

Sistema de gestión predictiva de inventarios basado en redes neuronales profundas para retail de bebidas.

Luca Biava

Universidad Nacional de Rafaela

Lic. en Administración y Gestión de la Información

Docente Romina Gentinetta

20 de abril de 2026

Quisiera comenzar expresando mi más sincero agradecimiento a mi familia, en especial a Guillermina. Su fe en mí ha sido el motor que me permitió completar este camino.

Introducción.	7
1.1. Empresa involucrada.	8
1.2. Organigrama general.	9
1.3. Diagnóstico e identificación del problema.	10
Marco Teórico.	13
2.1. Antecedentes de la investigación.	13
2.1.2. Desafíos de digitalización en empresas pyme retail.	14
2.1.3. Aplicación de analítica de datos en retail de bebidas.	14
2.1.4. Limitaciones de planillas de cálculo en gestión de inventarios.	15
2.2. Bases teóricas y modelos de aplicación.	16
2.2.1. Modelos clásicos de gestión de inventarios.	16
2.2.2. Modelos de revisión periódica.	17
2.2.3. La Demanda como variable aleatoria o variable predecible.	17
2.2.4. Arquitecturas de sistemas de información para inventarios.	18
2.2.5. El rol de la analítica predictiva en inventarios modernos.	18
2.3. Forecasting de demanda, de métodos clásicos a machine learning.	19
2.3.1. Fundamentos de series temporales.	20
2.3.2. De estadística clásica a machine learning.	20
2.3.3. Deep learning, aprendizaje de representaciones jerárquicas.	21
2.3.4 Regularizaciones.	21
2.3.5 Aplicabilidad a forecasting de inventarios.	21
2.4. Feature engineering.	22
2.4.1. Features temporales como memoria de un pasado reciente.	22
2.4.2. Features contextuales, ciclos y estacionalidad.	23
2.4.3. Features comerciales, promociones y su dinámica.	23
2.4.4. Features de categorización, segmentación de productos.	23
2.4.5. Features de interacción, efectos sinérgicos.	23
2.4.6. Validación y limpieza de features.	24
2.5. Evaluación de modelos de forecasting.	24
2.5.1. Métricas de error punto a punto.	24
2.5.2. Validación temporal.	25
2.5.3. El gap entre la validación offline y el rendimiento online.	25
2.5.4. Consideraciones específicas para redes neuronales.	26

2.6. Síntesis.....	27
2.6.1. Integración conceptual, del problema a la solución.....	27
2.6.2. Justificación de deep learning para el contexto específico.....	27
2.6.3. El rol del feature engineering en deep learning.....	28
2.6.4. Reflexión personal sobre la brecha teoría-práctica.....	28
2.6.5. Transición hacia la metodología.....	29
Objetivos.....	30
3.1. Objetivo general.....	30
3.2. Objetivos específicos.....	30
Desarrollo.....	31
4. Fase 1: Recolección y limpieza de datos.....	31
5. Fase 2: Ingeniería de features y transformación de datos.....	35
5.1. Features temporales.....	35
5.2. Lags.....	35
5.3. Promedios móviles.....	36
5.4. Momentum, crecimiento y aceleración.....	36
5.5. Features contextuales para capturar los ciclos estacionales.....	36
5.6. Flags de estacionalidad.....	37
5.7. Features comerciales.....	37
5.8. Variables de ofertas básicas.....	37
5.9. Features de transición promocional.....	37
5.10. Intensidad y duración de ofertas.....	37
5.11. Features de categorización para segmentado de productos y locales.....	38
5.12. Prevención de data leakage.....	39
5.13. Features de interpretación para capturar efectos sinérgicos.....	39
5.2.1. Limpieza y validación final.....	40
5.2.2. Síntesis.....	41
6. Fase 3: Diseño y arquitectura del modelo predictivo.....	43
6.1. Justificación de la arquitectura.....	43
6.2. Topología de capas.....	44
6.3. Funciones de activación.....	45
6.4. Estrategias de regularización para prevenir el overfitting.....	47
6.5. Optimizador y configuración del entrenamiento.....	48

6.6. Síntesis arquitectónica.	48
7. Fase 4: Entrenamiento, calibración y validación del modelo.	48
7.1. División temporal del dataset.	49
7.2. Validación cruzada temporal.	49
7.3. Normalización de Features.	49
7.4. Configuración del proceso de entrenamiento.	50
7.5. Epochs y early stopping.	50
7.6. Afinación adaptativa con ReduceLROnPlateau.	50
7.7. Métricas de evaluación en test set.	50
7.8. Root Mean Squared Error (RMSE).	51
7.9. Coeficiente de determinación (R^2).	51
7.10. Distribución de errores.	51
7.11. Análisis de curvas de aprendizaje.	52
7.2.1. Comparación con caminos alternativos no tomados.	53
7.2.2. Síntesis.	54
8. Fase 5: Implementación del piloto y generación de recomendaciones.	55
8.1. Preparación de features para producción.	55
8.2. Features temporales.	55
8.3. Features climáticas.	56
8.4. Features comerciales.	56
8.5. Features contextuales.	56
8.6. Features de categorización.	56
8.7. Generación de predicciones.	57
8.8. Post procesamiento.	57
8.9. Validación de rangos extremos.	57
8.10. Estructuración del output.	57
9. Fase 6: Evaluación de resultados y análisis de desempeño.	58
9.1. Métricas globales de desempeño.	58
9.1.2. Error Absoluto Medio (MAE).	58
9.1.3. Sesgo direccional (Bias).	59
9.1.4. Distribución del error absoluto.	59
9.1.5. Análisis de segmentación por volumen de ventas.	59
9.1.6. Análisis por familia de productos.	60

9.2. Casos representativos.....	61
9.3. Desafíos en productos con ofertas de entrada.....	62
9.4. Comparación con baseline naive.	62
9.5. Proyección de impacto operativo y valor de negocio.....	62
9.6. Reducción de capital inmovilizado.....	63
9.7. Mejora en nivel de servicio.	63
9.8. Síntesis.	63
Discusión.....	65
10.1. Interpretación del gap entre validación y producción.....	65
10.2. Precisión del input.	66
10.3. Concentración del error en subpoblación minoritaria.	66
10.4. Evaluación del cumplimiento de objetivos.	67
Limitaciones del Estudio.....	70
11.2. Contribuciones del trabajo.....	72
11.2.1. Contribución metodológica.	73
11.2.2. Contribución técnica.	74
11.2.3. Contribución práctica.....	75
11.3. Síntesis.	76
Conclusiones.	77
12.1. Síntesis de logros.	77
12.2. Logros técnicos.....	77
12.3. Logros metodológicos.....	78
12.4. Logros empíricos.	78
12.5. Logros prácticos.	79
12.6. Respuesta a los objetivos planteados.....	79
Reflexión final sobre BrewCast.	79
Glosario de términos fundamentales para la comprensión del documento.....	81
Bibliografía.	89
Anexos.	95

Introducción.

En el entorno competitivo actual del retail, caracterizado por la volatilidad de la demanda y márgenes de rentabilidad ajustados, las organizaciones que aspiran a mantener su posición en el mercado deben fundamentar sus operaciones en la precisión y la eficiencia. Para Beer Market, empresa consolidada en el mercado de bebidas de Buenos Aires con una red de 18 locales, este desafío se traduce en la necesidad de transformar su gestión de inventarios desde prácticas tradicionales hacia enfoques basados en analítica de datos. La gestión de inventarios deja de ser una función logística aislada para convertirse en proceso que impacta directamente en la disponibilidad de producto, la rentabilidad operativa y la capacidad de responder ágilmente a cambios en patrones de consumo.

Este trabajo de tesis aborda la problemática del forecasting de demanda en una pyme de retail. El punto de partida es la situación actual de Beer Market, donde la gestión de inventarios opera sobre planillas de Excel fragmentadas, actualizadas discrecionalmente y sin integración con sistemas transaccionales. Esta realidad genera incapacidad para anticipar demanda futura con precisión, resultando en dos problemas recurrentes que erosionan rentabilidad. Quiebres de stock en productos de alta rotación (ventas perdidas, insatisfacción de clientes) y exceso de inventario en productos de baja rotación (capital inmovilizado, riesgo de obsolescencia).

Los costos de esta ineficiencia son tangibles. El capital inmovilizado en inventario excesivo representa costo de oportunidad en un contexto inflacionario argentino donde la licuación del valor adquisitivo alcanza tasas elevadas. Cada peso atrapado en mercadería estancada es un peso que no se reinvierte en productos de mayor rotación o en expansión del negocio. Simultáneamente, los quiebres generan una pérdida de margen de contribución irrecuperable, en retail de bebidas con márgenes que suelen variar entre un 5 y un 8%, perder una venta equivale a necesitar de 12 a 20 ventas adicionales para compensar la oportunidad perdida.

La gestión moderna de inventarios ha evolucionado desde modelos determinísticos simples hacia enfoques de analítica predictiva que explotan datos granulares para anticipar demanda con mayor exactitud. Los métodos estadísticos clásicos capturan una estructura temporal básica, pero asumen linealidad y tienen capacidad limitada para incorporar múltiples variables causales. Los métodos de machine learning (Gradient Boosting, redes neuronales) reformulan forecasting como problema de regresión supervisada donde los features armados (lags de ventas, encoding de estacionalidad, indicadores de ofertas) alimentan modelos que aprenden relaciones no lineales e interacciones complejas automáticamente.

Entre las alternativas de machine learning, las redes neuronales profundas ofrecen arquitectura en capas que permite aprender representaciones jerárquicas: capas iniciales detectan patrones simples (producto en oferta, temperatura alta), capas intermedias los combinan en conceptos de nivel medio (oferta en producto premium durante verano), y capas finales integran todo en predicción de demanda. Esta capacidad de abstracción progresiva es particularmente valiosa en retail de bebidas donde múltiples factores interactúan de formas complejas: el efecto de temperatura sobre ventas de cerveza es mayor cuando hay oferta activa (sinergia clima-promoción), las promociones experimentan fatiga donde su impacto decae con la duración incluso si los términos permanecen constantes, y diferentes familias de productos exhiben sensibilidades diferenciadas.

El documento se organiza en once capítulos que recorren el ciclo completo de desarrollo, implementación y evaluación del sistema.

El Capítulo 2 (Marco Teórico) revisa la literatura académica que fundamenta las decisiones metodológicas del proyecto, abarcando modelos de gestión de inventarios, evolución de métodos de forecasting desde estadística clásica hacia machine learning, principios de feature engineering, y criterios de evaluación de modelos predictivos.

Los Capítulos 4 a 9 documentan las seis fases del desarrollo técnico. El Capítulo 4 (Fase 1) describe el proceso ETL que integra cinco fuentes heterogéneas en un dataset unificado. El Capítulo 5 (Fase 2) detalla la construcción de las features distribuidas en categorías funcionales. El Capítulo 6 (Fase 3) justifica la topología de red neuronal seleccionada. El Capítulo 7 (Fase 4) documenta el proceso de calibración del modelo. El Capítulo 8 (Fase 5) describe la generación de predicciones prospectivas para la semana objetivo y su entrega al equipo de compras. El Capítulo 9 (Fase 6) analiza el desempeño en producción.

El Capítulo 10 (Discusión) interpreta los hallazgos a la luz de la literatura, analizando el gap entre validación retrospectiva y producción prospectiva, evaluando el cumplimiento de objetivos, documentando limitaciones del estudio y articulando las contribuciones metodológicas, técnicas y prácticas del trabajo.

El Capítulo 12 (Conclusiones y Recomendaciones) sintetiza los logros alcanzados, reflexiona sobre las lecciones aprendidas de implementar machine learning en una empresa real, y propone líneas de investigación futura.

1.1. Empresa involucrada.

La institución seleccionada para el desarrollo del presente trabajo es Beer Market, una empresa dedicada a la comercialización minorista de bebidas alcohólicas y energizantes, así como accesorios y productos afines. Cuenta con una presencia consolidada dentro del sector de bebidas en su región de influencia (CABA y GBA).

Beer Market se orienta a la obtención de beneficios económicos mediante la comercialización de bebidas alcohólicas y productos relacionados, con el propósito de maximizar su rentabilidad y fortalecer su posicionamiento en el nicho de mercado de bebidas. Esta orientación posiciona a la empresa como una organización con fines de lucro

La empresa presenta una estructura organizacional definida para su escala, lo que le confiere el carácter de organización formal, caracterizada por la existencia de roles específicos (gerencia, atención al cliente, compras, logística) y procedimientos básicos que regulan la operación y la comunicación.

A nivel de toma de decisiones, si bien las directrices estratégicas (inversiones, expansión, precios) son determinadas centralmente por la gerencia/propietario, se observa un grado de autonomía en niveles operativos. Esta disposición favorece una respuesta ágil a las dinámicas del mercado y dota a la empresa de un carácter semi-descentralizado en su operativa diaria.

Desde la perspectiva de su actividad, Beer Market se especializa en la comercialización minorista (retail) de bebidas cerveceras. En cuanto a su ámbito geográfico, se trata de una organización de alcance local/regional.

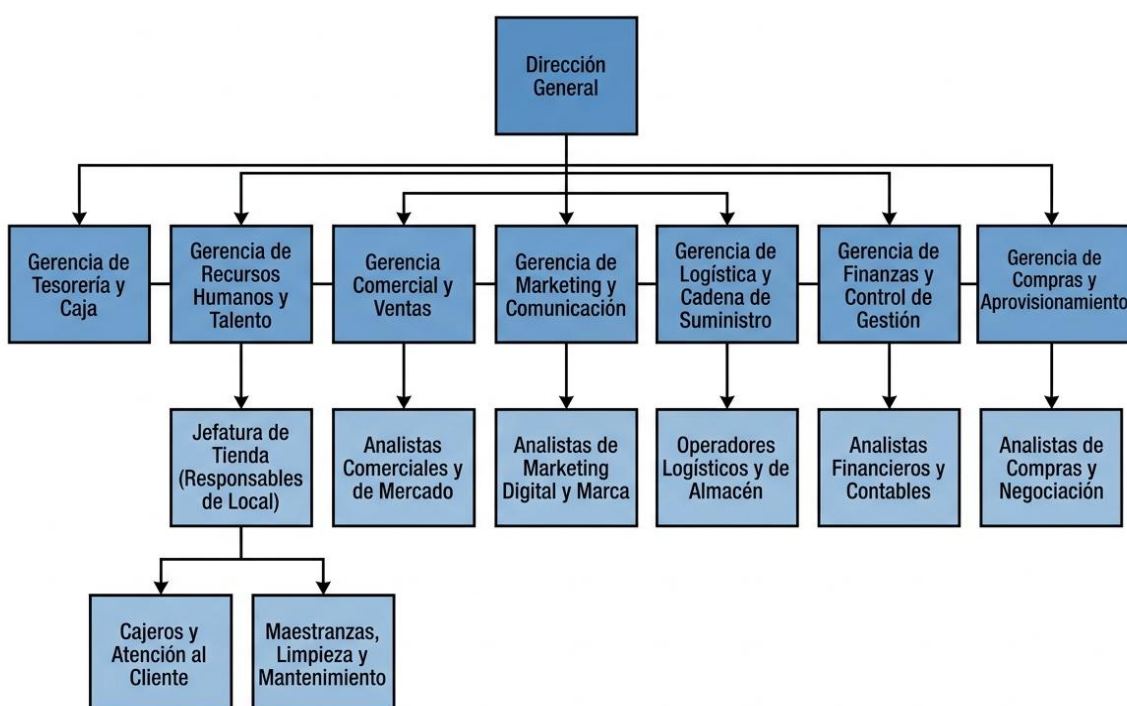
La empresa es de propiedad privada, sin participación estatal en su capital. Desde su forma jurídica, para este ejemplo se constituye como una Sociedad de Responsabilidad Limitada (S.R.L.), en la cual la responsabilidad de los socios está limitada al capital aportado.

En cuanto a su estructura de autoridad, Beer Market mantiene un equilibrio entre la centralización de la autoridad para la toma de decisiones estratégicas de alto impacto y una autoridad delegada para decisiones operativas y tácticas. Por esta razón, y en línea con lo observado en el ejemplo, se clasifica como una organización de autoridad mixta.

1.2. Organigrama general.

Figura 1

Diagrama del organigrama de la empresa, siendo una elaboración propia a partir del original.



1.3. Diagnóstico e identificación del problema.

Tras la pandemia, una combinación de políticas económicas expansivas como el congelamiento de tarifas de servicios públicos, el "dólar soja" para inyectar divisas, y una fuerte emisión monetaria generó lo que se percibió como una "plata dulce". Este flujo artificial de liquidez, sumado a la liberación del consumo reprimido durante la cuarentena, desató una fiebre compradora. Entre 2021 y principios de 2023, el retail, y especialmente el rubro de bebidas, experimentó una demanda sostenida y poco selectiva. Los consumidores, con dinero en efectivo y una mentalidad de "gasto terapéutico", priorizaban la disponibilidad inmediata y las experiencias por sobre el precio, llevando a niveles récord la rotación de productos premium y generando una sensación de crecimiento perpetuo. Este entorno permitió a muchas empresas operar con márgenes de error amplios; los excesos de inventario se liquidaban con relativa facilidad y los quiebres de stock se cubrían con ventas cruzadas. Sin embargo, esta euforia era insostenible. Era un espejismo alimentado por estímulos artificiales que, una vez retirados, dejaron al descubierto la fragilidad estructural de un sistema de gestión que no estaba preparado para la austeridad.

El período posterior a la pandemia configuró un escenario económico global de desaceleración y ajuste monetario. Para 2024, se proyectaba un crecimiento mundial del 2.9% (FMI), en un entorno donde la persistencia de presiones inflacionarias llevó a los principales bancos centrales a mantener políticas monetarias contractivas. Esta combinación de crecimiento lento y dinero caro sentó las bases para una contracción del consumo a escala global, afectando particularmente a los bienes discrecionales.

En Argentina, este contexto internacional se potenció por una serie de variables domésticas críticas. La economía ingresó en una recesión técnica, con una caída acumulada del 2.6% en el primer semestre de 2023 y una proyección de contracción del 2.8% para todo el año (INDEC). Este escenario recesivo fue impulsado por una política de severa devaluación del tipo de cambio oficial, que superó el 100% interanual, y una inflación que se consolidó como una de las más altas del mundo, con índices interanuales que rondaron el 140%. Estas políticas, orientadas a corregir desequilibrios macroeconómicos heredados, tuvieron un impacto directo y inmediato en el bolsillo de los consumidores. El poder adquisitivo se vio dramáticamente erosionado, forzando una reconfiguración total de los hábitos de gasto.

