estos métodos presentan limitaciones cuando los datos incluyen múltiples factores externos, tales como campañas publicitarias, variaciones en los precios, promociones o comportamiento del usuario.

En los últimos años se ha observado un aumento significativo en la adopción de técnicas de aprendizaje profundo (*Deep Learning*) y aprendizaje automático (*Machine Learning*), que ofrecen una mayor capacidad para modelar relaciones no lineales y para integrar información proveniente de diversas fuentes. Modelos como Redes Neuronales Recurrentes (RNN), LSTM (*Long Short-Term Memory*) y *Random Forest Regressor* se han aplicado con éxito en la predicción de ventas en entornos de comercio electrónico, debido a su habilidad para capturar dependencias temporales y correlaciones entre variables [10, 11, 12, 13].

De forma más reciente, la literatura destaca los modelos basados en transformadores, como *Temporal Fusion Transformer* (TFT) [14], Informer [15] y N-BEATS [16], que alcanzaron resultados competitivos en competencias internacionales de series temporales y en aplicaciones industriales. Estos modelos incorporan mecanismos de atención que permiten identificar qué variables externas influyen en la demanda y en qué momentos lo hacen, lo que resulta especialmente relevante en contextos con campañas publicitarias, promociones o cambios abruptos en la visibilidad de los productos.

El interés por evaluar rigurosamente estas técnicas ha impulsado la creación de competencias a gran escala, como M4 y M5 [17], organizadas por la comunidad académica y la industria. Estas competencias demostraron que la combinación de enfoques tradicionales con métodos basados en aprendizaje profundo permite alcanzar mejoras significativas en la precisión de los pronósticos, especialmente cuando los datos provienen de entornos comerciales reales.

Asimismo, el uso de plataformas integradas de datos, tales como Shopify [7] y Triple Whale [6], permite el desarrollo de soluciones personalizadas que combinan información transaccional, métricas de marketing y comportamiento del usuario. La integración de estas fuentes facilita la incorporación de variables exógenas, como inversión publicitaria, características de los productos, descuentos, actividad de suscripciones o lanzamientos de nuevas variantes. Estas variables contribuyen a mejorar la capacidad predictiva de los modelos.

Entre las soluciones comerciales más reconocidas se destacan Amazon Forecast [18], un servicio basado en redes neuronales desarrollado por Amazon Web Services [19] que automatiza la creación de modelos de predicción de demanda, y Prophet [20], una herramienta de código abierto creada por Meta [21] que utiliza un enfoque aditivo para modelar tendencias y estacionalidades de manera flexible. Ambas herramientas representan referencias relevantes en el campo del pronóstico de series temporales con datos de negocios.

La tendencia actual apunta hacia el desarrollo de modelos híbridos que combinan la capacidad de las redes neuronales para capturar relaciones complejas con la interpretabilidad y la robustez de los métodos tradicionales. Esta combinación permite enfrentar entornos caracterizados por factores externos altamente variables, como campañas digitales, promociones estacionales, descuentos temporales o lanzamientos de productos.

En este marco, el trabajo desarrollado se apoya en los enfoques actuales de predicción de demanda mediante aprendizaje automático, datos históricos de ventas, inversión publicitaria y comportamiento del usuario. El objetivo consiste en construir un modelo que represente de forma adecuada las tendencias reales del negocio y que funcione como herramienta de apoyo para la toma de decisiones estratégicas en la empresa Latech.

1.3. Objetivo y alcance

En esta sección se mencionan los propósitos principales del trabajo y los límites de su implementación.

1.3.1. Objetivo del trabajo

El propósito de este trabajo es desarrollar un modelo de inteligencia artificial que permita predecir con precisión las ventas del producto alimenticio de la empresa Latech, a partir de datos históricos de ventas y de inversión publicitaria.

Los objetivos específicos incluyen:

- Construir un sistema predictivo que anticipe la demanda para horizontes de pronóstico corto (1 a 6 meses), período crítico para la planificación operativa de la empresa.
- Identificar y cuantificar el impacto de las campañas publicitarias en las ventas mediante mecanismos de atención interpretable.
- Proporcionar a la empresa herramientas para optimizar la planificación de inventario y producción.
- Reducir los riesgos de faltantes y excesos de stock mediante predicciones confiables.
- Establecer una base técnica para la toma de decisiones estratégicas basada en datos.

1.3.2. Alcance del trabajo

A continuación, se enuncian los items que incluye el trabajo:

- Relevamiento y análisis de los datos históricos de ventas de la tienda en línea obtenidos mediante la API de Shopify.
- Relevamiento y análisis de los datos históricos de inversión publicitaria provenientes de la API de Triple Whale.
- Limpieza, transformación y consolidación de datos provenientes de ambas plataformas.
- Análisis exploratorio de datos (EDA) para identificar patrones, tendencias y relaciones entre las variables.
- Desarrollo y entrenamiento de modelos de inteligencia artificial orientados a la predicción de ventas.
- Evaluación comparativa de distintos modelos para seleccionar el que ofrezca la mejor precisión y capacidad predictiva.
- Implementación de un módulo funcional integrado al sistema Inventory Tracker de la empresa, que permita realizar predicciones de ventas para períodos futuros.
- Documentación técnica del proceso, del modelo elegido y de su uso.
- Entrega de reportes que expliquen los hallazgos y recomendaciones derivadas del análisis.

Por otra parte, quedan excluidas del alcance las siguientes actividades:

- Implementaciones en tiempo real del modelo, es decir, sistemas que actualicen las predicciones de forma instantánea ante cada nuevo dato recibido. Sí

se contempla la posibilidad de programar ejecuciones periódicas del modelo (por ejemplo, una vez al día) para actualizar las predicciones de manera regular.

- Garantía de precisión absoluta en las predicciones, ya que la precisión depende de la calidad y estabilidad de los datos futuros y de factores externos no controlables.
- Acciones o recomendaciones específicas sobre estrategias de marketing más allá de lo inferido de los análisis de datos.
- Re-ingeniería de procesos operativos internos de producción o logística.
- Optimización de procesos internos de producción o logística, salvo en lo que respecta a la estimación de demanda.

Capítulo 2

Introducción específica

En este capítulo se presentan los requerimientos más importantes, los modelos de inteligencia artificial utilizados y las herramientas usadas para el tratamiento de datos.