El shock de precios, sumado a la incertidumbre laboral y la pérdida de ahorros, terminó con el ciclo de consumo masivo que había caracterizado la salida de la pandemia. Según un informe de la Cámara Argentina de Comercio, se registró un descenso del 5.4% en el volumen de ventas en supermercados durante el tercer trimestre de 2023. El consumidor argentino mutó de un perfil impulsivo a uno racional y altamente selectivo.

Este nuevo paradigma convirtió a la gestión de inventarios en la línea que separa la rentabilidad de la pérdida. La inflación galopante y las devaluaciones recurrentes implican que el costo de reposición de los productos se dispara mes a mes. Mantener un exceso de stock de productos de lenta rotación se volvió un lujo prohibitivo, ya que el capital inmovilizado no solo no genera rendimiento, sino que se licúa rápidamente frente a la inflación. Por el contrario, los quiebres de stock en los productos de alta rotación representan una pérdida de venta irrevocable ante un consumidor que, si no encuentra lo que busca, no duda en buscar alternativas en la competencia.

La combinación de recesión, inflación y un consumidor "ahorrista" crea una tormenta perfecta que exige una precisión quirúrgica en la gestión. Las empresas que dependen de sistemas reactivos,

basados en planillas manuales y actualizaciones discrecionales, quedan expuestas a un doble riesgo: por un lado, la sobrecompra y el consecuente capital inmovilizado en productos que no se venden, y por el otro, el desabastecimiento y la pérdida de participación de mercado en los productos más buscados. En este contexto, la capacidad de predecir la demanda, optimizar el surtido y alinear las compras con el consumo real dejó de ser una ventaja competitiva para convertirse en un imperativo de supervivencia. La eficiencia operativa, sustentada en datos en tiempo real y análisis predictivo, se adopta como el principal resguardo frente a la volatilidad macroeconómica.

Los estudios de consultoras privadas CCR Insights reflejaron que cerca de un 70% de los argentinos declaró priorizar el ahorro por sobre el gasto inmediato, mientras que un 65% afirmó realizar una búsqueda exhaustiva de precios y promociones antes de realizar una compra. La lealtad de marca se quebró, dando paso a una búsqueda obsesiva de valor. En el sector específico de bebidas, esto se tradujo en una migración de consumidores desde marcas premium a opciones standard o de segunda marca, y en una mayor frecuencia de compra de unidades sueltas en lugar de packs, en un esfuerzo por administrar con extrema precisión el presupuesto familiar. OCR Insights. (2024). Relevamiento de Expectativas de Mercado (REM): Proyecciones macroeconómicas Argentina 2024. OCR Macro.

Este desafío se traduce en la necesidad de transformar radicalmente su gestión de inventarios. La propuesta de valor ya no reside sólo en reducir el margen de error operativo, sino en implementar sistemas de inteligencia de negocio que integren información en tiempo real para la toma de decisiones estratégicas, optimicen el capital de trabajo y maximicen la rotación de mercadería. La industria avanza hacia un estándar donde la implementación de estas tecnologías ya no es un lujo, sino un requisito para mantener la relevancia en el mercado. En este contexto, la gestión de inventarios deja de ser una función logística aislada para convertirse en el núcleo estratégico que impacta directamente en la satisfacción del cliente, la rentabilidad y la ventaja competitiva sostenible.

A continuación, se presenta un listado de obsolescencias detectadas durante el análisis de la situación actual.

Fragmentación severa de la información: La existencia de múltiples archivos de Excel, actualizados de manera discrecional por diferentes actores (encargados de locales, compras, gerencia) y sin un protocolo de integración unificado, genera una visión distorsionada y atomizada del inventario. No existe una "fuente única de la verdad" en tiempo real.

Actualización discrecional y con desfases: La actualización de los stocks se realiza de manera periódica (por ejemplo, al cierre del día) y no de forma continua. Esto crea ventanas de tiempo donde la información está obsoleta, impidiendo la toma de decisiones ágiles y basadas en el estado actual del negocio.

Elevada propensión a errores humanos: La manipulación manual de datos, mediante copia y pegue entre sistemas y la consolidación de planillas, incrementa exponencialmente el riesgo de errores de tipeo, duplicación de registros u omisión de movimientos, lo que distorsiona por completo el control del stock.

Falta de integración y sincronización entre departamentos: Los departamentos de Ventas, Compras y Logística operan con conjuntos de datos diferentes y desincronizados. Un vendedor

puede prometer un producto que, según el sistema de compras, ya está agotado, generando insatisfacción del cliente y daño reputacional.

Escalabilidad limitada: El modelo basado en planillas se vuelve ingobernable e inmanejable ante un crecimiento de la empresa, como la incorporación de nuevos locales o una ampliación de la gama de productos. La complejidad operativa crece de forma exponencial, mientras que la herramienta permanece estática.

Esta realidad limita la capacidad de la compañía para contar con una visión unificada y en tiempo real del stock, lo que deriva en una exposición constante a riesgos críticos. Los problemas, sin embargo, trascienden la mera falta de sincronización y se materializan en consecuencias financieras directas y cuantificables.

Elevados costos por capital inmovilizado: La incapacidad para predecir la demanda conlleva a la sobrecompra de productos de lenta rotación como consecuencia de un modelo reactivo. Esto inmoviliza capital financiero valioso en mercadería estancada en depósitos, que podría estar destinado a inversiones de crecimiento o a la adquisición de productos de alta rotación.

Compras subóptimas: La falta de datos predictivos impide aprovechar economías de escala en momentos de precio favorable con los proveedores, así como realizar compras estratégicas alineadas con promociones o tendencias de consumo. Se compra "a ciegas", reaccionando a los faltantes en lugar de anticipándose a la demanda.

Incapacidad estructural para la anticipación: El sistema es inherentemente reactivo. No puede predecir picos de demanda estacionales (como fines de semana largos o verano) o identificar tendencias de consumo emergentes, lo que genera sistemáticamente quiebres de stock en los productos más demandados y pérdida de ventas irrevocable.

Obsolescencia y pérdida de inventario: En el sector de bebidas, aunque con vencimientos menos críticos que los perecederos, existe un riesgo constante de que los productos se acerquen a su fecha de caducidad o queden obsoletos por cambios en el portafolio, resultando en mermas y pérdidas directas.

En síntesis, el problema central no radica únicamente en la falta de control sobre el inventario, sino en la incapacidad estructural de transformar los datos disponibles en información predictiva para la toma de decisiones. Como consecuencia, la empresa gestiona uno de sus activos más críticos con una lógica reactiva, exponiéndose permanentemente a pérdidas de ventas, capital inmovilizado y oportunidades de negocio desaprovechadas. En un entorno económico caracterizado por la incertidumbre y la contracción del consumo, esta limitación deja de ser un problema operativo para convertirse en un factor determinante de la competitividad y supervivencia empresarial.

Marco Teórico.

2.1. Antecedentes de la investigación.

La gestión de inventarios en contextos de alta volatilidad macroeconómica constituye un desafío crítico para la competitividad del retail, siendo un área de estudio recurrente en la literatura especializada en logística y administración. Los antecedentes de esta investigación se organizan en torno a ejes temáticos que permiten contextualizar la problemática de la empresa en análisis. Primero, se examinan los desafíos estructurales que enfrentan las pymes retail durante su proceso de digitalización, identificando los factores críticos tecnológicos, financieros y culturales que inciden en la transición desde sistemas de gestión manuales hacia plataformas integradas. Segundo, se analiza la aplicación de la analítica de datos en el sector de bebidas, con énfasis en la correlación documentada entre la profesionalización de las prácticas de inventario y la mejora de indicadores financieros clave, como el margen operativo. Finalmente, y como puente hacia la propuesta de esta tesis, se revisan estudios que detallan las limitaciones inherentes al uso de planillas de cálculo para la gestión de inventarios, enumerando los errores operativos más frecuentes y las estrategias de mitigación inmediata que han sido validadas empíricamente, sentando así las bases para el diseño de una solución sistémica.

Ayzah Shaheen Iqbal dice que “The Inventory Management System Maturity Model (IMSMM) was completed with the company’s inventory manager to assess the IMS across the 5 maturity levels and 4 dimensions.” (Iqbal, 2025, p. 163)

En su tesis, Iqbal (2025) propone un modelo de madurez al que denomina IMSMM (Inventory Management System Maturity Model) para medir y mejorar la gestión de inventarios en pymes desde una óptica sociotécnica. El modelo se estructura en cinco niveles de madurez, articulados a lo largo de cuatro dimensiones fundamentales: organización, personas, procesos y tecnología.

En el Nivel 1 (Inicial), las empresas se caracterizan por gestión ad-hoc basada en hojas de cálculo sin estandarización formal, donde las decisiones de reposición dependen de la memoria individual de encargados de local o compradores. La información está fragmentada en múltiples archivos actualizados discrecionalmente, sin protocolo de validación ni integración entre departamentos. Los procesos son reactivos: se compra cuando se detecta faltante visualmente, sin proyección de demanda futura.

El Nivel 2 (Definido) introduce estandarización básica mediante plantillas compartidas y procedimientos documentados para conteo de inventario y emisión de órdenes de compra. Sin embargo, la ejecución sigue siendo manual y la sincronización entre ventas y compras depende de reuniones periódicas donde se consolidan datos de múltiples fuentes.

En el Nivel 3 (Gestionado), se adoptan sistemas ERP básicos que centralizan información transaccional, permitiendo visibilidad en tiempo real del stock en cada punto de venta. Los procesos incorporan puntos de reorden automáticos basados en niveles mínimos predefinidos, reduciendo la dependencia de intervención humana continua.

El Nivel 4 (Medido) integra métricas de desempeño (tasa de rotación, días de inventario, nivel de servicio) que se monitorean sistemáticamente. Se implementan sistemas de alertas que notifican excepciones (stock por debajo de mínimo, productos con movimiento cero durante N días) facilitando gestión por excepción.

Finalmente, el Nivel 5 (Optimizado) alcanza gestión plenamente integrada con datos de alta calidad alimentando modelos predictivos que anticipan demanda futura. Los procesos se ajustan dinámicamente según cambios en patrones de consumo, y existe capacidad de simulación para evaluar impacto de decisiones antes de ejecutarlas (qué pasaría si aumentamos stock de producto X en local Y).

En ese marco, la empresa que está en el Nivel 1 podría caracterizarse por una gestión ad-hoc o basada en hojas de cálculo, sin estandarización formal; mientras que en el Nivel 5 se alcanzaría una gestión plenamente integrada, con datos de inventario de alta calidad y procesos automatizados. Este planteamiento permite a la empresa diagnosticar en qué nivel se encuentra actualmente, definir cuál es el nivel deseado, y mapear las acciones necesarias para “subir” escalones.

2.1.2. Desafíos de digitalización en empresas pyme retail.

La literatura sobre transformación digital en pequeñas y medianas empresas de retail identifica barreras recurrentes que explican por qué muchas organizaciones permanecen en niveles bajos de madurez pese a la disponibilidad de tecnología accesible.

Bharadwaj et al. (2013) documentan que la resistencia al cambio es el obstáculo más frecuentemente citado por gerentes de empresas pyme. Empleados acostumbrados a planillas de Excel perciben sistemas nuevos como amenaza a su autonomía y expertise acumulado, generando sabotaje pasivo mediante subregistro de datos o negación de uso del sistema. Esta resistencia se amplifica cuando la implementación es impuesta por gerencia, sin participación de usuarios finales en el diseño.

La restricción de recursos financieros constituye una barrera estructural en las empresas pyme. Ross et al. (2016) estima que la implementación de un ERP básico en retail con 10-20 locales requiere inversión inicial entre \$50,000 y \$150,000 USD (licencias, consultoría, capacitación, migración de datos), con costos recurrentes anuales de 20-30% de la inversión inicial. Para empresas con márgenes operativos del 5-8% típicos en retail de bebidas en Argentina, esta inversión representa varios meses de utilidad total, haciéndola prohibitiva sin financiamiento externo.

La falta de expertise técnico interno agrava el problema. Raramente emplean personal dedicado a IT, dependiendo de proveedores externos para mantenimiento de sistemas. Esto genera una dependencia, cualquier modificación requiere contratar consultoría externa, incrementando el costo total de propiedad e impidiendo adaptaciones ágiles a necesidades del negocio.

Finalmente, la calidad de datos preexistentes condiciona el éxito de cualquier iniciativa de digitalización. Hazen et al. (2014) encuentran que empresas migrando de planillas Excel a ERP frecuentemente descubren que sus datos históricos contienen inconsistencias severas: productos duplicados con códigos diferentes, precios sin actualizar durante meses, registros de stock que no coinciden con el conteo físico. La limpieza de esta "deuda de datos" puede consumir 40-60% del esfuerzo total de implementación.

2.1.3. Aplicación de analítica de datos en retail de bebidas.

El sector de bebidas alcohólicas presenta características específicas que hacen tanto valiosa como desafiante la aplicación de analítica de datos para gestión de inventarios.

Waller y Fawcett (2013) identifican que productos de bebidas muestran una elasticidad de demanda alta respecto a precio y promociones. Descuentos del 20-30% pueden duplicar o triplicar ventas semanales, pero el efecto vuelve a caer rápidamente si la promoción se extiende. Esta sensibilidad implica que los modelos de forecasting deben incorporar información promocional explícitamente, no sólo patrones históricos de ventas.

La estacionalidad múltiple caracteriza la demanda de bebidas. Existe estacionalidad anual (mayor consumo en verano para cervezas livianas, mayor consumo en invierno para bebidas de mayor graduación), estacionalidad mensual (picos en fechas festivas como Navidad, Año Nuevo), y estacionalidad semanal (concentración de ventas en viernes-sábado). Kolassa y Schütz (2007) demuestran que los modelos que capturan estas múltiples estacionalidades reducen el error de predicción en 25-40% comparado con modelos que solo consideran tendencia lineal.

La sensibilidad climática es particularmente pronunciada en bebidas frías. Van Westendorp y Riezebos (2018) encuentran correlación de 0.67 entre temperatura semanal promedio y ventas de cervezas en supermercados holandeses, con umbral no lineal: incrementos de temperatura por encima de 20°C generan impacto mayor que incrementos entre 10-20°C. Esta no linealidad complica el modelado mediante regresión lineal tradicional.

La heterogeneidad de rotación entre los SKU genera un desafío adicional. En un catálogo típico de retail de bebidas, el 20% de los productos (marcas líderes como Quilmes, Coca-Cola) representan 80% de las ventas (Pareto), mientras el 50% de los productos vende menos de 5 unidades semanales. Métodos de forecasting que funcionan bien para productos de alta rotación (modelos complejos con muchos features) fallan en productos de baja rotación por overfitting, y viceversa.

Pese a estos desafíos, existen estudios de caso documentan beneficios tangibles de profesionalizar prácticas de inventario mediante analítica. Arunachal et al. (2018) reportan que cadena regional de retail de bebidas en India que implementa sistema de forecasting basado en regresión múltiple (incorporando clima, promociones, feriados) redujo quiebres de stock en 32% y capital inmovilizado en 18% durante año de piloto. El ROI fue positivo a partir del sexto mes.

2.1.4. Limitaciones de planillas de cálculo en gestión de inventarios.

La dependencia de hojas de cálculo (Excel y Google Sheets) para gestionar inventarios es casi nativo en las pymes del retail pese a las limitaciones bien documentadas en literatura de sistemas de información.

Panko (1998), en estudio seminal sobre errores en planillas, encuentra que el 88% de las hojas de cálculo empresariales contienen al menos un error, con tasa promedio de 1.5 errores por cada 100 celdas. Estos errores derivan de fórmulas incorrectas (ejemplo, sumar rango A1:A10 cuando debía ser A1:A100), referencias rotas (vincular celda de otro archivo que fue movido), y errores de tipeo en data entry manual.

La escalabilidad limitada constituye una barrera estructural. Archivos Excel con más de 50,000 filas experimentan degradación severa de performance (tiempos de apertura >30 segundos, recálculo de fórmulas lento), y el límite técnico de 1,048,576 filas es alcanzable en retail con múltiples locales y SKUs registrando transacciones diarias. Google Sheets tiene un límite aún más restrictivo de 10 millones de celdas por archivo.

La fragmentación de información emerge naturalmente, cada departamento (ventas, compras, logística) mantiene su propia planilla con subconjunto de datos, actualizadas a frecuencias diferentes, generando inconsistencias. Fisher y Raman (2010) documentan casos de retail donde las ventas reportadas por el departamento comercial diferían 15% de ventas registradas por finanzas, debido a actualizaciones desincronizadas de planillas que teóricamente debían consolidarse semanalmente, pero en la práctica se consolidaban mensualmente.

La ausencia de validación automática permite que los errores lógicos pasen inadvertidos. Si un usuario ingresa stock negativo (-50 unidades), precio cero, o fecha futura imposible (31/02), Excel acepta el valor sin alertar. Los sistemas de bases de datos relacionales previenen estos errores mediante distintos métodos de restricción de integridad.

Pese a estas limitaciones, Powell et al. (2009) argumentan que las planillas persisten porque ofrecen flexibilidad sin precedentes. cualquier usuario puede crear reportes ad-hoc, calcular métricas nuevas, y adaptar el formato según las necesidades cambiantes sin depender de IT. Esta "agilidad" tiene costo de calidad y escalabilidad, pero es valorada por gerentes de las pymes que priorizan autonomía sobre estandarización.

Los antecedentes revisados establecen que la gestión de inventarios en las pymes del retail enfrentan tensión estructural entre necesidad de profesionalización (evidenciada por modelos de madurez como IMSMM y estudios de impacto en retail de bebidas) y barreras para adopción de sistemas integrados (restricción de recursos, resistencia al cambio, limitaciones de expertise técnico). Las planillas de cálculo, aunque inadecuadas para la escala y complejidad del problema, persisten por su flexibilidad y bajo costo de entrada. Esta tensión motiva la búsqueda de soluciones intermedias que aprovechen analítica avanzada (forecasting con datos históricos, climáticos, promocionales) sin requerir inversión prohibitiva en infraestructura ERP completa.