2.1. Requerimientos

En esta sección se presentan algunos requerimientos establecidos para el trabajo. En cada requerimiento se detalla su objetivo y alcance, con el fin de precisar las condiciones necesarias para el desarrollo y la correcta implementación del modelo propuesto.

- Importación de datos de ventas desde Shopify: el sistema debe permitir la obtención de datos históricos y actualizados de ventas desde la plataforma Shopify, con el objetivo de utilizarlos en el proceso de entrenamiento del modelo de predicción de demanda.
- Importación de datos de inversión publicitaria desde Triple Whale: el sistema debe permitir la incorporación de datos de inversión publicitaria provenientes de la plataforma Triple Whale, con el propósito de integrar esta variable en el análisis y modelado de ventas.
- Consolidación de datos provenientes de Shopify y Triple Whale: el sistema debe unificar la información de ventas y de inversión publicitaria en un único conjunto de datos, con el fin de preparar una base consistente para el entrenamiento del modelo de predicción.
- Desarrollo y entrenamiento del modelo predictivo: el sistema debe implementar y entrenar un modelo de inteligencia artificial capaz de generar predicciones de ventas precisas sobre los datos históricos consolidados.
- Visualización de predicciones de ventas en Inventory Tracker: el sistema debe presentar las predicciones de ventas generadas por el modelo dentro de la plataforma Inventory Tracker, integradas con los datos históricos para facilitar el análisis y la planificación de producción.
- Descarga de predicciones de ventas en formato CSV: el sistema debe permitir la exportación de las predicciones de ventas a archivos CSV para su análisis externo o integración con otras herramientas de planificación.

2.2. Modelos de inteligencia artificial utilizados

Para abordar el trabajo se evaluaron y aplicaron diversos modelos de inteligencia artificial, dado que el comportamiento de la demanda no responde a un único patrón fijo ni puede ser capturado adecuadamente por un solo tipo de modelo. También, se consideró seleccionar aquel que ofreciera el mejor equilibrio entre precisión, interpretabilidad y capacidad de generalización.

A continuación, se describen los tipos de modelos aplicados:

- Random Forest Regressor [22]: se utilizó por su robustez frente a sobreajuste y su capacidad para capturar relaciones no lineales entre variables. Este modelo resulta especialmente útil cuando se integran múltiples fuentes de datos, como ventas históricas e inversión publicitaria.
- Gradient Boosting Machines [23] (XGBoost, LightGBM): se evaluaron por su alto rendimiento en competencias de ciencia de datos y su eficiencia computacional. Estos modelos permiten optimizar la función de pérdida de manera iterativa, mejorando la precisión en la predicción de series temporales con características externas.
- SARIMA (*Seasonal Autoregressive Integrated Moving Average*): se aplicó para modelar la estacionalidad y tendencia presentes en los datos de ventas. SARIMA es especialmente efectivo cuando el comportamiento temporal es predominante y estable a lo largo del tiempo.
- *Exponential Smoothing* (ETS) [24]: se consideró como alternativa para capturar patrones estacionales y de tendencia con un enfoque más intuitivo y menos paramétrico que SARIMA.
- *Long Short-Term Memory* (LSTM): se implementaron redes LSTM debido a su habilidad para retener información a largo plazo y modelar dependencias temporales complejas. Este tipo de red es adecuado para series temporales con patrones no lineales y múltiples variables exógenas, como la inversión publicitaria por plataforma.
- GRU (*Gated Recurrent Unit*) [25]: se evaluó como una variante más simple y computacionalmente eficiente de las LSTM, útil cuando los recursos de entrenamiento son limitados. Además, mantienen la capacidad esencial de capturar dependencias temporales a largo plazo y manejar el problema del gradiente *vanishing*, que en ocasiones demuestra un desempeño comparable al de las LSTM.
- *Temporal Fusion Transformer* (TFT): Se implementó este modelo de vanguardia por su capacidad de atención interpretable, que permite identificar qué variables exógenas y qué períodos históricos influyen en cada predicción. El TFT resulta particularmente adecuado para el contexto de Latech, donde es crucial entender el impacto de las inversiones publicitarias en diferentes horizontes temporales.
- Enfoque híbrido: se exploró la combinación de modelos clásicos (como SARIMA) con redes neuronales (como LSTM o TFT), con el fin de aprovechar la capacidad de los primeros para modelar componentes estacionales y de tendencia, y la flexibilidad de las segundas para capturar relaciones no lineales y efectos de variables externas.

- Prophet: se incluyó el modelo Prophet, desarrollado por Meta [21], por su facilidad de uso y capacidad para manejar estacionalidades múltiples, días festivos y cambios en la tendencia. Este modelo resultó de interés para validar la estructura temporal de los datos antes de aplicar modelos más complejos.

Cada uno de estos modelos fue entrenado y validado utilizando el conjunto de datos consolidado de ventas e inversión publicitaria, y se compararon mediante métricas como MAE (error absoluto medio), RMSE (raíz del error cuadrático medio) y MAPE (error porcentual absoluto medio), con el fin de seleccionar el que mejor se adaptara a las necesidades de pronóstico de Latech.

2.3. Tratamiento de datos (herramientas)

En este a sección se describen los procesos y herramientas utilizadas para la limpieza, transformación y análisis de los datos, así como los recursos externos desarrollados por terceros que resultaron fundamentales para la obtención y procesamiento de la información.

2.3.1. Bibliotecas para procesamiento y análisis de datos

- Pandas [26]: permite manipular y transformar datos tabulares, incluidas operaciones de filtrado, agrupamiento y combinación de datasets.
- NumPy [27]: ofrece operaciones numéricas y manejo eficiente de arreglos multidimensionales.
- Scikit-learn [28]: empleada para el preprocesamiento de datos (escalado, codificación de variables categóricas) y la implementación de modelos clásicos de *machine learning*.
- Pytorch [29]: framework que permite definir arquitecturas personalizadas, calcular gradientes automáticamente y realizar entrenamientos acelerados mediante GPU. Su integración con bibliotecas como NumPy, Pandas y Matplotlib facilita incorporarlo a flujos de trabajo existentes y explorar modelos predictivos más sofisticados que los basados únicamente en series temporales clásicas.

2.3.2. Bibliotecas para visualización

- Matplotlib [30] y Seaborn [31]: permiten generar gráficos estáticos que facilitan la identificación de patrones, tendencias y valores atípicos.
- Plotly [32]: utilizado para la creación de visualizaciones interactivas dentro de los notebooks.

2.3.3. Plataformas y APIs externas

- Shopify API [7]: proporciona acceso automatizado a datos históricos de ventas, productos y transacciones.
- Triple Whale API [6]: ofrece datos de inversión publicitaria desglosados por plataforma y campaña.

- PostgreSQL [33]: funciona como sistema de gestión de bases de datos destinado al almacenamiento estructurado de los datos consolidados.
- Cron [34]: permite ejecutar tareas programadas a horas, fechas o intervalos fijos periódicos para automatización de *pipelines*.
- AWS S3 [35]: ofrece un servicio de almacenamiento de objetos en la nube de AWS que con alta escalabilidad, disponibilidad, durabilidad y seguridad necesarios para guardar los modelos y datos intermedios.
- Git [36]: sistema de control de versiones distribuido que mantiene un registro histórico de cambios en archivos de código, configuraciones y documentación, esencial para el desarrollo colaborativo y la reproducibilidad de experimentos.

2.3.4. Exposición de servicios

Para exponer endpoints que permiten consumir resultados, ejecutar procesos o servir modelos se utiliza FastAPI. Este framework destaca por su alto rendimiento y por su arquitectura asíncrona basada en ASGI [37].

Las características más relevantes para este proyecto son:

- Alto rendimiento: maneja múltiples solicitudes de manera eficiente gracias a su soporte nativo para operaciones asíncronas.
- Validación y documentación automática: incorpora Pydantic [38] para validar estructuras de entrada y genera documentación OpenAPI/Swagger [39] sin requerir configuraciones adicionales.
- Flexibilidad: permite definir endpoints destinados al consumo de modelos, consultas filtradas, refresco de datos o ejecución de procesos administrativos.

2.3.5. Contenerización

- Docker [40]: permite definir entornos de ejecución consistentes para los servicios del proyecto. Sus contenedores aseguran reproducibilidad, aislamiento y portabilidad entre distintos entornos.
- Docker compose [41]: herramienta que simplifica la orquestación de múltiples contenedores Docker mediante archivos de configuración en formato YAML. Permite la definición y ejecución de aplicaciones multi-servicio de manera coordinada, lo que facilita la puesta en marcha de entornos complejos que incluyen la API de FastAPI, la base de datos PostgreSQL y servicios auxiliares de forma integrada. Docker Compose maneja automáticamente las dependencias entre servicios, las redes virtuales y los volúmenes de persistencia.

2.3.6. Orquestación y pipelines

- Metaflow [42]: se utiliza como herramienta principal para estructurar y ejecutar pipelines de datos y experimentos, con registro automático de artefactos y trazabilidad.

- Apache Airflow [43]: framework de orquestación que define flujos de trabajo como DAGs (*Directed Acyclic Graphs*) [44] y permite programar ejecuciones periódicas, manejar dependencias entre tareas y monitorizar el estado de los procesos de datos.

Capítulo 3

Diseño e implementación

En este capítulo se describen los criterios utilizados para la construcción del desarrollo y la arquitectura definida para la solución. Se presenta la estructura general del sistema, las decisiones de diseño adoptadas y los componentes que permiten integrar fuentes de datos externas, procesarlas y generar pronósticos confiables de ventas.

3.1. Arquitectura del sistema

La arquitectura diseñada para el sistema de predicción de ventas se basa en un flujo de procesamiento secuencial y automatizado que conecta las fuentes de datos externas con el entorno interno del cliente. El diseño se orientó por tres criterios principales:

- Centralización y coherencia de datos. Se priorizó la unificación de múltiples fuentes (ventas, comportamiento de usuarios, inversión en publicidad).
- Escalabilidad operativa que permita que el sistema integre nuevos canales, nuevas métricas o nuevos modelos sin alterar la estructura existente.
- Procesos reproducibles y auditables para asegurar que cada etapa sea trazable y que el sistema pueda ser mantenido o ampliado por equipos técnicos futuros.

El sistema se organiza en una arquitectura por capas que separa claramente las responsabilidades, desde la obtención de los datos hasta la entrega del pronóstico al usuario final. Estas capas son:

1. Capa de recolección de datos (Data Ingestion Layer).
2. Capa de procesamiento y transformación (ETL).
3. Capa de almacenamiento estructurado (Data Storage Layer).
4. Capa de modelado predictivo (Modeling Layer).
5. Capa de integración operativa mediante microservicios.
6. Capa de visualización y consumo dentro de Inventory Tracker.

Cada una de estas capas se comunica de manera acotada mediante APIs internas o a través de la base de datos para garantizar un bajo acoplamiento.

3.1.1. Capa de recolección de datos

Esta capa se encarga de conectarse periódicamente a APIs externas para obtener la información necesaria para entrenar y actualizar los modelos. Las dos fuentes externas son:

- Shopify: funciona como el pilar central de la información comercial del sistema. A través de su API es posible acceder a los históricos de ventas, que incluye detalles como estado, valor total y descuentos. Además, cada orden incluye etiquetas que permiten distinguir si la compra corresponde a un cliente nuevo o recurrente, lo que posibilita analizar el comportamiento de los usuarios a lo largo del tiempo. Esta distinción es un insumo clave para caracterizar patrones de fidelidad, frecuencia de compra y recurrencia. A partir de la combinación de órdenes, etiquetas de comportamiento y series temporales de ventas, se logra reconstruir el ciclo completo de los clientes y diferenciar entre la demanda estable generada por suscriptores activos y la demanda variable asociada a compras espontáneas o impulsadas por campañas de marketing.
- Triple Whale: una plataforma que centraliza la información de inversión publicitaria proveniente de TikTok Ads [2], Instagram Ads [3], Facebook Ads [4] y Google Ads [5]. En este caso, su API no provee datos desagregados de campañas individuales, sino únicamente el monto diario total invertido en publicidad considerando todas las campañas activas. A pesar de esta limitación, esta información resulta suficiente para analizar la relación entre los niveles diarios de inversión publicitaria y las variaciones observadas en las ventas. De esta forma, el gasto diario consolidado funciona como un indicador de la presión publicitaria ejercida en cada jornada, lo que permite estudiar su impacto en el comportamiento de compra y en la demanda general del sistema.

Ambas integraciones están preparadas para ejecutarse en un proceso automatizado que consulta las APIs diariamente y almacena la información en una base de datos PostgreSQL [33] interna. Este proceso opera bajo un módulo ETL [45] donde se llevan a cabo tareas de validación, normalización y estandarización para garantizar la coherencia entre fuentes heterogéneas.