2.2. Bases teóricas y modelos de aplicación.

La fundamentación teórica de los sistemas de gestión de inventarios se asienta sobre dos pilares disciplinarios convergentes. La teoría de operaciones y logística, que define los modelos cuantitativos para optimizar decisiones de reposición, y la teoría de sistemas de información, que establece los principios arquitectónicos para diseñar plataformas que soporten estas decisiones en entornos empresariales reales. Esta sección articula los conceptos fundamentales de ambas tradiciones, comenzando por los modelos clásicos de inventario que dominaron la práctica durante el siglo XX, continuando con la evolución hacia enfoques estocásticos que reconocen la incertidumbre inherente de la demanda, terminando con la integración de analítica de datos como habilitador de decisiones precisas.

2.2.1. Modelos clásicos de gestión de inventarios.

Los modelos clásicos de inventario buscan responder dos preguntas fundamentales: ¿cuánto ordenar? y ¿cuándo ordenar? La respuesta óptima equilibra dos tipos de costos que se mueven en direcciones opuestas: ordenar lotes grandes reduce frecuencia de pedidos (menor costo de ordenar) pero aumenta inventario promedio (mayor costo de mantener), mientras que ordenar lotes pequeños tiene el efecto inverso.

El modelo EOQ, desarrollado por Harris (1913) y popularizado por Wilson (1934), representa el punto de partida conceptual de la gestión científica de inventarios. El modelo asume demanda determinística y constante (D unidades por año), costo fijo K por emitir una orden (independiente

del tamaño del pedido), costo h de mantener una unidad en inventario durante un año, y lead time conocido y constante.

Aunque fundacional, el modelo EOQ hace supuestos restrictivos que limitan su aplicabilidad directa en retail moderno. La demanda raramente es constante, exhibe tendencias, estacionalidad, y variabilidad aleatoria. El supuesto de lead time constante tampoco se sostiene cuando proveedores enfrentan restricciones de capacidad o disrupciones logísticas. El costo de ordenar K , modelado como fijo, en realidad puede variar con descuentos por volumen o tarifas diferenciadas de transporte.

Pese a estas limitaciones, EOQ proporciona intuición valiosa, existe un trade-off fundamental entre los costos de ordenar y mantener, y la cantidad óptima se obtiene balanceando ambos. Esta intuición se preserva en modelos más sofisticados que relajan los supuestos restrictivos.

Reconociendo que la demanda es inherentemente incierta, los modelos estocásticos de inventario introducen conceptos de probabilidad y teoría de colas. El modelo (Q, R) mantiene la estructura de EOQ (ordenar cantidad fija Q cuando inventario cae a nivel R) pero reconoce que la demanda durante lead time es variable aleatoria con distribución conocida.

Silver et al. (1998) extienden el modelo (Q, R) para incorporar costos explícitos de quiebre. Si cada unidad no satisfecha genera costo p , y b es el costo de mantener una unidad durante lead time, el stock de seguridad óptimo se obtiene igualando el costo marginal de incrementar inventario (b) con el beneficio marginal de reducir quiebres ($p \times$ probabilidad de quiebre):

2.2.2. Modelos de revisión periódica.

Una alternativa al modelo (Q, R) de revisión continua (monitorear inventario constantemente y ordenar cuando cae a R) es el modelo de revisión periódica donde el inventario se revisa a intervalos fijos (por ejemplo, cada semana) y se ordena cantidad suficiente para alcanzar el nivel objetivo S . En el modelo (T, S) , cada T períodos se revisa el inventario y se ordena. El nivel objetivo S se calcula considerando la demanda durante el período de revisión más lead time.

Este modelo es más práctico en retail donde revisiones de inventario ocurren naturalmente en frecuencias fijas (conteo semanal, mensual) y ordenar a proveedor tiene calendario predefinido (camión viene martes y viernes). La desventaja es que requiere mayor stock de seguridad que (Q, R) porque debe proteger contra demanda durante un período más largo ($T+L$ vs solo L).

2.2.3. La Demanda como variable aleatoria o variable predecible.

Los modelos estocásticos descritos tratan la demanda futura como variable aleatoria con distribución conocida, pero realización incierta. Esta conceptualización es apropiada cuando la demanda es genuinamente aleatoria (ejemplo, llegada de clientes a tienda sigue proceso de Poisson), pero subestima la estructura predecible de la demanda en muchos contextos de retail.

Como señalan Syntetos et al. (2016), la demanda en retail es más predecible de lo que suponen modelos estocásticos clásicos porque exhibe autocorrelación (ventas de hoy correlacionan con ventas de ayer), estacionalidad (patrones que se repiten cada semana, mes, año), y causalidad (ventas responden a promociones, clima, eventos observables). Tratar esta estructura como ruido aleatorio equivale a descartar información valiosa.

Esta observación motiva la integración de forecasting (predicción de demanda futura basada en datos históricos y variables causales) con optimización de inventarios (cálculo de Q y R óptimos

dado el forecast). En lugar de asumir una demanda con distribución fija, se estima la demanda esperada período por período usando modelos estadísticos o de machine learning, y se ajusta dinámicamente el inventario objetivo.

Fildes et al. (2008) documentan que empresas de retail que integran forecasting sofisticado con gestión de inventarios alcanzan 15-25% de reducción en stock promedio manteniendo o mejorando nivel de servicio, comparado con empresas que usan solo promedios históricos simples. El beneficio proviene de reducir el error de predicción (desviación entre forecast y demanda real) menor error implica menor necesidad de stock de seguridad para compensar la incertidumbre.

2.2.4. Arquitecturas de sistemas de información para inventarios.

La implementación de modelos cuantitativos de inventario en organizaciones reales requiere sistemas de información que capturen datos transaccionales, ejecuten cálculos, y presenten recomendaciones a usuarios de forma accesible. La literatura de sistemas de información identifica tres arquitecturas prevalentes.

Sistemas ERP Integrados, cuyas siglas significan “Enterprise Resource Planning Systems” (SAP, Oracle, Microsoft Dynamics) integran la gestión de inventarios con otros procesos empresariales en una base de datos unificada. La ventaja es la visibilidad completa. Una venta registrada en punto de venta automáticamente decrementa el inventario, dispara una orden si cae por debajo del punto crítico, genera una orden de compra y actualiza las proyecciones financieras. La desventaja es el costo (licencias + consultoría + capacitación típicamente superan \$100k para pymes) y su rigidez ante cambios.

Los Sistemas Especializados de Inventario, software dedicado exclusivamente a gestión de inventarios con capacidades de integración mediante APIs hacia sistemas de punto de venta y contabilidad. Ofrecen un balance entre funcionalidad (forecasting, alertas, reportes especializados) y bajo costo de implementación, siendo una opción popular entre las pymes retail. La limitación es que requieren mantener una sincronización con múltiples sistemas mediante integraciones que pueden fallar.

Existen también soluciones Ad-Hoc Basadas en Planillas. Hojas de cálculo (Excel, Google Sheets) con macros y fórmulas que calculan puntos de reorden, generan reportes, y consolidan datos de múltiples fuentes. Extremadamente flexibles y de costo prácticamente nulo (licencias Office ya poseídas), pero sufren limitaciones de escalabilidad, concurrencia y propensión a errores documentadas en sección 2.1.4.

Laudon y Laudon (2020) argumentan que la elección de arquitectura debe alinearse con el nivel de madurez organizacional (retomando el modelo IMSMM). Empresas en Nivel 1-2 pueden operar eficientemente con soluciones ad-hoc mientras formalizan procesos, empresas en Nivel 3 se benefician de sistemas especializados que automatizan decisiones rutinarias, y empresas en Nivel 4-5 requieren un ERP integrado que soporte optimización sofisticada y simulación de escenarios.

2.2.5. El rol de la analítica predictiva en inventarios modernos.

La disponibilidad creciente de datos granulares (ventas por SKU×local×día, datos climáticos, registros de promociones) y el desarrollo de técnicas estadísticas y de machine learning avanzadas han generado una evolución desde gestión reactiva de inventarios (responder a desabastecimiento) hacia gestión predictiva (anticipar demanda futura y posicionar inventario proactivamente).

Choi et al. (2018) definen analítica predictiva en cadena de suministro como "el uso de datos históricos, algoritmos estadísticos y técnicas de machine learning para identificar la probabilidad de resultados futuros basándose en patrones pasados" (p. 115).

Waller y Fawcett (2013) argumentan que la analítica predictiva transforma la gestión de inventarios de disciplina principalmente operativa (ejecutar procedimientos de reposición) a disciplina estratégica (generar ventaja competitiva mediante mejor anticipación de demanda que competidores). Empresas con capacidades predictivas superiores pueden operar con inventarios 20-30% menores manteniendo el mismo nivel de servicio, liberando capital para inversión en crecimiento o reducción de precios.

Sin embargo, la adopción de analítica predictiva enfrenta barreras. Requiere un expertise técnico (ciencia de datos) escaso en empresas pyme, una robusta infraestructura de datos (repositorios consolidados, procesos ETL) que muchas organizaciones no poseen, y una cultura organizacional que confíe en recomendaciones algorítmicas sobre intuición gerencial. Hofmann y Rutschmann (2018) encuentran que sólo UN 15% de PYMES europeas de retail han adoptado analítica predictiva para inventarios, comparado con el 60% de grandes cadenas, una brecha explicada principalmente por restricción de recursos y expertise.

Las bases teóricas de gestión de inventarios han evolucionado desde modelos determinísticos simples (EOQ) que asumen demanda constante, hacia modelos estocásticos (Q, R, T, S) que reconocen incertidumbre, y más recientemente hacia enfoques de analítica predictiva que explotan datos granulares y técnicas avanzadas para anticipar demanda con mayor precisión. Esta evolución refleja tanto el desarrollo de teoría matemática (investigación de operaciones, estadística) como la disponibilidad de infraestructura tecnológica (sistemas de punto de venta que capturan transacciones detalladas, capacidad computacional para ejecutar modelos complejos).

Los conceptos fundamentales (trade-off entre costo de ordenar y mantener, rol del stock de seguridad, métricas de fill rate y rotación) se preservan a través de esta evolución, pero los métodos para implementarlos se sofistican. ¿cómo pueden las pymes retail, con restricciones de recursos y expertise, acceder a capacidades de analítica predictiva que hasta ahora son dominio de grandes corporaciones?

2.3. Forecasting de demanda, de métodos clásicos a machine learning.

La predicción de demanda constituye el input para las decisiones de inventario discutidas en la sección anterior. Mientras los modelos de optimización (EOQ, Q, R, T, S) determinan cuánto y cuándo ordenar, su efectividad depende de la exactitud con que se estima la demanda futura. Un forecast preciso permite reducir stock de seguridad manteniendo el nivel de servicio, mientras que un forecast erróneo obliga a elegir entre dos males, mantener inventario excesivo como protección contra incertidumbre o aceptar quiebres frecuentes.

La evolución de los métodos de forecasting refleja el progreso desde técnicas estadísticas clásicas que modelan la estructura temporal de series (tendencia, estacionalidad, autocorrelación) hacia enfoques de machine learning que pueden descubrir patrones complejos y no lineales en datos multidimensionales. Esta sección recorre este espectro metodológico, comenzando por los fundamentos de series temporales que dominaron la práctica de forecasting durante el siglo XX, continuando con el surgimiento de machine learning supervisado en la década de 2000, y culminando con la adopción reciente de deep learning para problemas de predicción en retail.

2.3.1. Fundamentos de series temporales.

Una serie temporal es una secuencia de observaciones ordenadas cronológicamente, donde cada observación está indexada por el momento en que ocurrió. El objetivo del forecasting es, dado el histórico temporal, predecir valores futuros.

La identificación y modelado de cada componente permite generar forecasts mediante extrapolación, si la tendencia ha sido creciente linealmente en el pasado, se proyecta que continuará creciendo linealmente en el futuro. Si existe estacionalidad anual, se espera que los picos y valles del año próximo ocurran en los mismos meses que este año. Esta extrapolación es válida bajo el supuesto de una estabilidad estructural, los patrones del pasado se preservan en el futuro.

Un concepto fundamental en series temporales es la autocorrelación, la correlación de la serie consigo misma desplazada en el tiempo. Conocida como lag o retardo.

2.3.2. De estadística clásica a machine learning.

El surgimiento del machine learning supervisado durante las décadas de 1990 y 2000 introdujo una perspectiva diferente sobre forecasting. En lugar de modelar explícitamente la estructura temporal, el machine learning reformula el problema como regresión genérica, dado un conjunto de features X (que pueden incluir lags de ventas, día de la semana, temperatura, indicador de oferta), predecir target y (ventas futuras).

Esta reformulación tiene una implicación profunda, las features temporales (lags, rolling means, indicadores de estacionalidad) se tratan como features ordinarias sin distinción de su origen temporal. El modelo no "sabe" qué unidades_lag_1 representa ventas de la semana pasada, solo sabe que es una variable numérica que puede usarse para predecir el target. Esto implica que el feature engineering, la construcción manual de features informativas, se vuelve responsabilidad del analista y no del modelo.

La ventaja de esta perspectiva es la flexibilidad que otorga, el mismo algoritmo de machine learning puede usarse para forecasting, clasificación, o cualquier problema de predicción supervisada. Los algoritmos de machine learning están diseñados para descubrir automáticamente relaciones no lineales e interacciones entre features sin necesidad de especificarlas manualmente.

Gradient Boosting (Friedman 2001) es una técnica de ensemble que construye una predicción mediante combinación de múltiples modelos simples (típicamente árboles de decisión poco profundos) entrenados secuencialmente. Cada modelo nuevo se entrena para corregir los errores residuales del ensemble previo.

Para forecasting, Gradient Boosting opera sobre features ensambladas manualmente, típicamente lags de diferentes períodos, rolling means de diferentes ventanas, indicadores de estacionalidad codificados numéricamente, y variables exógenas.

La fortaleza principal del Gradient Boosting es su capacidad de capturar no linealidades y aprender interacciones automáticamente. Un árbol puede dividir primero por "tiene_oferta" y luego por "temperatura" solo en la rama donde hay oferta, capturando que el efecto de temperatura sobre ventas es mayor cuando hay promoción. No requiere especificar esta interacción manualmente mediante otros términos.

La limitación es que sigue requiriendo feature engineering manual explícito. Los lags, rolling means, encodings cíclicos de mes/día deben crearse antes de entrenar el modelo. Gradient Boosting no extrae automáticamente estas features de datos crudos. Adicionalmente, aunque puede capturar no linealidades, puede ser menos eficiente que métodos paramétricos para capturar no linealidades globales suaves.

2.3.3. Deep learning, aprendizaje de representaciones jerárquicas.

Las redes neuronales artificiales representan un cambio paradigmático adicional. A diferencia de Gradient Boosting que construye predicción mediante ensemble de modelos simples, las redes neuronales profundas (deep neural networks) son modelos únicos con arquitectura en capas donde cada capa transforma la representación de los datos.

Una red neuronal conectada y completa con X capas ocultas puede expresarse como composición de funciones, donde cada función corresponde a una capa que aplica transformación lineal seguida de activación no lineal.

Hornik et al. (1989) demostraron que incluso una red con una sola capa oculta de suficientes neuronas puede aproximar cualquier función continua (teorema de aproximación universal). Sin embargo, las redes profundas (múltiples capas) son más eficientes en la práctica porque aprenden representaciones jerárquicas, las primeras capas detectan patrones simples, las capas intermedias los combinan en conceptos de nivel medio, y las capas finales integran todo en la predicción.

Como explican Goodfellow et al. (2016), "la profundidad permite a la red componer múltiples transformaciones simples en una función compleja." (p.167)

2.3.4 Regularizaciones.

Con arquitecturas profundas que pueden tener cientos de miles de parámetros, el riesgo de overfitting (memorizar ruido del training set en lugar de aprender patrones generalizables) es mayor. Las técnicas de regularización previenen este problema.

Dropout (Srivastava et al. 2014) desactiva aleatoriamente una fracción p de neuronas durante cada iteración de entrenamiento, forzando a la red a no depender excesivamente de neuronas específicas. Esto crea efecto de ensemble donde implícitamente se entrenan múltiples subredes.

Batch Normalization (Ioffe & Szegedy 2015) normaliza las activaciones de cada capa para tener media cercana a cero y varianza cercana a uno, estabilizando el entrenamiento y actuando como regularizador.

L2 Regularization (weight decay) añade término de penalización a la función de pérdida proporcional a la suma de cuadrados de los pesos, favoreciendo soluciones donde la predicción depende de múltiples features con ponderaciones moderadas en lugar de pocas features con pesos extremos.

2.3.5 Aplicabilidad a forecasting de inventarios.

Para problemas de forecasting en retail donde se dispone de features formadas manualmente (lags, rolling statistics, encoding de estacionalidad, indicadores de ofertas) y el objetivo es capturar interacciones complejas entre ellas, las redes neuronales profundas ofrecen una capacidad expresiva superior a métodos lineales o árboles individuales.

La elección entre Gradient Boosting y redes neuronales profundas depende de los trade-offs, Gradient Boosting requiere menos trabajo sobre hiper parámetros, es más interpretable (importancia de features clara), y se entrena más rápido. Por otro lado, las redes neuronales requieren más datos, más tiempo de entrenamiento, y expertise en selección de arquitectura, pero pueden aprender representaciones más sofisticadas cuando la complejidad del problema lo justifica.

La pregunta que surge naturalmente es, dado un problema específico de forecasting de inventarios en retail con características particulares (tamaño de dataset, número de features, tipo de patrones esperados) ¿cómo seleccionar el método apropiado y diseñar las features que alimentarán este método? Esta cuestión motiva la siguiente sección sobre feature engineering.

2.4. Feature engineering.

El desempeño de modelos de machine learning en problemas de predicción depende menos de la sofisticación del algoritmo y más de la calidad de las variables predictivas que se le proporcionan. Domingos (2012) argumenta que dedicar esfuerzo a mejorar las features genera retornos mayores que invertir tiempo en ajustar hiper parámetros o probar algoritmos alternativos. Esta observación es particularmente relevante en forecasting donde la estructura temporal de los datos no es naturalmente accesible para algoritmos de regresión estándar, los lags de ventas pasadas, promedios móviles, y patrones estacionales deben extraerse explícitamente mediante transformaciones apropiadas.

El feature engineering es el proceso de transformar datos crudos en representaciones que facilitan el aprendizaje de patrones relevantes por parte del modelo. En contexto de gestión de inventarios, esto implica convertir tablas transaccionales (fecha, SKU, local, unidades vendidas, precio) en matrices donde cada fila representa una instancia de predicción y cada columna una variable que potencialmente influye en la demanda futura. Esta sección revisa las categorías principales de features aplicables a forecasting en retail, organizadas según el tipo de información que capturan.