La decisión de almacenar localmente los datos en PostgreSQL responde a tres objetivos principales:

- Autonomía del sistema: aunque se dispone de acceso a las APIs de Shopify y Triple Whale, estas pueden presentar interrupciones temporales, límites de tasa de consulta (*rate limits*) o cambios no anunciados en su estructura. Al mantener una copia local de los datos, el sistema de pronóstico continúa funcionando incluso durante caídas de las APIs externas, garantizando disponibilidad continua.
- Rendimiento y velocidad: consultar datos históricos desde una base de datos local es significativamente más rápido que realizar múltiples solicitudes HTTP a APIs externas, especialmente cuando se procesan grandes volúmenes de datos durante el entrenamiento de modelos o la generación de informes.
- Desacoplamiento y control de datos: al poseer una copia estructurada de los datos, se evita la dependencia directa de la disponibilidad y formato de las

APIs externas. Esto permite realizar transformaciones, enriquecimientos y agregaciones previas que optimizan los procesos posteriores de modelado y análisis.

De esta manera, PostgreSQL actúa como una capa de abstracción y resiliencia, que permite que el sistema opere de manera eficiente, predecible y con menor latencia, sin comprometer su funcionalidad ante eventuales fallos externos.

3.1.2. Capa de procesamiento y transformación

Una vez obtenidos los datos desde las APIs externas, se realiza un proceso de transformación que normaliza y estructura la información. Las principales tareas incluyen:

- Limpieza de campos inconsistentes o incompletos.
- Conversión de formatos de fechas, monedas y valores numéricos.
- Enriquecimiento de datos combinando ventas, comportamiento de usuarios y gasto publicitario.
- Integración de información transaccional con métricas de suscripción.
- Control de duplicados y estandarización de claves primarias.
- Creación de agregaciones temporales diarias.
- Creación de variables derivadas de los datos como medias móviles o ratios de inversión.
- Identificación de picos de campañas.
- Señalización de eventos especiales tales como: promociones, feriados, lanzamientos.
- Integración del estado de suscripciones activas e inactivas.

Este proceso de transformación se implementó mediante DAGs (*Directed Acyclic Graphs*) en Apache Airflow, lo que permite orquestar de manera programada, reproducible y monitorizable cada etapa del pipeline de datos.

Los datos intermedios y finales de este proceso se almacenan temporalmente en disco, en un formato estructurado y accesible llamado Parquet, para permitir su reutilización en ejecuciones posteriores sin necesidad de reprocesar desde cero. Este enfoque no solo optimiza el uso de recursos, sino que también facilita la depuración y auditoría de los datos generados en cada etapa.

Dado que Shopify y Triple Whale estructuran sus datos de manera diferente, se construyó un esquema interno unificado que respeta la granularidad diaria necesaria para los modelos de predicción. Este esquema queda materializado en una tabla consolidada dentro de PostgreSQL, lista para su consumo por el módulo de modelado.

La implementación mediante Airflow permite además:

- Ejecución programada o bajo demanda de los flujos de transformación.
- Reejecución selectiva de tareas fallidas sin afectar etapas exitosas.
- Monitoreo visual del estado del pipeline y alertas ante incidencias.

- Versionado y mantenimiento independiente de la lógica de transformación.

Este módulo permite encapsular toda la lógica de preparación del dataset, manteniéndolo desacoplado del motor de predicción para que se pueda actualizar fácilmente sin impactar en la generación de pronósticos.

3.1.3. Capa de almacenamiento estructurado

Luego de su procesamiento, los datos se almacenan en una base de datos PostgreSQL diseñada específicamente para consultas analíticas y para servir como fuente unificada de información durante el entrenamiento y evaluación de los modelos. Dentro de esta capa se construye la tabla principal denominada *orders*, en la que se almacena la información proveniente de Shopify ya depurada y enriquecida. Esta tabla contiene las columnas fundamentales para caracterizar el comportamiento de compra de los usuarios:

- *created*: registra la fecha de creación de cada orden.
- *totalPrice*: correspondiente al monto total pagado.
- *customerId*: identifica al cliente.
- *lineItems*: donde se detalla cada producto incluido en la compra junto con sus cantidades.
- *channel*: indica el canal por el cual ingresó la orden.
- *tags*: permite distinguir si la compra corresponde a un cliente nuevo o recurrente mediante etiquetas provistas por Shopify.
- *orderNumberForCustomer*: señala el número de orden que representa dentro del historial del cliente (por ejemplo, un valor igual a 1 implica un cliente nuevo).
- *diffWeeksFromFirstPurchase*: expresa la cantidad de semanas transcurridas desde la primera compra del usuario, lo que permite analizar patrones de frecuencia y retención.

A partir de esta tabla estructurada, se generan columnas derivadas mediante procesos de enriquecimiento en Python, con el objetivo de facilitar el análisis y preparar los datos para su uso en modelos predictivos. Entre estas variables se incluyen:

- *created_weekday*: identifica el día de la semana de cada orden.
- *created_month*: permite estudiar estacionalidades mensuales.
- *unique_customers*: calcula la cantidad de clientes distintos por día.
- *new_customers*: contabiliza cuántos usuarios realizaron su primera compra dentro de cada fecha.
- *returning_customers*: contabiliza cuántos usuarios realizaron más de una compra.

Estas nuevas columnas permiten construir vistas agregadas y analizar la evolución del comportamiento de los clientes en el tiempo.

Finalmente, otra tabla denominada *forecast* consolida la información diaria extraida de TripleWhale. Esta tabla incorpora, entre otras variables, el monto de inversión publicitaria bajo la columna *ad_spend*. La tabla contiene además otros campos utilizados por el sistema Inventory Tracker, derivados de los datos ya existentes. Sin embargo, dichas columnas no forman parte del alcance del presente trabajo, ya que responden a necesidades operativas específicas del cliente y no intervienen en el proceso de modelado.

Este diseño facilita tanto el acceso para entrenamiento de modelos como la consulta rápida desde Inventory Tracker. En la figura 3.1 se muestran las dos tablas principales del trabajo con las columnas que se tomaron en cuenta para este proceso de almacenamiento.

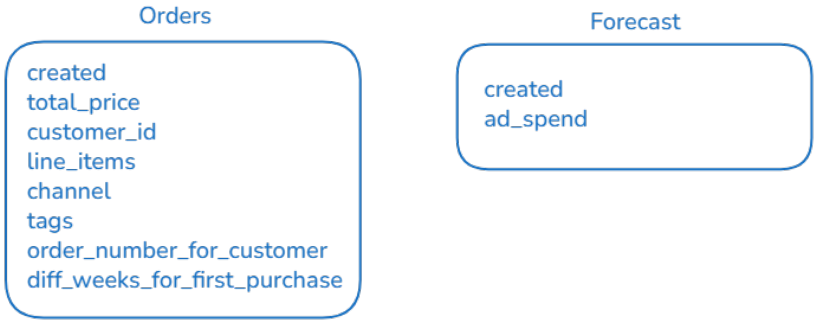

FIGURA 3.1. Diagrama Entidad-Relación. Muestra las dos tablas principales.

3.1.4. Capa de modelado predictivo

Esta capa es responsable del entrenamiento, evaluación y selección de los modelos de pronóstico de ventas. Se implementa un enfoque comparativo donde se ejecutan simultáneamente los modelos mencionados en el capítulo 2.2 (SARIMA, Exponential Smoothing, Random Forest, XGBoost, LightGBM, LSTM y Prophet), con el objetivo de identificar el que ofrezca el mejor desempeño predictivo según las métricas definidas.

El proceso de modelado sigue las siguientes etapas:

1. Entrenamiento múltiple: todos los modelos son entrenados utilizando el mismo conjunto de datos históricos consolidado y enriquecido.
2. Evaluación y métricas: cada modelo es evaluado con la métrica RMSE para penalizar errores grandes que impactan significativamente en la planificación financiera y de producción.
3. Selección automática: el modelo que obtiene el menor valor de RMSE en el conjunto de validación es seleccionado automáticamente para su uso en producción.
4. Persistencia del modelo: una vez seleccionado, el modelo ganador es serializado y almacenado en un registro de modelos donde queda disponible para su uso en las predicciones futuras.
5. Exposición mediante API: el modelo guardado es integrado en un endpoint REST que forma parte del sistema de microservicios para que el sistema Inventory Tracker consulte las predicciones de ventas en tiempo real.

Este diseño no solo asegura que siempre se esté utilizando el modelo más preciso disponible, sino que también facilita la actualización periódica del mismo mediante reentrenamientos programados. Con cada actualización se mantiene la capacidad predictiva del sistema ante cambios en el comportamiento de las ventas o en las dinámicas de mercado.

3.1.5. Capa de integración operativa con microservicios

La arquitectura del sistema se basa en un enfoque de microservicios, donde cada funcionalidad clave se encapsula en un servicio independiente, desplegado y mantenido de forma aislada. Los microservicios implementados son los siguientes:

- Servicio de ingestión (cronjobs + conectores API).
- Servicio de ETL.
- Servicio de predicción.
- Servicio de API interna para Inventory Tracker.

La comunicación entre estos servicios se realiza mediante endpoints REST internos para operaciones síncronas y, cuando se requiere procesamiento asincrónico o desacoplamiento temporal, a través de colas de mensajería (por ejemplo, con RabbitMQ [46] o Redis [47]). Este diseño permite que cada servicio evolucione y escale de forma independiente, sin afectar al resto del sistema.

Los principales beneficios de esta capa de microservicios son:

- Actualizaciones incrementales: cada servicio puede ser actualizado, modificado o reemplazado sin impactar en el funcionamiento de los demás, lo que facilita la incorporación de mejoras o nuevas fuentes de datos.
- Escalabilidad selectiva: los módulos con mayor carga computacional, como el servicio de predicción o de ingestión, pueden escalarse horizontalmente de forma independiente, optimizando el uso de recursos.
- Aislamiento de fallos: un fallo en un servicio (por ejemplo, la indisponibilidad temporal de una API externa) no propaga su error al resto del sistema, gracias al desacoplamiento y a los mecanismos de tolerancia a fallos implementados.
- Facilidad de mantenimiento y despliegue: cada servicio cuenta con su propio repositorio, configuración y ciclo de vida, lo que simplifica las tareas de desarrollo, pruebas y despliegue continuo.

3.1.6. Capa de visualización y consumo dentro de Inventory Tracker

Las predicciones de ventas finales se integran en el sistema Inventory Tracker mediante una interfaz intuitiva, la cual permite:

- Solicitar pronósticos para distintos horizontes temporales.
- Visualizar ventas históricas junto con proyecciones.
- Combinar la predicción con datos operativos del cliente como: stock, producción, etc.

Este componente cierra el ciclo de valor al convertir la información generada por el sistema en una herramienta de apoyo para la gestión diaria.

3.1.7. Justificación global de la arquitectura

La arquitectura fue diseñada bajo los siguientes principios:

- Modularidad: permite reemplazar o actualizar partes sin reestructurar toda la solución.
- Escalabilidad: preparada para el aumento del volumen de datos y la posible integración de nuevas plataformas.
- Trazabilidad: indispensable para un sistema basado en machine learning.
- Flexibilidad: permite combinar modelos clásicos, modelos de machine learning y eventualmente servicios administrados en la nube.

3.2. Automatización mediante tareas programadas

Para garantizar el funcionamiento continuo y automático del sistema, se diseñó e implementó un conjunto de tareas programadas mediante cronjobs que se ejecutan de manera secuencial y periódica, sin requerir intervención manual. Este flujo automatizado permite mantener la información del sistema actualizada de forma confiable y oportuna. Las principales tareas programadas incluyen:

- Extracción diaria de datos: se ejecuta una conexión automatizada a las APIs de Shopify y Triple Whale para obtener los datos más recientes de ventas e inversión publicitaria. Este proceso garantiza que el sistema opere con información actualizada y evita retrasos en la disponibilidad de datos.
- Actualización del dataset consolidado: los nuevos datos se procesan, limpian y unifican con el historial existente. Este paso genera un dataset integral y consistente que sirve como base para el entrenamiento del modelo y la generación de pronósticos.
- Reentrenamiento periódico del modelo: con una frecuencia configurada (por ejemplo, semanal o mensual), el sistema ejecuta automáticamente el proceso de reentrenamiento del modelo de pronóstico utilizando el dataset actualizado. Esto permite que el modelo se adapte a cambios en las tendencias de ventas, comportamiento del mercado o nuevos patrones estacionales.
- Generación de nuevas predicciones: después del reentrenamiento, el modelo genera pronósticos actualizados para los períodos futuros definidos. Estas predicciones reemplazan a las versiones anteriores y se almacenan en la base de datos para su consulta.
- Invalidación y regeneración de caché en Inventory Tracker: Para asegurar que los usuarios accedan siempre a la información más reciente, el sistema invalida automáticamente la caché del Inventory Tracker y la regenera con las nuevas predicciones y datos consolidados. Esto elimina posibles desfases entre los datos mostrados y la realidad del sistema.