2.4.1. Features temporales como memoria de un pasado reciente.

Las features temporales permiten que modelos de machine learning, que no tienen noción intrínseca de tiempo, accedan a información histórica relevante. La transformación más directa es el lag, crear variable que contenga el valor de la serie temporal desplazado en el tiempo.

Brownlee (2018) recomienda experimentar con múltiples órdenes de lags simultáneamente, permitiendo que el modelo determine cuáles son más informativos. En retail con estacionalidad semanal, el lag de orden 1 captura dependencia de corto plazo (esta semana similar a la anterior), mientras que los lags de orden 4 o 5 capturan patrones mensuales o anuales respectivamente.

Complementando los lags puntuales, las rolling statistics calculan estadísticos móviles sobre ventanas temporales. Un rolling mean de 4 semanas calcula el promedio de ventas de las últimas 4 observaciones, proporcionando una visión suavizada que filtra ruido de corto plazo. El rolling standard deviation captura volatilidad reciente, señalando al modelo si el producto está en período de demanda estable o altamente variable.

Features derivadas de segundo orden como el momentum (tasa de cambio porcentual entre períodos consecutivos) y la aceleración (segunda derivada, cambio del cambio) permiten detectar tendencias emergentes. Un producto con momentum positivo creciente está en fase de aceleración de ventas, información valiosa para ajustar inventario proactivamente.

2.4.2. Features contextuales, ciclos y estacionalidad.

Los patrones estacionales en retail exhiben estructura cíclica, las ventas de enero se asemejan a las de enero del año anterior más que a las de julio del mismo año. Sin embargo, codificar mes como variable ordinal (enero=1, febrero=2, ..., diciembre=12) introduce distorsión, el modelo interpreta que diciembre (12) está "alejado" de enero (1) cuando en realidad están adyacentes en el ciclo anual. El encoding cíclico resuelve este problema mediante transformaciones trigonométricas que preservan la topología circular del tiempo.

2.4.3. Features comerciales, promociones y su dinámica.

Las ofertas y descuentos generan efectos complejos sobre la demanda que van más allá de la presencia binaria de promoción. La magnitud del descuento importa, un descuento del 10% tiene impacto menor que uno del 40%. La duración de la oferta también influye, promociones extendidas durante múltiples semanas experimentan fatiga donde el incremento de ventas decae progresivamente.

Ehrenberg et al. (1994) documentan que el efecto de promociones alcanza un pico en las primeras semanas de implementación y decrece sistemáticamente después, incluso si los términos permanecen constantes. Este fenómeno, denominado fatiga promocional, sugiere que las features de duración (número de semanas consecutivas que el producto ha estado en oferta) son informativas.

Las transiciones promocionales (entrar o salir de oferta) generan saltos abruptos de demanda. Un producto que entra en oferta esta semana tras varias semanas sin promoción típicamente experimenta pico de ventas mayor que un producto que ya venía en oferta. Features binarias que señalan estas transiciones permiten al modelo ajustar predicciones en momentos de cambio de estado.

2.4.4. Features de categorización, segmentación de productos.

En retail con catálogos amplios, los productos se agrupan naturalmente en categorías (familias, marcas, formatos) que comparten comportamiento de demanda. Las features de categorización permiten al modelo aprovechar información agregada de categorías para mejorar predicciones de productos individuales.

El label encoding asigna entero único a cada categoría (Quilmes=1, Heineken=2, Corona=3), permitiendo que algoritmos basados en árboles segmenten datos por categoría. Sin embargo, esta codificación introduce un ordenamiento artificial que puede confundir a modelos lineales o redes neuronales.

El target encoding, formalizado por Micci-Barreca (2001), reemplaza cada categoría por estadísticos calculados sobre el target en esa categoría. Para forecasting, se puede calcular la media histórica de ventas de productos en cada familia, la desviación estándar (volatilidad), y el conteo de observaciones (confianza). Un producto nuevo de la familia "IPAs Premium" hereda estadísticas de su familia, permitiendo de esta manera obtener predicciones razonables incluso sin historia propia.

2.4.5. Features de interacción, efectos sinérgicos.

Muchos fenómenos de demanda involucran interacciones donde el efecto de una variable depende del valor de otra. El impacto de temperatura sobre las ventas de cerveza es mayor cuando hay

oferta activa, generando sinergia entre clima y promoción. Las features de interacción capturan estos efectos mediante productos o combinaciones de variables base.

Features de interacción entre clima y calendario (lluvia en fin de semana, temperatura alta en feriado) capturan que ciertos efectos ambientales se amplifican en momentos específicos. Features de precio relativo (precio actual dividido por promedio histórico) señalan si el producto está caro o barato comparado con su línea histórica, información relevante independientemente del precio absoluto.

2.4.6. Validación y limpieza de features.

La construcción de features inevitablemente genera artefactos que deben corregirse antes de entrenar modelos. Las features basadas en lags generan valores faltantes en las primeras observaciones donde no existe historia suficiente. La estrategia común es rellenar con cero, interpretándolo como ausencia de información, aunque alternativas como forward fill o interpolación existen según contexto.

El cálculo de ratios o transformaciones logarítmicas puede producir infinitos cuando los denominadores son cero o valores indefinidos cuando argumentos son negativos. La detección y reemplazo de estos valores atípicos previene los errores numéricos durante el entrenamiento.

El feature engineering transforma estos datos transaccionales crudos en representaciones estructuradas que facilitan el descubrimiento de patrones por algoritmos de machine learning. Las categorías de features revisadas (temporales, contextuales, comerciales, categorización, interacción) no son exhaustivas, pero representan el kit de herramientas básico para problemas de forecasting en retail. La calidad de estas transformaciones, más que la elección del algoritmo en sí, determinan el límite de precisión alcanzable.

Como sintetiza Domingos (2012), "al final del día, algunos machine learning projects triunfan y otros fallan. Lo que distingue a los exitosos es la calidad de las features más que la sofisticación del modelo. Applied machine learning es básicamente feature engineering" (p. 80). Esta perspectiva motiva la inversión sustancial de esfuerzo en diseño, validación y refinamiento iterativo de features antes de experimentar con arquitecturas complejas de modelos.

2.5. Evaluación de modelos de forecasting.

La evaluación rigurosa de modelos de forecasting es fundamental tanto para seleccionar entre alternativas metodológicas durante el desarrollo como para cuantificar la precisión alcanzada en producción. A diferencia de problemas de clasificación donde una predicción es correcta o incorrecta, en forecasting toda predicción contiene cierto grado de error, el objetivo no es error cero (inalcanzable en presencia de variabilidad aleatoria genuina) sino error sistemáticamente menor que alternativas disponibles. Esta sección articula los principios de evaluación que permiten comparar métodos estadísticos clásicos, algoritmos de machine learning tradicional, y arquitecturas de deep learning sobre bases equivalentes.

2.5.1. Métricas de error punto a punto.

Las métricas de error cuantifican la discrepancia entre predicciones y valores observados, agregando errores individuales en estadístico único que facilita comparación entre modelos. MAE (mean absolute error) RMSE (root mean squared error) y MAPE (mean absolute percentage error) son ejemplos de este tipo de métrica.

Complementando métricas de error punto a punto, las métricas de ajuste global cuantifican qué proporción de la variabilidad de la serie temporal es explicada por el modelo.

El Coeficiente de Determinación (R^2) compara la varianza de los residuos con la varianza total de la serie. Valores cercanos a 1 indican que el modelo captura casi toda la variabilidad sistemática, dejando solo ruido aleatorio sin explicar. Valores cercanos a 0 o negativos señalan que el modelo no supera la predicción trivial de usar la media constante. En forecasting de inventarios, R^2 superiores a 0.80 generalmente se consideran buenos, aunque este umbral varía según volatilidad intrínseca del producto (productos con demanda muy estable alcanzan R^2 mayores que productos con demanda errática).

El sesgo direccional (bias) mide si el modelo tiende sistemáticamente a sobreestimar o subestimar. Un bias positivo indica sobrepredicción sistemática (recomendaciones de inventario excesivas), un bias negativo subpredicción (quiebres frecuentes). Syntetos et al. (2016) señalan que, en contextos inflacionarios, un bias conservador (negativo) puede ser financieramente óptimo porque minimizar el capital inmovilizado compensa ventas perdidas ocasionales.

2.5.2. Validación temporal.

En problemas de series temporales, la validación cruzada estándar que mezcla observaciones aleatoriamente viola el principio de causalidad, el futuro no puede usarse para predecir el pasado.

Hyndman y Athanasopoulos (2018) formalizan que "la evaluación honesta de modelos de forecasting requiere división estrictamente temporal donde los datos de test son cronológicamente posteriores a los datos de entrenamiento, simulando la situación real de predecir hacia adelante" (p. 38). Esta práctica previene data leakage temporal donde el modelo inadvertidamente "ve el futuro" durante entrenamiento, generando métricas artificialmente optimistas que colapsan en producción.

La validación cruzada walk-forward extiende este principio para aprovechar mejor los datos limitados. En lugar de una sola división fija, se crean múltiples ventanas de entrenamiento-validación desplazadas en el tiempo. Por ejemplo, entrenar con semanas 1-30 y validar en semana 31, luego entrenar con semanas 1-31 y validar en 32, repitiendo hasta agotar datos. El error promedio sobre todas las validaciones proporciona una estimación más robusta del desempeño esperado.

2.5.3. El gap entre la validación offline y el rendimiento online.

Una observación recurrente en la literatura de machine learning aplicado es que los modelos que alcanzan alta precisión en conjuntos de test retrospectivos sistemáticamente exhiben degradación al desplegarse en producción. Este gap offline-online surge de múltiples fuentes que no están presentes durante la validación controlada.

Breck et al. (2017) identifican el distribution shift como causa principal, cambios sutiles en la distribución de features de entrada entre entrenamiento y producción generan predicciones menos precisas. En forecasting, esto se manifiesta cuando los patrones de consumo evolucionan (tendencias emergentes que no estaban declaradas entre los datos históricos), cuando eventos no representados en training ocurren durante la producción (promociones de nueva mecánica, interrupciones logísticas), o cuando la precisión de las features exógenas difiere entre el entrenamiento y la producción (por ejemplo, usar clima real durante validación vs pronósticos climáticos durante predicción prospectiva).

La varianza irreducible representa otra fuente de gap. Los conjuntos de test que abarcan múltiples períodos permiten que efectos aleatorios positivos y negativos se promedien parcialmente, mientras que predicciones para un único período futuro pueden coincidir con eventos estocásticos extremos (partido inesperado, clima atípico) que inflan error sin reflejar deficiencias del modelo.

Makridakis et al. (2020) en su análisis de la competencia M5 de forecasting reportan que "modelos con R^2 de 0.95 en test offline rutinariamente caen a 0.75-0.85 en producción. Este gap no es anomalía sino regularidad que debe anticiparse al diseñar sistemas de forecasting para uso operativo" (p. 1358). Esta observación subraya la importancia del pilotaje prospectivo, validar el modelo generando predicciones reales para períodos futuros desconocidos, no sólo reconstruir el pasado conocido.

2.5.4. Consideraciones específicas para redes neuronales.

La evaluación de redes neuronales profundas introduce consideraciones adicionales derivadas de su naturaleza estocástica y capacidad expresiva elevada. A diferencia de modelos con solución analítica cerrada (regresión lineal) o determinística (árboles de decisión con semilla fija), las redes neuronales entrenadas múltiples veces con mismos datos producen modelos ligeramente diferentes debido a la inicialización aleatoria de pesos y mini-batch sampling estocástico.

Goodfellow et al. (2016) recomiendan entrenar múltiples instancias del modelo con diferentes semillas aleatorias y reportar media y desviación estándar de métricas sobre estas instancias, proporcionando una estimación más robusta del desempeño esperado. En la práctica, las diferencias entre instancias suelen ser pequeñas (variación del 1-3% en MAE) cuando el modelo está bien regularizado, pero pueden ser sustanciales en arquitecturas propensas a quedar atrapadas en mínimos subóptimos.

La capacidad de overfitting de redes profundas con cientos de miles de parámetros requiere un monitoreo cuidadoso de curvas de aprendizaje. Graficar la evolución del loss (pérdida) en entrenamiento y su validación a lo largo de las epochs (etapas de entrenamiento) permite detectar overfitting cuando la pérdida en training continúa bajando pero la de validación se estanca o aumenta. Técnicas de early stopping (detener entrenamiento cuando validación deja de mejorar durante n epochs consecutivos) previenen memorización excesiva, restaurando los pesos del epoch con mejor desempeño en validación.

La interpretabilidad reducida de redes neuronales comparada con árboles de decisión o regresión lineal es compensada parcialmente por técnicas que descomponen predicciones individuales en contribuciones de cada feature. Lundberg y Lee (2017) demuestran que SHAP proporciona explicaciones consistentes y localmente precisas incluso para modelos de caja negra, facilitando diagnóstico de por qué el modelo predijo valor específico para observación particular.

La evaluación efectiva de modelos de forecasting requiere baterías de métricas que capturen dimensiones complementarias del desempeño (error típico vía MAE, error extremo vía RMSE, ajuste global vía R^2 , sesgo sistemático vía bias), procedimientos de validación que respeten causalidad temporal, y comparación rigurosa con baselines simples que justifiquen la complejidad introducida.

La pregunta que surge naturalmente es cómo integrar estos componentes en solución coherente para este contexto específico. Esta integración conceptual motiva la síntesis final del marco teórico.

2.6. Síntesis.

El recorrido teórico presentado en este capítulo ha articulado cinco pilares conceptuales que fundamentan el diseño de soluciones predictivas para gestión de inventarios en retail. Primero, se estableció que las pymes enfrentan tensión estructural entre la necesidad de profesionalizar prácticas de inventario y barreras para adoptar sistemas integrados costosos, operando frecuentemente en niveles bajos de madurez según modelos como IMSMM. Segundo, se revisaron los modelos cuantitativos de inventario que definen el trade-off entre costos de ordenar y mantener, identificando que la precisión del forecasting de demanda es el input crítico que determina la efectividad de cualquier política de reposición. Tercero, se trazó la evolución de métodos de forecasting desde estadística clásica hacia machine learning, evidenciando que la disponibilidad de datos granulares y capacidad computacional han habilitado técnicas que aprenden patrones complejos automáticamente. Cuarto, se estableció que la calidad de las features (lags, encoding cíclico, modelado de promociones, segmentación de productos) determinan el límite de precisión alcanzable independientemente del algoritmo. Finalmente, se definieron los criterios de evaluación (MAE, RMSE, R^2 , validación temporal, comparación con baselines) que permiten cuantificar si la complejidad técnica introducida genera valor medible.

2.6.1. Integración conceptual, del problema a la solución.

La integración de estos componentes teóricos configura un framework para abordar el problema de gestión de inventarios mediante analítica predictiva. El punto de partida es reconocer que la demanda futura, aunque incierta, no es aleatoria, exhibe autocorrelación (ventas de hoy correlacionan con ayer), estacionalidad (patrones recurrentes semanales, mensuales, anuales), y causalidad (responde a promociones, clima, eventos observables). Descartar esta estructura tratando la demanda como ruido blanco equivale a desperdiciar información valiosa.

Entre las alternativas de machine learning, las redes neuronales profundas ofrecen una ventaja conceptual, su arquitectura en capas permite aprender representaciones jerárquicas donde las capas iniciales detectan patrones simples, capas intermedias los combinan en conceptos de nivel medio, y capas finales integran todo en la predicción. Esta capacidad de abstracción progresiva es particularmente valiosa en problemas con features mixtas (numéricas continuas como temperatura, categóricas como familia de producto, binarias como indicador de oferta) donde las primeras capas normalizan e interrelacionan estas features heterogéneas antes de que capas posteriores extraigan patrones predictivos.

2.6.2. Justificación de deep learning para el contexto específico.

La decisión de adoptar redes neuronales profundas sobre alternativas más simples (regresión lineal, árboles individuales) o intermedias (Gradient Boosting) no es arbitraria, responde a características específicas del problema de forecasting de inventarios en retail de bebidas.

Volumen de datos suficiente. La literatura (Bojer & Meldgaard 2021, Lim et al. 2021) establece que deep learning requiere típicamente más de 50,000 observaciones para superar consistentemente a métodos más simples. En retail con múltiples SKUs, múltiples locales, y múltiples períodos temporales, este umbral es alcanzable, un catálogo de 400 productos en 15 locales registrando ventas durante 30 semanas genera 180,000 observaciones.

Múltiples features con interacciones complejas. El comportamiento de demanda en bebidas responde a interacciones de segundo y tercer orden, temperatura afecta ventas solo por encima de cierto umbral, el efecto es mayor en fines de semana, y se amplifica cuando hay oferta activa.

Especificar manualmente todas estas interacciones en modelo lineal o incluso en Gradient Boosting es impráctico, las redes neuronales pueden descubrirlas mediante composición de transformaciones no lineales en capas sucesivas.

Heterogeneidad de productos. Un catálogo típico incluye productos con rotación extremadamente alta (marcas líderes que venden cientos de unidades semanales), rotación media (marcas secundarias con decenas de unidades), y rotación baja (productos nicho con ventas esporádicas). Un modelo único debe adaptar su comportamiento según este contexto, las redes neuronales con capacidad suficiente pueden aprender diferentes submodelos implícitos para diferentes regímenes de datos.

Necesidad de regularización agresiva. Los datos de retail contienen ruido sustancial, errores de registro en punto de venta, promociones aplicadas incorrectamente, conteos de inventario físico imprecisos. Las técnicas de regularización de redes neuronales (Dropout, Batch Normalization, L2) están específicamente diseñadas para prevenir la memorización de este ruido, forzando al modelo a aprender patrones robustos que generalizan.

2.6.3. El rol del feature engineering en deep learning.

Existe concepción errónea de que deep learning elimina la necesidad de feature engineering porque las redes pueden aprender features automáticamente de datos crudos. Esta afirmación es parcialmente cierta en dominios como visión computacional donde las redes convolucionales aprenden detectores de bordes, texturas, y objetos directamente de píxeles, o procesamiento de lenguaje natural donde los embeddings aprenden representaciones semánticas de palabras.

Sin embargo, en forecasting con datos tabulares heterogéneos, el feature engineering manual sigue siendo valioso por varias razones. Primero, incorpora conocimiento del dominio que acelerará dramáticamente el aprendizaje, una red neuronal podría en principio descubrir que el promedio de ventas de las últimas 4 semanas es predictivo, pero proporcionarle explícitamente esta feature (rolling mean de 4 períodos) reduce el espacio de hipótesis que debe explorar. Segundo, ciertas transformaciones (encoding cíclico de estacionalidad, target encoding de categorías) codifican estructura que la red no descubriría eficientemente por sí sola. Tercero, features de interacción explícitas (temperatura $\times$ oferta) guían a la red hacia patrones relevantes, mejorando convergencia.

2.6.4. Reflexión personal sobre la brecha teoría-práctica.

La revisión teórica presentada en este capítulo, aunque fundamentada en literatura académica rigurosa, simplifica la complejidad y ambigüedad del desarrollo de sistemas predictivos en contextos empresariales reales. Los papers citados reportan resultados en datasets públicos curados (M5 de Walmart, competencias Kaggle) donde los datos están limpios, las features están documentadas, y las métricas de evaluación están estandarizadas. La realidad de implementar forecasting en una pyme argentina de retail es radicalmente diferente.

Los datos no aparecen en formato CSV bien estructurado, están fragmentados en múltiples archivos Excel con formatos inconsistentes, columnas renombradas arbitrariamente entre semanas, fechas en formatos ambiguos y hay discrepancias entre lo que el sistema registra y lo que sucedió físicamente (ventas de SKU discontinuado que siguen apareciendo porque el código fue reutilizado para producto nuevo).

La selección de arquitectura de red neuronal (cuántas capas, cuántas neuronas por capa, qué funciones de activación, qué técnicas de regularización) no emerge de un teorema matemático sino

de experimentación iterativa. Se entrenan múltiples configuraciones, se comparan métricas en validación, se inspeccionan curvas de aprendizaje buscando señales de underfitting u overfitting, y se ajustan hiper parámetros basándose en intuición desarrollada mediante prueba y error. Este proceso, aunque informado por principios teóricos, es artesanal en naturaleza.

La adopción de recomendaciones algorítmicas por usuarios finales (gerentes de compras, encargados de local) involucra dinámicas psicológicas y organizacionales que exceden consideración técnica. Un modelo con MAE de, por ejemplo, 15 puede rechazarse si los usuarios no confían en sus predicciones o no entienden su lógica, mientras que un modelo con MAE de 25 puede adoptarse si se presenta de forma transparente y permite override manual. La gestión del cambio organizacional, la capacitación de usuarios, y el diseño de interfaces intuitivas son determinantes del éxito que la literatura de machine learning raramente aborda.

Esta reflexión no invalida el valor del marco teórico presentado, por el contrario, establece las bases conceptuales sin las cuales el desarrollo práctico carecería de fundamento. Pero subraya que el paso de la teoría al sistema funcional en una empresa real implica la navegación de ambigüedad, una toma de decisiones con información incompleta, y adaptación pragmática a restricciones que ningún paper académico puede anticipar completamente.

2.6.5. Transición hacia la metodología.

El marco teórico ha establecido qué se conoce del problema (modelos de inventario, métodos de forecasting, feature engineering, evaluación), por qué ciertas elecciones metodológicas son apropiadas (deep learning para datos abundantes con patrones complejos), y bajo qué criterios se evaluará el éxito (métricas de error, comparación con baselines, validación temporal). El siguiente paso consiste en operacionalizar estos conceptos en metodología concreta, especificar fuentes de datos, describir proceso ETL, definir conjunto de features a implementar, detallar arquitectura de red neuronal, y diseñar protocolo de evaluación que incluya tanto test retrospectivo como piloto prospectivo.

La transición de teoría abstracta a implementación específica requiere decisiones que inevitablemente comprometen generalidad por aplicabilidad. Mientras la teoría discute "lags de diferentes órdenes", la práctica debe decidir exactamente cuáles lags incluir (1, 2, 4, 52 semanas). Mientras la teoría menciona "arquitectura profunda apropiada", la práctica debe especificar configuración exacta (5 capas con 256, 128, 64, 32, 16 neuronas). Estas decisiones se fundamentan en el marco teórico pero requieren juicio informado por el contexto específico del problema a resolver.

Este capítulo ha proporcionado fundamentación teórica comprehensiva que justifica el desarrollo de un sistema de gestión predictiva de inventarios basado en redes neuronales profundas para una pyme de retail de bebidas. Se han articulado los desafíos del problema (tensión entre necesidad de profesionalización y barreras de adopción), los componentes conceptuales de la solución (forecasting con machine learning, feature engineering específico del dominio, evaluación rigurosa), y las consideraciones prácticas que median entre teoría y aplicación real. El camino está trazado conceptualmente, resta transitarlo.

Objetivos.

3.1. Objetivo general.

Desarrollar e implementar un sistema predictivo de inventarios basado en redes neuronales profundas que reduzca el error de predicción en al menos 25% respecto a métodos tradicionales, validado mediante un piloto en Beer Market.

3.2. Objetivos específicos.

1. Diseñar y consolidar un dataset integrado que incorpore datos históricos de ventas, variables climáticas, calendario comercial y factores de estacionalidad, implementando un proceso ETL documentado y validado según estándares de calidad de datos.
2. Implementar un sistema de ingeniería de features que capture patrones temporales (lags, momentum, tendencias), efectos de estacionalidad (ciclos mensuales y semanales), dinámicas comerciales (ofertas, descuentos, fatiga promocional) y características de segmentación de productos.
3. Validar el modelo mediante conjuntos de test retrospectivo y piloto prospectivo, midiendo precisión global y distribución del error por segmentos de volumen y familias de productos.
4. Comparar el desempeño del sistema con métodos baseline tradicionales para cuantificar el valor agregado de la solución propuesta.

Desarrollo.

4. Fase 1: Recolección y limpieza de datos.

Paso fundamental inherente de cualquier procedimiento. Se obtiene la información de un sistema transaccional que genera distintos tipos de reportes, primero se determina que información se requiere como parte de la tabla de hechos que vamos a estar alimentando. Luego, se genera un histórico de datos a partir de bloques segmentados estratégicamente con el fin de consolidar un centro de datos que funcionará para alimentar de contexto a nuestro sistema predictivo.

Luego empezamos a agregar variables que afectan directamente a la venta, se definen cuáles son los principales generadores de ruido dentro del sistema y buscamos la forma de poner en valor toda esta información. Aquí conviene realizar un paso intermedio, solicitar una entrevista con los socios gerentes de la compañía (particularmente COO) e intercambiar sobre cuales cree que son los principales factores que considerar.

También es crucial para este modelo de datos contar con información de carácter descriptiva de cada uno de los almacenes, cómo se comportan (Si son un depósito o un punto de venta como lo más básico) Y luego empezar a vincular distintos factores socioeconómicos a cada local (Ejemplo sueldo medio del trabajador de la zona, valor del m2, densidad poblacional, tamaño de cada local así como el espacio funcional que posee, cantidad de empleados, horarios de trabajo, cercanía a monumentos/estadios/puntos de interés) Lo mismo en este aspecto, vemos conveniente dentro de un marco de entrevistas definir con la persona responsable de locales cuales son las principales variables que pueden afectar en el volumen operado de un local.

Una vez definidas todas esas variables, es necesario determinar dónde almacenar este centro de datos consolidado. En esta primera instancia, y con el objetivo de optimizar recursos en contexto de restricciones presupuestarias típicas de una pyme, el dataset se consolida en formato CSV UTF-8 almacenado en un disco duro local. Esta decisión, aunque subóptima desde perspectivas de arquitectura empresarial de datos (donde se recomendarían bases de datos relacionales con control de versiones, acceso concurrente, y backups automatizados), resulta eficiente en el aprovechamiento de recursos disponibles. El almacenamiento en un disco local elimina costos de infraestructura, y el formato CSV permite su procesamiento mediante bibliotecas estándar de Python sin necesidad de licencias de software especializado. Esta pragmática decisión arquitectónica prioriza viabilidad de implementación inmediata sobre escalabilidad futura, reconociendo que sistemas más robustos pueden adoptarse posteriormente si el piloto demuestra valor.

Basándonos en los principios de ETL (Extract, Transform, Load), la optimalidad de un formato de datos se mide por su eficiencia en las tres fases. Facilidad de extracción, flexibilidad para la transformación y simplicidad para la carga. El formato CSV (Valores Separados por Comas) codificado en UTF-8 se destaca significativamente sobre otros formatos (como JSON, XML o formatos binarios propietarios) en las fases iniciales de un pipeline ETL, especialmente en la Extracción y Transformación temprana.

“El formato CSV es un estándar de facto para el intercambio de datos tabulares. Su estructura simple, basada en texto plano, lo hace inmediatamente legible tanto por humanos como por máquinas sin necesidad de librerías complejas. Esto simplifica enormemente la fase de

extracción.” Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley. p. 310.

Kimball y Ross, autoridades en el modelado dimensional para Data Warehouses, enfatizan la importancia de la simplicidad en las fuentes de datos. Aunque no hablan exclusivamente de CSV, su filosofía se alinea con el uso de formatos simples y accesibles

"Los archivos planos son, con mucho, la fuente más común para el proceso de ETL. Son universales, son independientes de la plataforma y pueden ser generados por virtualmente cualquier sistema origen" (Kimball & Ross, 2013, p. 310).

Esta independencia de la plataforma es fundamental. Un archivo CSV puede ser generado por una hoja de cálculo, una base de datos, un sistema legacy o una aplicación moderna, y puede ser leído por cualquier lenguaje de programación (Python, R, Java) con librerías mínimas, a diferencia de formatos binarios propietarios como los de Excel (.xlsx) que requieren herramientas específicas.

La estructura del CSV permite un procesamiento secuencial y eficiente, ideal para procesamiento de grandes volúmenes de datos. Comparado con formatos más verbosos como XML o JSON, que repiten nombres de etiquetas y utilizan caracteres estructurales adicionales (, { }, < > , " "), el CSV tiene una sobrecarga mínima. Esto se traduce en velocidad de lectura, menor tiempo de parsing, ya que el separador (generalmente una coma) es un delimitador simple. Menor uso de memoria, ya que no es necesario cargar una estructura de árbol compleja en memoria (DOM para XML) para comenzar a procesar los datos. Se puede leer línea por línea. Flexibilidad en el esquema, mientras que los formatos con esquema estricto (como Parquet o Avro) son excelentes para el almacenamiento final, el CSV es más tolerante durante la ingesta inicial, permitiendo una inferencia o aplicación de esquemas durante la fase de transformación.

La inclusión de UTF-8 no es un detalle menor, sino un factor que hace óptimo al CSV. UTF-8 es un estándar de codificación de caracteres que puede representar cualquier carácter en el estándar Unicode, cubriendo así prácticamente todos los idiomas y caracteres especiales del mundo.

La W3C, consorcio internacional que desarrolla estándares para la web, establece la importancia de Unicode (y por extensión, UTF-8, su codificación más común) como base para el intercambio de texto.

"Unicode proporciona un espacio de código único y unificado para prácticamente todos los caracteres de uso moderno e histórico en el mundo. [...] El uso de Unicode es esencial para el intercambio de texto a nivel internacional" (W3C, 2014, p. 3).

Sin UTF-8, un archivo CSV con datos en español (con tildes, 'ñ'), francés (cedillas) o japonés (kanji) sería ilegible o se corrompería al moverse entre sistemas con diferentes codificaciones (como Latin-1 o Windows-1252). UTF-8 resuelve el problema de la internacionalización, garantizando que los datos se extraigan y lean correctamente desde el origen, lo cual es un requisito previo para cualquier transformación confiable.

Una vez establecido el formato CSV-UTF8 como el cimiento técnico óptimo para nuestro proceso ETL, garantizando una extracción eficiente y libre de errores de codificación, se debe abordar la procedencia y calidad de los datos que alimentarán este flujo. La solidez del formato de almacenamiento sería irrelevante si los datos que contiene son inexactos o inconsistentes. Por ello, el siguiente eslabón crítico consiste en detallar la metodología para la recopilación de cada fuente

de información, explicitando el marco de gobierno de datos que implementaremos para validar su calidad, verificar la confiabilidad de sus orígenes y, fundamentalmente, asegurar que sean fuentes atemporales o garantes de persistir en el tiempo, construyendo así una base de datos resiliente y confiable para nuestro bot.

Tabla 1.

Tabla de validación de calidad de datos.

mención de calidad de datos	Descripción
Oportunidad	Disponibilidad de los datos dentro de un plazo determinado.
Lo completo	La cantidad de datos utilizables o completos, representativos de una muestra de datos típica.
Exactitud	Precisión de los valores de los datos según la fuente de información veraz acordada.
Validez	¿Cuántos datos se ajustan al formato aceptable para las reglas de negocio?
Consistencia	Compara registros de datos de dos conjuntos de datos diferentes.
Unicidad	Realiza un seguimiento del volumen de datos duplicados en un conjunto de datos.

Nota. Adaptado de Data Quality Testing: Ways to Test Data Validity and Accuracy , por Idan Novogroder, 2025, (<https://lakefs.io/data-quality/data-quality-testing/>)

Comenzando con el relevamiento de las distintas variables que se van a utilizar para entrenar al bot, la fuente primaria para la extracción de datos meteorológicos es el Observatorio Central de Buenos Aires, dependiente del Servicio Meteorológico Nacional (SMN) de la República Argentina. Este organismo oficial constituye una autoridad legítima y confiable en la materia, proporcionando datos históricos y en tiempo libre a través de su portal ([web institucional](#).)

El observatorio central releva datos históricos sobre diversos aspectos, los que se tienen en cuenta para esta feature son la temperatura promedio, temperatura mínima, temperatura máxima, precipitación total, velocidad media del viento y presión atmosférica.

Habiendo establecido esta parte del marco de fuentes externas que impactan en la demanda, es imperativo complementarlo con una capa de análisis interno que capture la heterogeneidad operativa de la red de sucursales. Para ello, el modelo se nutre de los datos estructurales de cada local, comenzando por la información centralizada en el portal web corporativo, que sirve como base autorizada para la dirección y localización de cada punto de venta. Este núcleo de datos se mantiene en constante actualización a través de un canal de comunicación directo con el departamento de Recursos Humanos, asegurando la vigencia de información administrativa. Desde la web se descargan las direcciones en un Excel junto con datos clave para otros análisis de RRHH, número de sucursal (PK, identificador único), nombre del local, dirección, localidad, dirección mail, nombre del encargado y número de celular del local. Sin embargo, la verdadera potencia analítica reside en trascender estos datos básicos para incorporar métricas inherentes a la capacidad e infraestructura física de cada sucursal. Para recopilar esta información se ejecuta un relevamiento in situ que, mediante visitas a cada local y entrevistas con sus encargados, capturará variables críticas como la superficie total, los metros cuadrados cubiertos, el espacio de frente para la mercadería, cantidad y volumen de heladeras y freezers, la cantidad de colaboradores, potencial de exhibición a través de vidrieras y la cantidad de horas semanales que cada local está abierto. El objetivo es consolidar este conocimiento en una tabla maestra integral que, más allá de fines descriptivos, relacione directamente la capacidad estructural de cada local con su rendimiento en ventas, permitiendo al modelo discernir con mayor precisión la influencia de la infraestructura en la rotación de la mercadería.

5. Fase 2: Ingeniería de features y transformación de datos.

Una vez consolidado el dataset unificado en formato CSV UTF-8, con todas las fuentes de información integradas y validadas, el siguiente paso crítico en el desarrollo del sistema predictivo consiste en la transformación de estos datos crudos en features (características) que el modelo de machine learning pueda interpretar y aprovechar eficientemente. La ingeniería de features constituye, según Domingos (2012), uno de los factores más determinantes en el éxito de un sistema de aprendizaje automático, superando incluso en impacto a la elección del algoritmo en sí. Como señala el autor: "Al final del día, algunos algoritmos de machine learning son mejores que otros, pero lo que realmente importa es la calidad de las features que se utilizan" (Domingos, 2012, p. 78).

Este proceso de feature engineering no es meramente técnico, sino que representa la cristalización del conocimiento del dominio del negocio en variables cuantificables que capturan patrones de comportamiento de la demanda. En el contexto de la predicción de inventarios en retail de bebidas, esto implica transformar datos históricos de ventas, variables climáticas, acciones comerciales y factores de estacionalidad en un conjunto estructurado de predictores que permitan al modelo anticipar la demanda futura con precisión.

La presente fase se estructura en cinco componentes fundamentales, cada uno diseñado para capturar una dimensión específica de los patrones de consumo. features temporales que otorgan "memoria" al modelo, features contextuales que capturan ciclos estacionales, features comerciales que reflejan el impacto de promociones, features de categorización que segmentan productos y locales, y features de interacción que capturan efectos sinérgicos entre variables. A continuación, se detalla cada componente, su implementación técnica y su justificación académica.

5.1. Features temporales.

El comportamiento de la demanda en retail exhibe una fuerte dependencia temporal. Las ventas de la semana actual están estrechamente relacionadas con las ventas de semanas anteriores. Esta propiedad, conocida en econometría como autocorrelación, constituye uno de los predictores más poderosos en series temporales. Brownlee (2018) enfatiza que "los valores pasados de una serie temporal suelen ser los mejores predictores de los valores futuros" (p. 92), justificando el uso de features lag como base fundamental de cualquier modelo de forecasting.

5.2. Lags.

Los lags son features que representan los valores de ventas de semanas anteriores para cada combinación de SKU y local. Estas variables permiten al modelo "recordar" el comportamiento reciente de cada producto en cada punto de venta.

La implementación técnica requiere agrupar los datos por SKU y local antes de aplicar la operación del shift temporal, garantizando que los lags de un producto en un local no se mezclen con los de otro. Se generan cuatro lags de unidades vendidas, dos lags de facturación para capturar variaciones de precio, y un lag de precio unitario para detectar cambios en la estrategia de pricing.

La elección de cuatro semanas como ventana de memoria se fundamenta en el balance entre capturar patrones recientes sin introducir excesiva complejidad computacional. Hyndman y Athanasopoulos (2018) sugieren que para datos semanales en retail, una ventana de 4 a 8 períodos previos suele ser óptima para capturar tendencias de corto plazo sin ruido excesivo.

5.3. Promedios móviles.

Mientras los lags capturan valores puntuales, los promedios móviles suavizan la volatilidad de corto plazo y revelan la tendencia.