Adicionalmente, se implementaron mecanismos de monitoreo y registro (*logging*) para cada tarea programada, lo que permite auditar el correcto funcionamiento del flujo automatizado y detectar posibles fallos de manera temprana.

Esta automatización integral asegura que el cliente disponga permanentemente de información actualizada, confiable y lista para la toma de decisiones, mientras se reduce significativamente el riesgo de errores humanos y se optimiza el mantenimiento operativo del sistema.

3.3. Análisis de datos

3.4. Desarrollo de modelos

3.5. Desarrollo de un framework modular

3.6. Despliegue con microservicios e integración con bases de datos

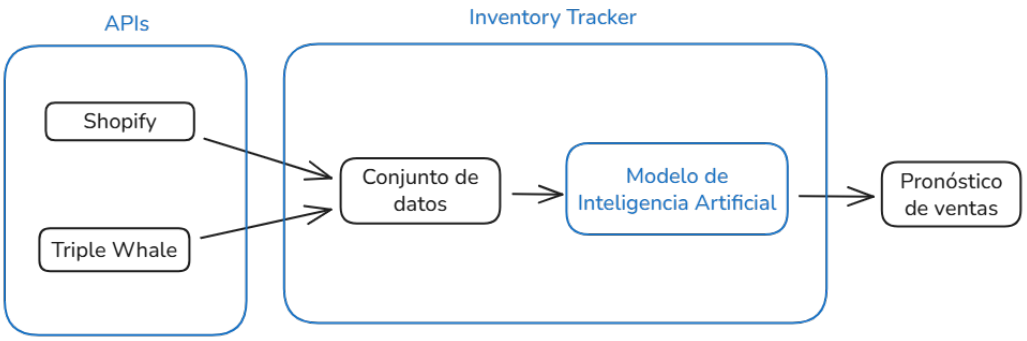

FIGURA 3.2. Flujo general del sistema para generar el pronóstico de ventas.

Capítulo 4

Ensayos y resultados

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

4.1. Ensayos de modelos

La idea de esta sección es explicar cómo se hicieron los ensayos, qué resultados se obtuvieron y analizarlos.

4.2. Desempeño del modelo

4.3. Desempeño de la predicción

4.4. Comparación de algoritmos

4.5. Validación de cumplimiento de requerimientos

A continuación se detalla la descripción técnica y alcance correspondientes a cada uno de los requerimientos especificados anteriormente.

Requerimiento funcional 1

Para cumplir este requerimiento, se desarrolló un módulo de integración que:

- Establece conexión con la API de Shopify mediante credenciales *API Key* y *Access Token*.
- Obtiene y procesa los siguientes atributos mínimos: fecha de compra, identificador de producto, cantidad vendida, precio unitario y canal de venta.
- Permite especificar intervalos de fechas para la descarga de datos.
- Almacena los datos importados en una base de datos PostgreSQL destinada a su posterior uso en el modelado.
- Implementa validaciones y manejo de errores en caso de fallos de autenticación, pérdida de conexión o respuestas vacías.
- Registra la trazabilidad del proceso de importación mediante un archivo de log, incluyendo volumen de datos procesados y errores detectados.

- Verifica duplicidades para evitar el doble procesamiento de registros previamente importados.

Requerimiento funcional 2

Para satisfacer este requerimiento se desarrolló un módulo de integración que:

- Establece conexión con la API de Triple Whale con una clave de acceso provista por la organización.
- Recupera la información de inversión por fecha, campaña y plataforma publicitaria.
- Permite seleccionar y configurar el intervalo temporal de los datos a importar.
- Almacena los datos obtenidos en una base de datos PostgreSQL, manteniendo una estructura consistente con el modelo de análisis utilizado.
- Implementa mecanismos de manejo de errores ante fallos de autenticación, interrupciones en la conexión o respuestas inválidas provenientes de la API.
- Registra el detalle del proceso mediante un archivo de log, incluyendo la cantidad de registros importados y los errores detectados durante la ejecución.
- Permite ejecutar el proceso de importación tanto de forma manual como programada mediante tareas periódicas.
- Incluye validaciones para evitar la duplicación de registros en caso de importaciones sobre el mismo rango de fechas.

Requerimiento funcional 3

Para satisfacer este requerimiento se desarrolló un proceso de consolidación que integra los datos provenientes de Shopify (ventas) y Triple Whale (inversión publicitaria), y garantiza la correcta correspondencia temporal entre ambas fuentes. En particular:

- Se generó un único *dataset* en el que las ventas diarias y la inversión total diaria se vinculan a través del campo de fecha.
- Se realizó el alineamiento temporal de las fechas, y se tuvo en cuenta posibles diferencias de zona horaria entre ambas plataformas.
- En los casos en que alguna de las fuentes no presentaba información para una fecha determinada, se mantuvo el registro correspondiente utilizando valores nulos para evitar la pérdida de información histórica.
- El conjunto de datos final incluye las siguientes columnas: `fecha`, `ventas_diarias`, `inversion_diaria` y `plataforma`.
- Se aplicaron validaciones para detectar y corregir duplicados, inconsistencias o formatos inválidos en los registros procesados.
- El proceso de consolidación se diseñó de manera automatizada y reproducible para su ejecución periódica sin intervención manual.

- El *dataset* resultante se almacenó en un formato estructurado apto para ser consumido directamente en las etapas de modelado y entrenamiento.
- Se registraron las operaciones del proceso mediante archivos de log, incluyendo la cantidad total de registros consolidados y cualquier incidencia detectada durante la ejecución.

Requerimiento funcional 4

Para cumplir con este requerimiento se implementó un proceso de modelado y entrenamiento estructurado que incluyó las siguientes etapas:

- Se empleó el *dataset* consolidado, compuesto por variables de ventas y de inversión publicitaria, como fuente principal para el entrenamiento del modelo.
- El conjunto de datos fue dividido en subconjuntos de entrenamiento y validación, siguiendo una proporción aproximada de 80 % para entrenamiento y 20 % para validación.
- Se evaluaron distintos modelos de predicción utilizando métricas de error adecuadas para series temporales, tales como MAE, RMSE y MAPE, con el fin de comparar su desempeño.
- Se seleccionó como modelo final aquel que presentó el mejor desempeño sobre el conjunto de validación.
- El modelo entrenado quedó habilitado para generar predicciones futuras a partir de valores recientes de entrada.
- Se garantizó la reproducibilidad del proceso mediante la definición de un *pipeline* de entrenamiento que permite repetir la ejecución bajo las mismas configuraciones.
- El modelo seleccionado y sus parámetros finales fueron almacenados para su posterior uso dentro del sistema Inventory Tracker.
- Se documentó el proceso de entrenamiento de manera detallada, incluyendo la selección de variables, las técnicas de preprocesamiento aplicadas y los métodos de evaluación utilizados.
- Se realizó una validación visual comparando las predicciones generadas frente a los valores reales observados, mediante gráficos que permitieron analizar la coherencia del modelo respecto a la dinámica histórica de las ventas.

Requerimiento funcional 5

Para cumplir con este requerimiento se implementaron las siguientes funcionalidades:

- Las predicciones de ventas se muestran en la interfaz de Inventory Tracker junto a las fechas correspondientes.
- La visualización integra tanto las predicciones como los datos históricos de ventas para comparaciones temporales y análisis de tendencias.

- La información se presenta de manera clara y entendible, utilizando gráficos de líneas y tablas según corresponda.
- Los usuarios pueden seleccionar rangos de fechas específicos para observar predicciones detalladas de periodos concretos.
- Las predicciones se actualizan automáticamente al entrenarse un nuevo modelo o al actualizarse los datos de entrada y se garantiza que la información refleje siempre el estado más reciente.
- La interfaz muestra un indicador visual de la fecha y hora de generación de la última predicción.
- Se distingue claramente entre valores predichos y datos históricos que evitan confusiones en el análisis.
- La integración de las predicciones no afecta la funcionalidad ni el rendimiento general de la interfaz del módulo Inventory Tracker.
- El rendimiento de la interfaz se mantiene aceptable incluso cuando se manejan conjuntos de datos de gran tamaño.

Requerimiento funcional 6

Para cumplir con este requerimiento se implementaron las siguientes funcionalidades:

- Se incorporó un botón o enlace claramente identificado que permite la descarga de las predicciones en formato CSV.
- El archivo CSV generado incluye las fechas y los valores de predicción correspondientes.
- De manera opcional, el archivo puede incluir los datos históricos de ventas para permitir la comparación con las predicciones.
- El formato del CSV es compatible con herramientas comunes de análisis, con valores separados por comas y codificación UTF-8.
- El nombre del archivo contiene la fecha y hora de generación que lo identifica.
- La descarga se realiza correctamente en navegadores modernos, incluyendo Chrome y Firefox.
- El contenido del archivo refleja exactamente la información presentada en la interfaz del módulo de predicción.
- Si aún no existen predicciones generadas, el botón de descarga se muestra deshabilitado.

Requerimiento funcional 7

Para cumplir con este requerimiento se implementaron las siguientes funcionalidades:

- El sistema verifica automáticamente la integridad del *dataset* antes de cada entrenamiento del modelo.

- Si se detectan datos faltantes críticos como fechas sin registros de ventas o de inversión publicitaria se genera una alerta.
- La alerta incluye información detallada sobre el tipo de dato faltante (ventas, inversión, o ambos), el rango de fechas afectado y el nivel de severidad del impacto estimado.
- La alerta se muestra de forma visible dentro del sistema para que los responsables puedan tomar conocimiento inmediato.
- Se envía una notificación por correo electrónico al responsable del sistema, en caso de estar configurada esta opción.
- El sistema no bloquea el entrenamiento del modelo, pero advierte que la precisión de las predicciones puede verse afectada.
- La alerta desaparece únicamente cuando los datos faltantes son completados o cuando se marca como revisada manualmente.
- Se registra un log de todas las alertas emitidas, accesible desde una sección de administración o monitoreo del sistema.

Capítulo 5

Conclusiones

Todos los capítulos deben comenzar con un breve párrafo introductorio que indique cuál es el contenido que se encontrará al leerlo. La redacción sobre el contenido de la memoria debe hacerse en presente y todo lo referido al proyecto en pasado, siempre de modo impersonal.

5.1. Resultados obtenidos

La idea de esta sección es resaltar cuáles son los principales aportes del trabajo realizado y cómo se podría continuar. Debe ser especialmente breve y concisa. Es buena idea usar un listado para enumerar los logros obtenidos.

En esta sección no se deben incluir ni tablas ni gráficos.

Algunas preguntas que pueden servir para completar este capítulo:

- ¿Cuál es el grado de cumplimiento de los requerimientos?
- ¿Cuán fielmente se puedo seguir la planificación original (cronograma incluido)?
- ¿Se manifestó algunos de los riesgos identificados en la planificación? ¿Fue efectivo el plan de mitigación? ¿Se debió aplicar alguna otra acción no contemplada previamente?
- Si se debieron hacer modificaciones a lo planificado ¿Cuáles fueron las causas y los efectos?
- ¿Qué técnicas resultaron útiles para el desarrollo del proyecto y cuáles no tanto?

5.2. Trabajo futuro

Acá se indica cómo se podría continuar el trabajo más adelante.

Bibliografía

[1] *LaTechFactory*. https://latechfactory.com/. Accedido: 2025-11-23.

[2] *TikTok for Business*. https://business.tiktok.com/en/. Accedido: 2025-11-23.

[3] *Instagram for Business*. https://business.instagram.com/. Accedido: 2025-11-23.

[4] *Facebook for Business*. https://business.facebook.com/. Accedido: 2025-11-23.

[5] *Google Ads*. https://ads.google.com/. Accedido: 2025-11-23.

[6] *Triple Whale API*. https://triplewhale.com/api. Accedido: 2025-11-09.

[7] *Shopify API*. https://shopify.dev/api. Accedido: 2025-11-09.

[8] George E. P. Box et al. *Time Series Analysis: Forecasting and Control*. 5th. Wiley, 2015.

[9] Rob J Hyndman y George Athanasopoulos. *Forecasting: principles and practice*. 2nd. OTexts, 2018. URL: https://otexts.com/fpp3/.

[10] Sepp Hochreiter y Jürgen Schmidhuber. «Long short-term memory». En: *Neural Computation*. Vol. 9. 8. MIT Press, 1997, págs. 1735-1780.