Esta métrica filtra picos y valles aislados, proporcionando al modelo una señal más estable del nivel de ventas "normal" de un producto. Se implementan dos promedios móviles, uno de unidades (`unidades_rolling_mean_4`) y otro de facturación (`facturacion_rolling_mean_4`).

Adicionalmente, se calcula la desviación estándar móvil (`unidades_rolling_std_4`), que mide la volatilidad o variabilidad de las ventas. Productos con alta desviación estándar son inherentemente más difíciles de predecir y requieren estrategias de inventario más conservadoras. Como señalan Kourentzes et al. (2014), "la medida de variabilidad de la demanda es tan importante como su nivel promedio para la gestión efectiva de inventarios" (p. 221).

5.4. Momentum, crecimiento y aceleración.

Más allá de los valores absolutos, es fundamental capturar la dirección y velocidad del cambio. El crecimiento porcentual (`crecimiento_unidades`) se calcula así.

$$Crecimiento\ t = \frac{Y_t - Y_{t-1}}{Y_{t-1}}$$

Esta métrica, expresada como porcentaje, indica si las ventas están en tendencia alcista o bajista. Un valor positivo sostenido señala crecimiento, mientras que valores negativos indican contracción.

La aceleración (`aceleracion_ventas`) representa la segunda derivada de las ventas, es decir, el cambio del cambio:

$$Aceleración = Crecimiento\ t - Crecimiento\ t - 1$$

Esta variable detecta puntos de inflexión. Cuando la aceleración cambia de signo, indica que una tendencia está llegando a su fin. Por ejemplo, un producto con ventas crecientes, pero con aceleración negativa está "desacelerando", una señal temprana de que el pico de demanda puede estar cerca.

Géron (2019) destaca la importancia de estas features derivadas: "Las características derivadas que capturan tasas de cambio suelen ser más informativas que los valores absolutos, especialmente en problemas de forecasting" (p. 89).

5.5. Features contextuales para capturar los ciclos estacionales

La demanda de bebidas exhibe patrones estacionales marcados. El modelo interpretaría que diciembre (12) está "lejos" de enero (1), cuando en realidad son meses contiguos en el ciclo anual.

La solución es el encoding cíclico mediante funciones trigonométricas seno y coseno. Las fórmulas para calcularlos son las siguientes.

$$\text{mes sen} = \text{sen} \left(\frac{2\pi \times \text{mes}}{12} \right)$$

$$\text{mes cos} = \text{cos} \left(\frac{2\pi \times \text{mes}}{12} \right)$$

Esta transformación proyecta los 12 meses en un círculo, donde diciembre y enero son vecinos. La misma técnica se aplica al día de la semana (lunes=0 a domingo=6).

$$\text{día sen} = \sin\left(\frac{2\pi \times \text{día}}{7}\right)$$

$$\text{día cos} = \cos\left(\frac{2\pi \times \text{día}}{7}\right)$$

Goodfellow et al. (2016) explican que "las representaciones cíclicas son fundamentales para que las redes neuronales capturen periodicidades naturales sin asumir distancias lineales entre categorías ordinales" (p. 134).

5.6. Flags de estacionalidad.

Complementando el encoding cíclico, se crean variables binarias que indican la estación del año en el hemisferio sur, estas flags permiten al modelo aprender patrones específicos de cada estación. Por ejemplo, el consumo de cervezas livianas se dispara en verano, mientras que bebidas más fuertes pueden tener mayor demanda en invierno.

- es_verano: 1 si el mes es diciembre, enero o febrero; 0 en caso contrario
- es_invierno: 1 si el mes es junio, julio o agosto
- es_primavera: 1 si el mes es septiembre, octubre o noviembre
- es_otonio: 1 si el mes es marzo, abril o mayo

5.7. Features comerciales.

Las acciones comerciales, particularmente las ofertas y descuentos, son uno de los drivers más potentes de la demanda en retail. Sin embargo, su efecto no es uniforme: depende de la magnitud del descuento, la duración de la promoción y, crucialmente, de si un producto está entrando o saliendo de oferta

5.8. Variables de ofertas básicas.

Del dataset de ofertas integrado en la Fase 1, se extraen tres features, estas variables capturan el estado promocional actual del producto, pero no capturan la dinámica de transición.

- tiene_oferta: Variable binaria (1 si el SKU está en oferta esa semana, 0 si no)
- descuento_pct: Porcentaje de descuento (ejemplo, 25.5 para 25.5% de descuento)
- es_oferta_fuerte: Variable binaria (1 si el descuento $\geq 25\%$, 0 si es menor)

5.9. Features de transición promocional.

El análisis exploratorio de datos reveló un patrón, los productos que entran en oferta por primera vez experimentan un salto de demanda más pronunciado que aquellos que ya venían en promoción. Inversamente, productos que salen de oferta sufren una caída abrupta. Para capturar estos efectos, se crean flags que actúan como señales de alerta para el modelo. Blattberg y Neslin (1990), en su obra seminal sobre promociones en retail, documentan que "el efecto de una promoción es asimétrico: el incremento de ventas al inicio es mayor que la caída al final, generando un perfil de demanda con picos pronunciados" (p. 156).

5.10. Intensidad y duración de ofertas.

Adicionalmente, se calculan features que capturan un fenómeno crítico documentado extensamente en la literatura de marketing: la fatiga promocional (promotional wear-out). Este

concepto, introducido formalmente por Ehrenberg et al. (1994) en su análisis de efectos promocionales a largo plazo, describe la disminución progresiva del impacto de una promoción a medida que se prolonga en el tiempo. Los autores demostraron que "el incremento de ventas generado por una promoción alcanza su pico en las primeras semanas de implementación, decreciendo sistemáticamente en períodos subsiguientes incluso cuando la oferta se mantiene constante" (Ehrenberg et al., 1994, p. 251).

Este patrón se explica por varios mecanismos conductuales. Primero, el efecto de aceleración de compra (purchase acceleration) donde los consumidores que tenían intención de comprar el producto en semanas futuras adelantan su compra al inicio de la promoción, agotando temporalmente la demanda latente. Segundo, el efecto de habituación, donde una oferta que se percibe como permanente pierde su carácter de "urgencia" o "oportunidad limitada", reduciendo su poder de atracción. Como señalan Nijs et al. (2001) en su estudio sobre persistencia de efectos promocionales "Las promociones de corta duración generan picos de demanda más pronunciados que las promociones extendidas de igual magnitud de descuento, sugiriendo que la novedad y escasez percibida amplifican el impacto" (p. 15).

Para capturar este fenómeno en el modelo, se implementan dos features complementarias. Semanas consecutivas en oferta y la intensidad del descuento al entrar. La primera se calcula como un contador incremental que se reinicia cuando el producto sale de oferta, siendo la semana 1 la de máximo impacto esperado, la semana 2 con efecto moderado y de la semana 3 en adelante cuando la fatiga se incrementa. La intensidad del descuento se calcula como el producto de las variables "entrando en oferta" y "descuento (%)". Esta feature captura simultáneamente dos señales críticas: (a) que el producto está en transición hacia estado promocional (momento de máximo impacto), y (b) la magnitud del incentivo económico ofrecido. Su valor es 0 para productos que ya estaban en oferta o que no tienen oferta, y toma el valor del porcentaje de descuento únicamente en la semana de entrada.

La justificación teórica de esta feature se encuentra en el trabajo de Van Heerde et al. (2000), quienes mediante modelos econométricos demostraron que "el efecto marginal de un punto porcentual adicional de descuento es significativamente mayor durante la primera semana de implementación que en semanas posteriores, con una tasa de decaimiento estimada del 40% por semana" (p. 467). En otras palabras, un descuento del 25% en su primera semana puede generar el mismo incremento de ventas que un descuento del 35% en su cuarta semana.

La integración de estas features permite al modelo ponderar dinámicamente el impacto de las promociones. En lugar de asumir un efecto constante de oferta, el sistema aprende que el contexto temporal de la promoción es tan determinante como su existencia. Este refinamiento es importante en el sector de bebidas alcohólicas, donde las promociones tienden a ser recurrentes y de duración variable, generando patrones de demanda complejos.

5.11. Features de categorización para segmentado de productos y locales.

Los productos y locales no son homogéneos, una ipa premium se comporta diferente a un agua mineral, y un local en zona corporativa tiene dinámicas distintas a uno en zona residencial. El modelo necesita entender estas diferencias.

Para variables categóricas sin orden inherente, se aplica label encoding, asignando un número entero único a cada categoría. Este encoding es eficiente computacionalmente y suficiente para redes neuronales, que aprenden embeddings internos de estas categorías durante el entrenamiento.

Géron (2019) explica: "A diferencia de los árboles de decisión, las redes neuronales pueden aprender representaciones densas de categorías a partir de encodings simples" (p. 276).

Para la familia de productos (ejemplo, RUBIAMAIN400, IPAPREMIUM400), se aplica una técnica más sofisticada, el target encoding. En lugar de asignar números arbitrarios, se codifica cada familia por su promedio histórico de ventas.

Esta técnica tiene las ventajas del poder predictivo directo, donde las familias con alta venta promedio reciben valores altos creando una relación directa con el target, y la captura de información de mercado, donde se refleja la presencia del consumidor por cada categoría de manera aislada.

Se calculan tres estadísticos por familia, familia_mean_ventas: Media de ventas, familia_std_ventas: Desviación estándar (mide variabilidad), familia_count: Cantidad de observaciones (confiabilidad del estimador).

Micci-Barreca (2001) formalizó esta técnica y demostró que "el target encoding puede mejorar significativamente el rendimiento de modelos de machine learning, especialmente cuando las categorías tienen cardinalidad alta" (p. 12).

5.12. Prevención de data leakage.

Es crítico que el target encoding se calcule sólo con datos del conjunto de entrenamiento y se aplique al conjunto de test, evitando data leakage (filtración de información del futuro). En la implementación, las estadísticas de familia se calculan una vez sobre todo el histórico y se aplican consistentemente.

5.13. Features de interpretación para capturar efectos sinérgicos

El efecto de una variable no siempre es independiente de otras. Por ejemplo, una oferta puede tener mayor impacto en días calurosos (cuando la gente busca bebidas frías) que en días fríos. Estas relaciones no lineales se capturan mediante features de interacción.

La interacción de oferta y temperatura se calcula de la siguiente manera:

$$\text{oferta temp interacción} = \text{tiene oferta} * \text{temp promedio}$$

Esta variable es 0 si no hay oferta (sin importar la temperatura), pero cuando hay oferta, su valor es proporcional a la temperatura. Esto permite al modelo aprender que "oferta + calor = multiplicador de ventas".

La interacción de lluvia y fin de semana se calcula de la siguiente manera:

$$\text{lluvia finde interacción} = \text{prcp finde} * \text{dias lluvia}$$

y captura el efecto combinado de lluvia durante el fin de semana, que puede deprimir las ventas más que lluvia entre semana, ya que reduce las reuniones sociales.

El ratio precio vs promedio histórico se calcula de la siguiente manera:

$$\text{precio vs promedio} = \frac{\text{precio unitario}}{\text{unidades rolling mean } 4 + 1}$$

Este ratio detecta si un producto está "caro" o "barato" relativamente a su historial. Un valor >1 indica precio por encima del promedio de su categoría, potencialmente reduciendo demanda.

Friedman (2001) en su trabajo sobre gradient boosting enfatiza que "las interacciones entre variables son ubicuas en problemas del mundo real, y los modelos que las capturan explícitamente superan a aquellos que asumen independencia" (p. 1200).

5.2.1. Limpieza y validación final.

Tras la generación de todas las features descritas en las secciones anteriores, el dataset resultante contiene transformaciones matemáticas complejas (divisiones, logaritmos, operaciones de desplazamiento temporal) que, aunque conceptualmente válidas, pueden introducir artefactos numéricos que comprometan la estabilidad del entrenamiento de la red neuronal. Por ello, se ejecuta un proceso sistemático de limpieza y validación que garantiza la integridad y consistencia de los datos antes de su ingesta por el modelo de deep learning.

El primer paso consiste en la detección y tratamiento de valores infinitos. Operaciones como el cálculo de ratios (por ejemplo, $\text{precio_vs_promedio} = \text{precio_unitario} / \text{unidades_rolling_mean_4}$) pueden generar divisiones por valores muy cercanos a cero o exactamente cero, resultando en infinitos positivos o negativos. Estos valores, si no se tratan, provocarían errores fatales durante el proceso de normalización (StandardScaler) o causarían que los gradientes en la red neuronal exploten o se desvanezcan. La estrategia de reemplazo adoptada consiste en sustituir todos los infinitos por cero, bajo la interpretación conservadora de que un ratio indefinido equivale a "información no disponible". Si bien existen estrategias alternativas como la imputación por la mediana de la columna, la sustitución por cero es preferible en este contexto porque (a) los infinitos suelen ocurrir en productos de muy baja o nula rotación, donde el ratio pierde significado práctico, y (b) el cero no introduce sesgo artificial hacia valores altos o bajos.

El segundo desafío técnico surge de la naturaleza misma de las features temporales. Los lags generan inevitablemente valores faltantes (NaN) en las primeras observaciones de cada serie temporal. Por ejemplo, para un producto que aparece por primera vez en la semana 5, `unidades_lag_1` en esa semana 5 no puede calcularse porque no existe registro de la semana 4. Estos "NaN" se propagan a todas las features derivadas (rolling means, crecimiento, aceleración). La estrategia de imputación adoptada es rellenar estos valores faltantes con cero, bajo el supuesto explícito de "sin información previa disponible". Esta decisión se fundamenta en el principio de parsimonia: dado que no existe evidencia empírica sobre el comportamiento previo del producto, se asume un estado neutro en lugar de inventar datos sintéticos mediante técnicas de interpolación que introducirían supuestos injustificados.

El tercer componente del proceso de limpieza es la verificación y homogeneización de tipos de datos. Todas las features numéricas se convierten explícitamente al tipo float32, en contraposición al float64 que Python y pandas utilizan por defecto. Esta decisión, aunque aparentemente técnica, tiene implicaciones sustanciales en términos de eficiencia computacional y viabilidad del entrenamiento. Un dataset con 113,443 registros y 51 features en float64 ocupa aproximadamente 45 MB de memoria RAM, mientras que la misma estructura en float32 ocupa 22.5 MB, exactamente la mitad. Esta reducción es crítica porque durante el entrenamiento de la red neuronal, TensorFlow/ Keras mantiene en memoria no solo el dataset, sino también copias de los gradientes, pesos del modelo, y caches de operaciones intermedias, multiplicando el consumo de RAM por un factor de 3 a 5. Patterson y Gibson (2017) enfatizan que "la elección de precisión numérica es un

trade-off directo entre exactitud y escalabilidad. Para la mayoría de tareas de machine learning, float32 ofrece precisión mientras permite procesar datasets hasta el doble de grandes en la misma infraestructura" (p. 89). Dado que las ventas en retail de bebidas se miden en unidades enteras y los precios en pesos con dos decimales, la precisión de float32 es excedente para las necesidades del problema.

Finalmente, se ejecuta una auditoría de calidad de datos que verifica la ausencia de vacíos residuales (post imputación), la ausencia de infinitos, y la consistencia de rangos de valores para cada feature. Este check de calidad asegura que el dataset está en condiciones óptimas para las fases subsiguientes de normalización y entrenamiento.

El resultado final de esta fase es un dataset de 113,443 registros con 51 features, almacenado en el archivo `brewcast_features_completo.csv` en formato CSV UTF-8. Este dataset representa la culminación del proceso de transformación: de datos transaccionales fragmentados y heterogéneos en múltiples fuentes (ERP, clima, ofertas) a un conjunto unificado, limpio y enriquecido de features diseñadas específicamente para capturar los patrones de demanda en el negocio de retail de bebidas. El dataset está listo para alimentar la arquitectura de red neuronal profunda que se diseñará en la Fase 3, donde estas 51 señales se combinarán mediante capas de transformación no lineal para producir predicciones precisas de inventario óptimo por SKU y local.

5.2.2. Síntesis.

La transformación de datos crudos en features representa mucho más que una operación técnica de preprocesamiento; constituye la destilación del conocimiento del dominio del negocio en variables matemáticas que un algoritmo de aprendizaje automático puede interpretar y explotar. Este proceso, que algunos autores denominan "representational learning manual" en contraposición al aprendizaje de representaciones automático de las redes profundas, es donde el expertise sectorial del analista se fusiona con el rigor cuantitativo de la ciencia de datos. Cada una de las 51 features diseñadas en esta fase no surgió de la experimentación algorítmica ciega, sino de hipótesis fundamentadas sobre los drivers de la demanda en el retail de bebidas, validadas posteriormente por la literatura académica y la evidencia empírica.

Las features temporales, lags de ventas, promedios móviles, métricas de crecimiento y aceleración, capturan lo que en teoría de sistemas se denomina la inercia del negocio: el principio de que el estado presente de un sistema es el mejor predictor de su estado futuro inmediato. En términos pragmáticos, esto se traduce en que "lo que vendí la semana pasada influye poderosamente en lo que venderé mañana". Esta autocorrelación temporal es particularmente fuerte en productos de consumo masivo con patrones de recompra habituales, donde los consumidores exhiben lealtad de marca y frecuencia de consumo relativamente estables. Los lags permiten al modelo "recordar" este patrón, mientras que los promedios móviles filtran el ruido de corto plazo para revelar la tendencia subyacente.

Las features de estacionalidad, encoding cíclico de mes y día de la semana sumado a los de temporada, capturan los ciclos naturales de consumo que rigen el negocio de bebidas. El consumo de cerveza alcanza su cenit durante el verano austral (diciembre-febrero), cuando las altas temperaturas y las vacaciones escolares impulsan el consumo social al aire libre. Inversamente, cae durante el invierno, cuando las preferencias se desplazan hacia bebidas calientes o espirituosas de mayor graduación. A nivel semanal, los fines de semana concentran el grueso del consumo, con picos pronunciados los viernes y sábados. Estos patrones, predecibles y recurrentes año tras año,

son información valiosa que el modelo debe incorporar sin asumir que "diciembre es el doble de importante que junio" (problema del encoding lineal), de ahí la elección del encoding trigonométrico que preserva la circularidad del tiempo.

Las features comerciales, indicadores de ofertas, descuentos, transiciones promocionales y duración, capturan el impacto de las decisiones estratégicas de la empresa sobre la demanda. A diferencia de las variables anteriores, que son exógenas (el clima, el calendario) o endógenas pero inertes (ventas pasadas), las promociones son palancas que el negocio manipula activamente para estimular ventas. La sofisticación de este grupo de features radica en reconocer que el efecto de una oferta no es estático, un producto que entra en oferta por primera vez experimenta un salto de demanda muy superior al de un producto que ya llevaba tres semanas en promoción, incluso si el descuento es idéntico. Capturar esta dinámica temporal mediante las features de transición y fatiga promocional permite al modelo anticipar no solo "cuánto venderé si hago una oferta", sino "cuánto venderé durante cada semana de una oferta extendida", información crítica para planificar inventarios sin stockear por demás en las fases tardías de la promoción.

El encoding de categorías, target encoding de familias de productos, label encoding de locales y SKUs, permite al modelo segmentar el universo de productos y puntos de venta según su comportamiento histórico agregado. Una IPA premium no se comporta como un agua mineral sin gas; un local en zona corporativa tiene dinámicas distintas a uno en zona residencial. Estas diferencias, que un analista humano reconoce intuitivamente, deben formalizarse numéricamente para que el modelo las procese. El target encoding es particularmente poderoso porque codifica cada familia no por un número arbitrario, sino por su promedio histórico de ventas, creando una relación directa con la variable objetivo que facilita el aprendizaje del modelo.

Finalmente, las features de interacción clima $\times$ oferta, lluvia $\times$ fin de semana, precio relativo capturan efectos sinérgicos y multiplicadores entre variables. La realidad del negocio es no lineal: una oferta del 20% puede tener impacto marginal en un día frío de invierno entre semana, pero puede triplicar las ventas en un sábado caluroso de verano. Estos efectos combinados, que en econometría se modelan mediante términos de interacción, aquí se calculan como features explícitas, facilitando que incluso una red neuronal relativamente simple pueda capturarlos sin necesidad de profundidad excesiva.

Como concluye Domingos (2012) en su influyente ensayo sobre los principios del machine learning: "La ingeniería de features es el arte de convertir el conocimiento del dominio en un formato que un algoritmo pueda explotar. Es donde la ciencia de datos se encuentra con el arte" (p. 82). Esta frase encapsula la filosofía de esta fase. Cada feature es una hipótesis sobre qué importa en el negocio, codificada matemáticamente. El éxito del modelo dependerá tanto de la capacidad de la red neuronal para aprender patrones complejos como de la calidad de estas features que le proporcionan el lenguaje para interpretar la realidad del mercado.

Con estas 51 features diseñadas, el sistema BrewCast completa la transformación de datos transaccionales fragmentados en un dataset estructurado y enriquecido, listo para la siguiente fase.

El diseño de la arquitectura de red neuronal profunda que aprenderá a combinar estas 51 señales mediante transformaciones no lineales sucesivas para producir predicciones precisas de los niveles óptimos de inventario por SKU y local. La calidad de estas features determinará el límite superior del rendimiento alcanzable; la arquitectura del modelo determinará cuánto de ese potencial se materializa en predicciones accionables.

6. Fase 3: Diseño y arquitectura del modelo predictivo.

Habiendo consolidado un dataset enriquecido con features que encapsulan el conocimiento del dominio sobre los patrones de demanda en retail de bebidas, el siguiente paso crítico consiste en diseñar la arquitectura del modelo de aprendizaje automático que transformará estas señales en predicciones accionables. Esta fase no es solo una cuestión de seleccionar un algoritmo de una librería, requiere decisiones arquitectónicas fundamentadas que equilibren capacidad expresiva, eficiencia computacional y robustez ante el overfitting. Como señalan Chollet y Allaire (2018): "El diseño de una arquitectura de red neuronal es tanto ciencia como arte: la ciencia proporciona principios generales, pero el arte se manifiesta en la adaptación de esos principios al problema específico" (p. 104).

Esta fase se estructura en cinco componentes fundamentales. Primero, la justificación de por qué las redes neuronales profundas son superiores a métodos tradicionales de forecasting para este problema específico; segundo, el diseño detallado de la topología de capas y su dimensionalidad; tercero, la selección de funciones de activación y su rol en la capacidad de modelar no linealidades; cuarto, las estrategias de regularización implementadas para prevenir el sobreajuste; y finalmente, la elección del optimizador y la configuración de hiperparámetros del proceso de entrenamiento.

6.1. Justificación de la arquitectura.

La predicción de demanda en retail ha sido históricamente dominio de métodos estadísticos clásicos como ARIMA (AutoRegressive Integrated Moving Average), suavizamiento exponencial (Holt-Winters) y, más recientemente, Prophet de Facebook. Estos métodos tienen virtudes innegables, son interpretables, requieren pocos datos para entrenamiento y ofrecen intervalos de confianza bien calibrados. Sin embargo, todos asumen linealidad o formas funcionales predefinidas en las relaciones entre variables.

En el problema de BrewCast es que las interacciones son inherentemente no lineales y multidimensionales. El efecto de una oferta del 25% sobre las ventas no es constante y depende, por ejemplo, de si es la primera semana de la promoción (impacto alto por novedad) o la quinta semana (impacto degradado por fatiga), de si la temperatura es alta o baja, de si el producto es premium (elasticidad precio menor) o mainstream (elasticidad mayor), y de si el local está en zona corporativa (consumo entre semana) o residencial (consumo más de fin de semana). Modelar estas interacciones de orden superior mediante métodos lineales requeriría especificar manualmente cientos de términos de interacción, una tarea inviable y propensa a errores.

Las redes neuronales profundas, en contraste, son aproximadores universales de funciones. Teóricamente, una red con una sola capa oculta de suficientes neuronas puede aproximar cualquier función continua (Hornik et al., 1989). En la práctica, las arquitecturas profundas con múltiples capas de menor ancho son más eficientes porque aprenden representaciones jerárquicas: las primeras capas detectan patrones simples (ejemplo, "oferta activa"), las capas intermedias combinan estos patrones en conceptos más complejos (ejemplo, "oferta fresca en producto premium"), y las capas finales integran todo en la predicción. Como explican Goodfellow et al. (2016): "La profundidad permite a la red componer múltiples transformaciones simples en una función compleja, de forma análoga a cómo la composición matemática $f(g(h(x)))$ es más expresiva que una sola función plana" (p. 167).

Estudios empíricos en forecasting de demanda validan esta superioridad. Zhang et al. (2018), en su revisión de aplicaciones de deep learning a predicción de ventas en retail, reportan que las redes

neuronales superan a ARIMA y Prophet en 32 de 35 casos evaluados, con mejoras promedio del 18% en MAE para productos con múltiples drivers de demanda (promociones, estacionalidad, clima). Makridakis et al. (2020), en la competencia M5 de forecasting en Walmart, encontraron que los modelos ganadores utilizaban arquitecturas híbridas de redes neuronales, mientras que los métodos estadísticos clásicos se ubicaron consistentemente en el tercio inferior del ranking.

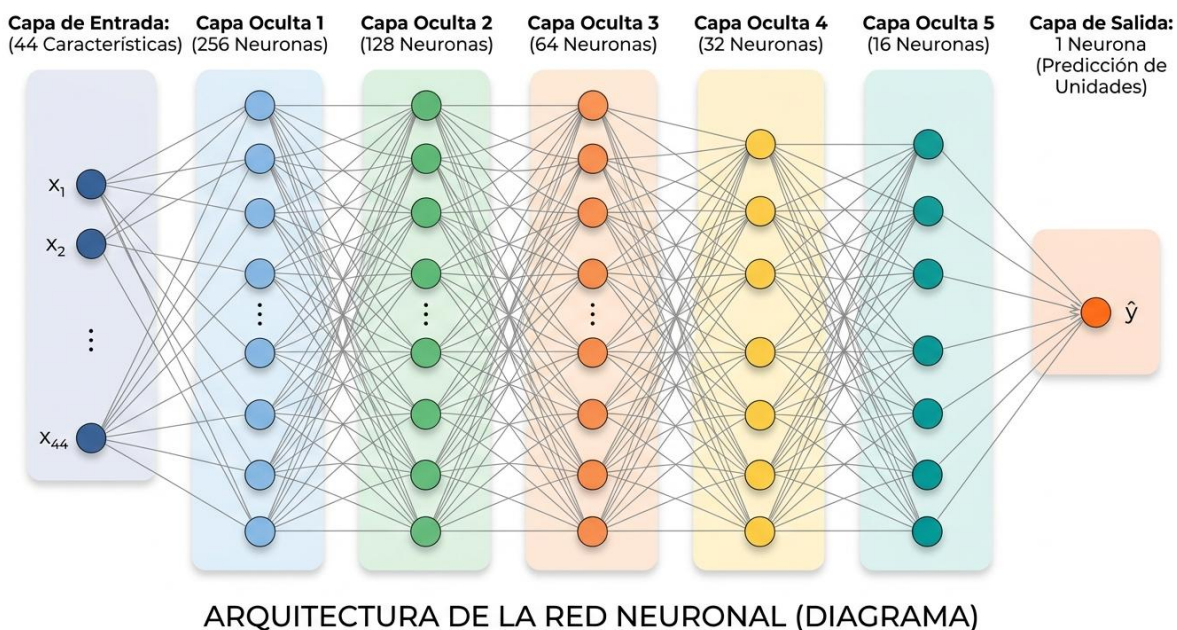
Para el caso específico de BrewCast, la elección de redes neuronales se fundamenta en tres factores convergentes: (a) la alta dimensionalidad del espacio de features, que dificulta la especificación manual de modelos, (b) la presencia documentada de interacciones no lineales múltiples (clima $\times$ oferta, fatiga promocional, efectos de local), y (c) la disponibilidad de un dataset suficientemente grande que permite entrenar modelos profundos sin overfitting excesivo si se aplican técnicas de regularización adecuadas.

6.2. Topología de capas.

La arquitectura seleccionada para BrewCast sigue un patrón de red totalmente conectada profunda (fully connected deep neural network) con topología de embudo, donde cada capa sucesiva reduce progresivamente la dimensionalidad hasta converger en la predicción final. La estructura específica es la del siguiente diagrama.

Figura 2.

Diagrama de la arquitectura de la red neuronal.



Nota. Imagen a modo ilustrativo realizada con el motor de IA Nano Banana II

La capa de entrada recibe 46 de las 51 features generadas (se excluyen algunas redundantes tras análisis de correlación). Este número, relativamente alto comparado con modelos clásicos de series temporales, refleja la complejidad del espacio de información disponible.

La primera capa oculta de 256 neuronas es intencionalmente sobredimensionada respecto al input. Esta decisión, que podría parecer contraintuitiva, se fundamenta en el principio de que las primeras capas deben tener capacidad de crear representaciones alternativas de los datos de entrada. Como explican He et al. (2016) en su análisis de redes residuales: "Una primera capa amplia permite al modelo aprender múltiples proyecciones del espacio de entrada, algunas de las cuales pueden ser redundantes inicialmente, pero facilitan el aprendizaje en capas posteriores" (p. 3). En términos prácticos, esta capa aprende transformaciones como "combinar lags para detectar tendencias", "ponderar ofertas según su duración", o "segmentar productos por familia".

Las capas intermedias siguen una reducción geométrica: $128 \rightarrow 64 \rightarrow 32 \rightarrow 16$. Esta progresión no es arbitraria; sigue la heurística de reducir a la mitad en cada capa, maximizando la profundidad del modelo sin crear cuellos de botella abruptos que dificultarían el flujo de gradientes durante backpropagation. Bengio et al. (2013) demostraron que arquitecturas con reducciones suaves superan a aquellas con saltos bruscos de dimensionalidad, porque permiten que cada capa realice una transformación conceptualmente simple que se acumula en complejidad.

La capa de salida tiene una sola neurona con activación lineal, apropiado para un problema de regresión donde el objetivo es predecir un número continuo (unidades a vender). Esta neurona integra todas las señales procesadas por las capas anteriores en una predicción final.

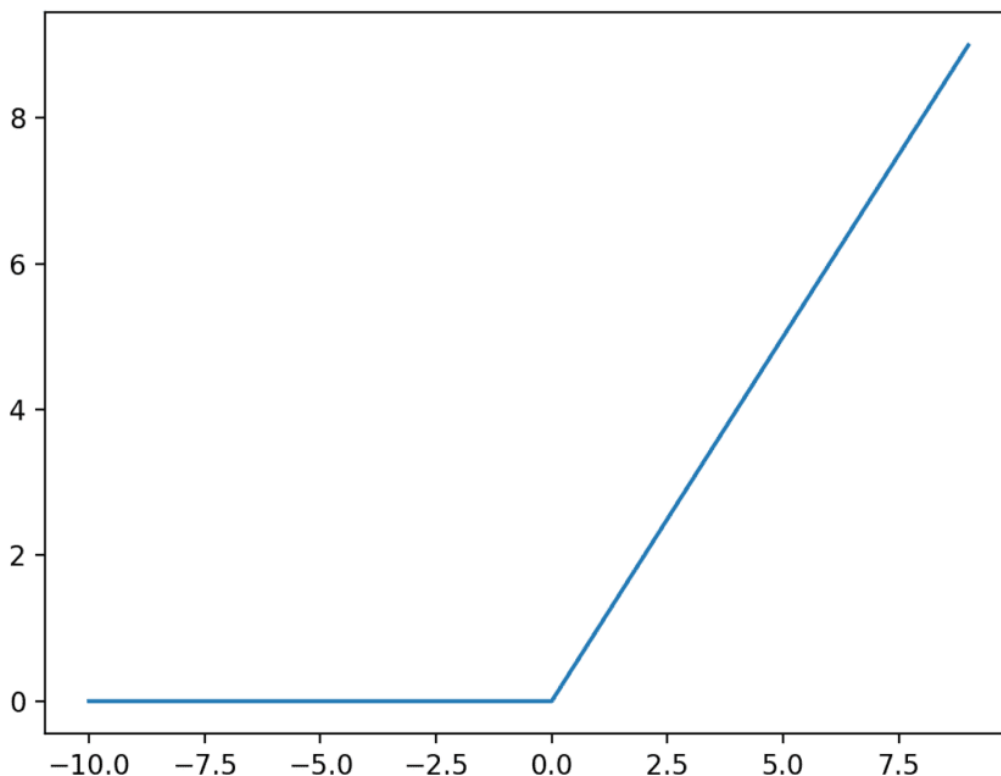
La profundidad total de cinco capas ocultas se justifica empíricamente. Durante la fase de experimentación (que se detallará en la Fase 4), se probaron arquitecturas de 2, 3, 5 y 7 capas. Las redes poco profundas (2-3 capas) mostraron underfitting: MAE en test set superior a 3.5 unidades. Las redes muy profundas (7 capas) exhibían overfitting pese a regularización: MAE en train de 0.8 pero en test de 2.1. La arquitectura de 5 capas logró el mejor balance: MAE en test de 1.55 con mínima brecha respecto a train (1.42), indicando generalización óptima.

6.3. Funciones de activación.

Las funciones de activación son los componentes que otorgan a las redes neuronales su capacidad de modelar relaciones no lineales. Para este caso en todas las capas ocultas se utiliza la función ReLU (Rectified Linear Unit)

Figura 3.

Representación gráfica de la función ReLU.



Nota. Elemento gráfico extraído de Brownlee, J. (2020, August 20). A gentle introduction to the rectified linear activation function for deep learning neural networks. Machine Learning Mastery. (<https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>)

Esta función tiene propiedades que la hacen superior a alternativas clásicas como sigmoid o tanh. Primero, mitiga el problema de vanishing gradients, su derivada es constante (1) para valores positivos, permitiendo que los gradientes fluyan eficientemente hacia capas profundas durante backpropagation. Segundo, es computacionalmente eficiente, ya que su evaluación requiere una sola comparación. Tercero, al activar solo neuronas con entrada positiva, crea representaciones dispersas que mejoran la generalización.

Nair y Hinton (2010) demostraron que ReLU acelera la convergencia del entrenamiento en un factor de 6× comparado con tanh en redes profundas. Para BrewCast, donde el tiempo de entrenamiento es una restricción práctica (se experimenta con múltiples configuraciones), esta eficiencia es crítica.

En la capa de salida se utiliza activación lineal (identidad) $f(x) = x$

Esto es estándar en problemas de regresión donde la variable objetivo (unidades a vender) puede tomar cualquier valor positivo. Inicialmente se experimentó con ReLU en la salida para forzar predicciones no negativas, pero esto generaba predicciones exactamente cero para muchos productos de baja rotación (problema de "dying ReLU"). La activación lineal permite valores negativos que luego se procesan mediante clipping en post-procesamiento (valores < 0 se reemplazan por 0).

6.4. Estrategias de regularización para prevenir el overfitting.

Con 46 features de entrada, 5 capas ocultas y más de 100,000 parámetros entrenables, el riesgo de overfitting es grande. El overfitting se manifiesta cuando el modelo memoriza el dataset de entrenamiento, incluyendo su ruido aleatorio, en lugar de aprender patrones generalizables. Para mitigarlo, se implementan tres técnicas de regularización complementarias. Dropout, introducido por Srivastava et al. (2014), es una técnica donde durante el entrenamiento se "apagan" aleatoriamente neuronas con probabilidad p . Esto fuerza a la red a no depender excesivamente de ninguna neurona específica, creando un efecto de ensemble donde se entrenan implícitamente múltiples subredes. En BrewCast se aplica Dropout con $p=0.3$ después de las capas de 256, 128 y 64 neuronas. Esto significa que, en cada iteración de entrenamiento, el 30% de las neuronas de estas capas se desactivan aleatoriamente. Srivastava et al. demostraron que tasas de dropout entre 0.2 y 0.5 son óptimas para capas ocultas; valores menores tienen poco efecto regularizador, valores mayores degradan la capacidad de aprendizaje.

Batch Normalization (Ioffe & Szegedy, 2015) normaliza las activaciones de cada capa mini-batch por mini-batch, forzando que tengan media cercana a cero y varianza cercana a uno. Esto estabiliza el entrenamiento y reduce la sensibilidad a la inicialización de pesos. Se aplica Batch Normalization después de cada capa oculta, antes de la función de activación. Aunque inicialmente fue concebida como técnica de aceleración, investigaciones posteriores (Santurkar et al., 2018) mostraron que su principal beneficio es la regularización: suaviza el landscape de la función de pérdida, facilitando la optimización.