[11] Ian Goodfellow, Yoshua Bengio y Aaron Courville. *Deep Learning*. MIT Press, 2016. URL: https://www.deeplearningbook.org/.

[12] Leo Breiman. «Random forests». En: *Machine Learning* 45.1 (2001), págs. 5-32.

[13] Kasun Bandara, Christoph Bergmeir y Slawek Smyl. «Forecasting sales with machine learning in e-commerce». En: *International Journal of Forecasting* (2020).

[14] Bryan Lim et al. «Temporal Fusion Transformers for interpretable multi-horizon time series forecasting». En: *International Journal of Forecasting* 37.4 (2021), págs. 1748-1764. URL: https://www.sciencedirect.com/science/article/pii/S0169207021000637.

[15] Haoyi Zhou et al. «Informer: Beyond efficient transformer for long sequence time-series forecasting». En: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.12 (2021), págs. 11106-11115.

[16] Boris N Oreshkin et al. «N-BEATS: Neural basis expansion analysis for interpretable time series forecasting». En: *arXiv preprint arXiv:1905.10437* (2019).

[17] Spyros Makridakis, Evangelos Spiliotis y Vassilios Assimakopoulos. «The M5 competition: Background, organization, and implementation». En: *International Journal of Forecasting* 36.4 (2020), págs. 1451-1457.

[18] *Amazon Forecast*. https://aws.amazon.com/forecast/. Accedido: 2025-11-09.

[19] *Amazon Web Services (AWS)*. https://aws.amazon.com/. Accedido: 2025-11-09.

[20] Sean J Taylor y Ben Letham. *Forecasting at scale*. Vol. 72. 1. Taylor & Francis, 2018, págs. 37-45.

[21] *Meta for Developers*. https://developers.facebook.com/. Accedido: 2025-11-09.

[22] *RandomForestRegressor — scikit-learn documentation*. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html. Accedido: 2025-11-23.

[23] Jerome H. Friedman. *Greedy Function Approximation: A Gradient Boosting Machine*. https://jerryfriedman.su.domains/ftp/trebst.pdf. Accedido: 2025-11-23. 2001. DOI: 10.1214/aos/1013203451.

[24] Rob J. Hyndman et al. *Forecasting with Exponential Smoothing: The State Space Approach*. Accedido: 2025-11-23. 2008. URL: https://www.academia.edu/76833440/Forecasting_with_Exponential_Smoothing_The_State_Space_Approach_by_Rob_J_Hyndman_Anne_B_Koehler_J_Keith_Ord_Ralph_D_Snyder.

[25] Kyunghyun Cho et al. «Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation». En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Accedido: 2025-11-23. ACL. 2014, págs. 1724-1734. URL: https://arxiv.org/abs/1406.1078.

[26] Jeff Reback et al. «pandas: powerful Python data analysis toolkit». En: *Zenodo* (2020). DOI: 10.5281/zenodo.3509134. URL: https://pandas.pydata.org/.

[27] Charles R. Harris et al. «Array programming with NumPy». En: *Nature* 585.7825 (2020), págs. 357-362. DOI: 10.1038/s41586-020-2649-2. URL: https://numpy.org/doc/stable/.

[28] Fabian Pedregosa et al. «Scikit-learn: Machine learning in Python». En: *Journal of Machine Learning Research* 12 (2011), págs. 2825-2830. URL: https://scikit-learn.org/stable/.

[29] Adam Paszke, Sam Gross, Francisco Massa et al. *PyTorch: An Open Source Machine Learning Framework*. https://pytorch.org/. Accedido: 2025-02-13. PyTorch Foundation, 2025.

[30] John D. Hunter. *Matplotlib: A 2D graphics environment*. Vol. 9. 3. IEEE, 2007, págs. 90-95. URL: https://matplotlib.org/.

[31] Michael L. Waskom. «Seaborn: statistical data visualization». En: *Journal of Open Source Software* 6.60 (2021), pág. 3021. DOI: 10.21105/joss.03021. URL: https://seaborn.pydata.org/.

[32] Plotly Technologies Inc. *Plotly Python Open Source Graphing Library*. https://plotly.com/python/. Accedido: 2025-11-23. 2015.

[33] The PostgreSQL Global Development Group. *PostgreSQL Documentation*. https://www.postgresql.org/docs/current/. Consultado: 2 de diciembre de 2025. 2024.

[34] *Cron Manual*. https://man7.org/linux/man-pages/man5/crontab.5.html. Accedido: 2025-02-13.

[35] *Amazon Simple Storage Service (S3) Documentation*. Accedido: 2025-02-13. Amazon Web Services. 2025.

[36] Scott Chacon y Ben Straub. *Pro Git*. 2.ª ed. Accedido: 2025-02-13. Apress, 2014. URL: https://git-scm.com/book/en/v2.

[37] ASGI Community. *ASGI: Asynchronous Server Gateway Interface*. https://asgi.readthedocs.io/en/latest/. Acceso: 2025-11-27. 2024.

[38] Pydantic Team. *Pydantic Documentation*. https://docs.pydantic.dev/latest/. Acceso: 2025-11-27. 2024.

[39] Swagger API. *Swagger: API Design Tools and Documentation*. https://swagger.io/. Acceso: 2025-11-27. 2024.

[40] Docker, Inc. *Docker Documentation*. https://docs.docker.com/. Acceso: 2025-11-27. 2024.

[41] *Docker Compose Documentation*. https://docs.docker.com/compose/. Accedido: 2025-02-13. Docker, Inc., 2025.
[42] Metaflow Team, Netflix. *Metaflow: Human-Centric Framework for Data Science*. https://docs.metaflow.org/. Acceso: 2025-11-27. 2024.
[43] Apache Software Foundation. *Apache Airflow Documentation*. https://airflow.apache.org/docs/. Acceso: 2025-11-27. 2024.
[44] *Directed Acyclic Graphs (DAGs) in Apache Airflow*. https://airflow.apache.org/docs/apache-airflow/stable/core-concepts/dags.html. Accedido: 2025-11-27.
[45] *¿Qué es el proceso ETL?* https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-etl/. Consultado: 2 de diciembre de 2025.
[46] *RabbitMQ: Messaging that just works*. https://www.rabbitmq.com/. Accedido: 2025-12-02.
[47] *Redis: In-memory data structure store*. https://redis.io/. Accedido: 2025-12-02.