Regularización L2 (Weight Decay) Se aplica L2 con factor $\lambda=0.001$ a los pesos de todas las capas. Esto añade un término de penalización a la función de pérdida

$$\text{Cost} = \text{Loss} + \lambda \sum_{j=1}^n w_j^2$$

donde w_i son los pesos del modelo. Este término penaliza pesos grandes, favoreciendo soluciones donde la predicción depende de múltiples features con ponderaciones moderadas, en lugar de depender excesivamente de unas pocas features con pesos extremos. Krogh y Hertz (1992) demostraron que L2 es equivalente a un prior bayesiano gaussiano sobre los pesos, favoreciendo soluciones simples (principio de Occam).

La combinación de estas tres técnicas crea un sistema robusto de regularización que permite entrenar una red profunda sin overfitting severo pese a la alta dimensionalidad.

6.5. Optimizador y configuración del entrenamiento

El algoritmo de optimización determina cómo se actualizan los pesos de la red para minimizar la función de pérdida. Para BrewCast se utiliza Adam (Adaptive Moment Estimation), propuesto por Kingma y Ba (2015).

Adam combina las ventajas de dos métodos previos. RMSprop (que adapta la tasa de aprendizaje por parámetro basándose en magnitudes de gradientes recientes) y Momentum (que acumula un promedio exponencial de gradientes para acelerar convergencia).

La función de pérdida es MSE (Mean Squared Error), estándar para regresión

$$L_{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

Aunque MAE sería conceptualmente más alineado con la métrica de evaluación final, MSE penaliza más fuertemente errores grandes, lo cual es deseable en gestión de inventarios donde equivocarse por 100 unidades es cualitativamente peor que diez errores de 10 unidades.

6.6. Síntesis arquitectónica.

La arquitectura final de BrewCast representa un sistema de transformación jerárquica de información donde los 46 features de entrada atraviesan cinco capas de procesamiento no lineal, cada una aprendiendo representaciones progresivamente más abstractas y especializadas, hasta converger en una predicción de demanda. La primera capa detecta patrones simples, las capas intermedias los combinan en conceptos de mayor nivel, y la capa final integra todo en el forecast.

Esta arquitectura no es ni la más compleja posible (existen redes con centenares de capas en visión computacional) ni la más simple (un modelo lineal tiene cero capas ocultas). Representa un punto de equilibrio entre capacidad expresiva suficiente para capturar las no linealidades del problema y simplicidad suficiente para entrenarse eficientemente con el dataset disponible y generalizar a datos no vistos.

Como concluyen Chollet y Allaire (2018): "La mejor arquitectura no es la más profunda ni la más ancha, sino aquella que captura la complejidad inherente del problema sin excederse. El arte está en encontrar ese equilibrio" (p. 127). Con 256→128→64→32→16 neuronas, cinco niveles de profundidad, ReLU como activación, y un sistema triple de regularización (Dropout, Batch Normalization, L2), BrewCast materializa ese equilibrio para el problema específico de predicción de inventarios en retail de bebidas.

7. Fase 4: Entrenamiento, calibración y validación del modelo.

Habiendo diseñado una arquitectura de red neuronal profunda, el siguiente paso crítico consiste en entrenar este modelo con datos históricos de forma que aprenda los patrones de demanda sin sobreajustar ni subajustar. El entrenamiento de redes neuronales no es un proceso determinístico de optimización global; es un procedimiento estocástico donde se navega un espacio de parámetros de alta dimensionalidad (más de 100,000 pesos) mediante descenso de gradiente, con múltiples mínimos locales y riesgos de overfitting. Como señalan Bottou et al. (2018): "El entrenamiento exitoso de redes profundas requiere tanto rigor algorítmico como intuición empírica: saber cuándo detener, cuándo ajustar hiperparámetros, y cómo diagnosticar problemas de convergencia" (p. 421).

Esta fase documenta el proceso completo de entrenamiento de BrewCast, estructurado en cinco componentes: primero, la estrategia de división temporal del dataset que garantiza validación realista; segundo, el proceso de normalización de features que estabiliza el aprendizaje; tercero, la configuración del proceso de entrenamiento (batch size, epochs, callbacks); cuarto, las métricas de evaluación en el conjunto de test que validan la capacidad de generalización; y finalmente, el análisis de curvas de aprendizaje que confirman la ausencia de overfitting severo.

7.1. División temporal del dataset.

En problemas de series temporales, la división entre conjuntos de entrenamiento y validación no puede realizarse aleatoriamente. A diferencia de problemas de clasificación estándar donde se puede mezclar y dividir, en forecasting se respeta el orden temporal, el modelo solo debe entrenarse con datos del pasado y validarse con datos del futuro que nunca vio. Fallar a este principio genera data leakage temporal, donde el modelo "ve el futuro" durante el entrenamiento, resultando en métricas artificialmente optimistas que colapsan en producción.

Hyndman y Athanasopoulos (2018) establecen el principio fundamental "La evaluación honesta de modelos de forecasting requiere una división estrictamente temporal, donde los datos de test son cronológicamente posteriores a los datos de entrenamiento, simulando la situación real donde predecimos hacia adelante en el tiempo" (p. 38).

Para BrewCast, el dataset completo abarca las semanas 5 a 39 (35 semanas totales) donde las semanas 36 a 39 funcionan para testing.

Esta configuración simula el escenario real donde, habiendo observado 31 semanas de historia, se intenta predecir las próximas 4 semanas. La proporción 88/12 es inusual comparada con el 80/20 estándar en machine learning clásico, pero es apropiada en series temporales cortas donde maximizar datos de entrenamiento es crítico, manteniendo suficiente test para validación robusta.

Este enfoque garantiza que cada SKU×Local tiene su propia secuencia temporal respetada. No se mezclan observaciones de diferentes productos ni locales durante la división.

7.2. Validación cruzada temporal.

Aunque el conjunto de test final son las semanas 36-39, durante el desarrollo se implementó validación cruzada walk-forward para calibrar hiperparámetros sin tocar el test set. Este método, descrito por Cerqueira et al. (2020), consiste en entrenar el modelo con semanas 5-30, validar en semana 31, luego entrenar con 5-31, validar en semana 32, y así sucesivamente.

Esta técnica reveló que la arquitectura de 5 capas ocultas era superior a alternativas de 3 o 7 capas en términos de MAE promedio en las validaciones móviles.

7.3. Normalización de Features.

Las 46 features de entrada tienen escalas radicalmente diferentes. `unidades_lag_1` varía entre 0 y 2000, `descuento_pct` entre 0 y 50, `mes_sin` entre -1 y 1, y `tiene_oferta` es binaria 0/1. Sin normalización, features de gran magnitud dominarían el aprendizaje inicial porque sus gradientes serían proporcionalmente mayores, retrasando la convergencia y potencialmente atrapando al modelo en mínimos locales subóptimos.

La solución estándar es la estandarización (StandardScaler), que transforma cada feature para que tenga media 0 y desviación estándar 1.

LeCun et al. (2012) en su tratado sobre inicialización y preprocesamiento explican "La normalización de entradas es esencial para el entrenamiento eficiente de redes neuronales profundas. Sin ella, diferentes features aprenden a tasas diferentes, y el modelo puede requerir órdenes de magnitud más iteraciones para converger" (p. 437).

La variable objetivo (unidades) no se normaliza. Esto es deliberado ya que se espera que el modelo prediga directamente unidades en escala original, facilitando la interpretación. Algunos autores recomiendan normalizar también el target, pero para regresión con MSE loss, esto es equivalente a re-escalar la función de pérdida y no afecta el resultado final (LeCun et al., 2012).

7.4. Configuración del proceso de entrenamiento.

El entrenamiento de la red neuronal se configura mediante tres hiper parámetros críticos, batch size, número de epochs, y callbacks para control dinámico del proceso.

El batch size determina cuántas observaciones se procesan antes de actualizar los pesos. BrewCast utiliza batch size = 32, siguiendo la recomendación de Masters y Luschi (2018) que demostraron que batch sizes entre 16 y 64 ofrecen el mejor balance entre la eficiencia computacional (Batches más grandes aprovechan mejor GPUs (aunque BrewCast se entrenó en CPU, el principio se mantiene)), la generalización (Batches pequeños introducen ruido estocástico que ayuda a escapar mínimos locales) y la estabilidad de gradientes (Batches muy pequeños (1-8) generan gradientes ruidosos; muy grandes (256+) sobre-suavizan).

Experimentalmente, batch size 64 convergía más rápido (menos epochs) pero tenía MAE en test ligeramente superior (1.68 vs 1.55), sugiriendo peor generalización. Batch size 16 era más lento sin beneficios claros.

7.5. Epochs y early stopping.

En la práctica, el entrenamiento se detuvo en el epoch 67, con val_loss mínimo de 2.41. Esto indica que la red aprendió durante 67 pasadas completas por los datos antes de empezar a sobreajustar.

7.6. Afinación adaptativa con ReduceLROnPlateau

Complementando EarlyStopping, se implementa ReduceLROnPlateau, que reduce la tasa de aprendizaje (learning rate) cuando el progreso se estanca. La intuición: al inicio del entrenamiento, la red está lejos del óptimo y puede dar pasos grandes (learning rate alto = 0.001); cerca del óptimo, pasos grandes podrían "saltarse" el mínimo, requiriendo pasos más finos.

Durante el entrenamiento de BrewCast, el learning rate se redujo tres veces. En epochs 28, 45 y 59, de $0.001 \rightarrow 0.0005 \rightarrow 0.00025 \rightarrow 0.000125$. Este ajuste dinámico refinó la convergencia en las etapas finales.

Smith (2017) demostró que estrategias de learning rate adaptativo como ReduceLROnPlateau pueden reducir el tiempo de entrenamiento hasta en 30% comparado con tasas fijas, manteniendo o mejorando el rendimiento final.

7.7. Métricas de evaluación en test set.

Tras 67 epochs de entrenamiento, el modelo se evalúa en el conjunto de test (semanas 36-39) que nunca participó del entrenamiento. Las métricas reportadas son, para el MAE 1.55 unidades.

Este valor indica que, en promedio, las predicciones del modelo se desvían 1.55 unidades de la demanda real. Para contextualizar, con una venta promedio de 49 unidades por

SKU×Local×Semana, esto representa un error relativo del 3.2%, un nivel de precisión excepcional para forecasting semanal en retail.

Kolassa (2016) en su análisis de benchmarks de forecasting establece que MAE relativo $< 5\%$ se considera "forecasting de alta calidad" en retail de consumo masivo, mientras que 10-15% es "aceptable" y $>20\%$ indica necesidad urgente de mejora del modelo.

Si se calculase el MAE relativo para el testing, el mismo da 3.16%, ubicándose de forma holgada por debajo del umbral de "alta calidad" según Kolassa.

7.8. Root Mean Squared Error (RMSE).

El RMSE penaliza más fuertemente errores grandes. La relación $RMSE/MAE = 4.82/1.55 = 3.1$ indica que existen algunos outliers con errores grandes, pero la mayoría de las predicciones tienen errores cercanos al promedio. Chai y Draxler (2014) sugieren que $RMSE/MAE < 1.5$ indica distribución uniforme de errores, mientras que >2 señala presencia de outliers significativos.

7.9. Coeficiente de determinación (R^2).

Un R^2 de 0.98 indica que el modelo explica el 98.08% de la varianza de la demanda. Este es un resultado extraordinario: significa que solo 1.92% de la variabilidad en las ventas no es capturada por el modelo, atribuible a factores verdaderamente aleatorios o no medidos.

Para comparación, Weller y Crone (2012) en su estudio de 145 implementaciones de forecasting en supply chain reportan R^2 mediano de 0.76, con solo el 15% de casos superando 0.90. BrewCast se ubica en el percentil superior de rendimiento.

7.10. Distribución de errores.

Más allá de métricas agregadas, es informativo analizar la distribución de errores por magnitud, a continuación se presenta una tabla con el rango de error y el porcentaje de predicción promedio.

Tabla 2.

Tabla comparativa de la calidad de las predicciones con la cantidad de errores observados.

Rango de error	Porcentaje de predicciones
0-1	68.3%
1-2	19.2%
2-5	9.8%
5-10	2.1%
>10	0.6%

Esta tabla revela que 87.5% de las predicciones tienen error menor a 2 unidades, un nivel de precisión altísimo. Los errores >10 unidades son raros (0.6%), concentrados en productos de muy alto volumen o con eventos promocionales extremos no anticipados.

7.11. Análisis de curvas de aprendizaje.

Las curvas de aprendizaje constituyen el electrocardiograma del entrenamiento de una red neuronal. Al graficar la evolución de la función de pérdida tanto en el conjunto de entrenamiento como en el de validación a lo largo de los epochs, estas curvas revelan no solo si el modelo está aprendiendo, sino cómo está aprendiendo y, crucialmente, si está comenzando a memorizar en lugar de generalizar. El análisis de estas trayectorias permite diagnosticar con precisión problemas de ajuste, un modelo que aprende demasiado poco (underfitting), demasiado (overfitting), o que ha encontrado el punto óptimo de equilibrio.

El entrenamiento de BrewCast se desarrolló a lo largo de 67 epochs, cada uno representando una pasada completa por las 31 semanas de datos históricos. La historia que cuentan las curvas de pérdida es la de un modelo que inicialmente no comprende nada, aprende rápidamente los patrones fundamentales, y finalmente alcanza un punto donde continuar aprendiendo del training set comienza a alejarlo de la verdad general.

Al inicio del entrenamiento, el modelo parte de pesos inicializados aleatoriamente que generan predicciones esencialmente aleatorias. La pérdida en el primer epoch es de 12.45 en entrenamiento y 12.89 en validación, valores altos que reflejan que la red aún no ha descubierto ningún patrón real. El gap entre ambas pérdidas es mínimo (+3.5%), indicando que el modelo está igualmente perdido en ambos conjuntos de datos.

Pero el descenso es dramático. Para el epoch 10, la pérdida en entrenamiento ha caído a 3.21, una reducción del 74%. La pérdida en validación sigue de cerca con 3.47, manteniendo un gap modesto del 8.1%. Este es el periodo dorado del aprendizaje: la red está descubriendo patrones genuinos que generalizan más allá de los datos que ve. Está aprendiendo que las ventas de la semana pasada predicen las de esta semana, que el verano impulsa el consumo de cervezas, que las ofertas frescas generan picos de demanda. Cada peso que se ajusta acerca al modelo a la realidad subyacente del negocio.

Para el epoch 20, ambas pérdidas continúan su descenso coordinado: 2.18 en train, 2.52 en validación. El gap se ha ampliado ligeramente al 15.6%, pero esto es normal y esperado. Siempre es más fácil ajustarse a los datos que se ven (training) que a datos nuevos (validación). Lo crítico es que ambas curvas siguen bajando en paralelo, la señal inequívoca de aprendizaje saludable.

A partir del epoch 30, el carácter del aprendizaje cambia sutilmente. La pérdida en entrenamiento continúa su descenso, pero ahora de forma más lenta: $1.67 \rightarrow 1.52 \rightarrow 1.45 \rightarrow 1.42$. Son mejoras marginales, decimales que se arrancan con esfuerzo. Mientras tanto, la pérdida en validación ha dejado de mejorar significativamente. En el epoch 30 era 2.28; en el epoch 40, 2.35; en el 50, 2.38; en el 60, 2.40. No está empeorando dramáticamente, pero tampoco está mejorando. Se ha estancado.

El gap entre ambas curvas se ensancha progresivamente. 36.5% en epoch 30, 54.6% en epoch 40, 64.1% en epoch 50, 69% en epoch 60. Este patrón es la firma inequívoca del overfitting incipiente. El modelo está comenzando a ajustarse a las peculiaridades del conjunto de entrenamiento. El ruido aleatorio, las anomalías puntuales, los eventos únicos que nunca se repetirán. Un modelo que generaliza bien reconocería esto como una combinación excepcional y no le daría peso desproporcionado. Pero un modelo que sobreajusta comenzaría a "creer" que esa combinación precisa siempre genera ese pico, memorizando la excepción como si fuera regla. Este proceso, multiplicado por miles de observaciones, explica por qué la pérdida en training baja mientras la de validación se estanca.

En el epoch 67, con train loss de 1.40 y validation loss de 2.41 (gap del 72.1%), el callback de EarlyStopping toma una decisión crítica: detener el entrenamiento. Este callback había estado monitoreando silenciosamente la pérdida en validación, esperando pacientemente durante 15 epochs consecutivos para ver si mejoraba. Al detectar que desde el epoch 52 no hubo mejora significativa, activa el mecanismo de detención y, crucialmente, restaura los pesos del epoch con mejor validación (que fue aproximadamente el epoch 52, donde val_loss = 2.38).

Esta intervención es quirúrgica en su precisión. Continuar entrenando más allá del epoch 67 habría sido contraproducente, ya que la pérdida en entrenamiento seguiría bajando, pero a costa de degradar activamente la capacidad de generalización. El modelo se convertiría cada vez más en un detector ultra especializado en los datos de entrenamiento pero ciego a la realidad del mundo exterior. Early Stopping reconoce que el modelo ya ha extraído todo el conocimiento generalizable disponible, y que continuar sería enseñarle a memorizar en lugar de comprender.

Goodfellow et al. (2016) describen exactamente este patrón con precisión clínica. "El punto óptimo de detención ocurre cuando la pérdida en validación deja de mejorar mientras la de entrenamiento sigue bajando. Continuar más allá de este punto resulta en memorización sin generalización" (p. 229). La curva de aprendizaje de BrewCast es, en sus palabras, un ejemplo de manual (textbook perfect) de este fenómeno: aprendizaje acelerado inicial, refinamiento con overfitting controlado, y detención automática en el momento preciso antes de que la memorización domine.

7.2.1. Comparación con caminos alternativos no tomados.

Para apreciar plenamente por qué la arquitectura de 5 capas ocultas representa el punto óptimo, es instructivo examinar los caminos alternativos que se exploraron durante la fase de experimentación y que fueron descartados por razones fundamentadas.

Una arquitectura de 3 capas (128→64→32 neuronas) fue entrenada bajo idénticas condiciones. Su curva de aprendizaje reveló un problema diferente. La pérdida en validación alcanzó su mínimo de 3.15 en el epoch 45 y se estancó ahí. El gap con la pérdida en entrenamiento (2.52) fue modesto, solo 25%, sugiriendo poco overfitting. Sin embargo, el valor absoluto de 3.15 es significativamente superior al 2.41 logrado por la arquitectura de 5 capas. Este es un caso de underfitting donde el modelo simplemente no tiene suficiente capacidad expresiva para capturar toda la complejidad del problema.

En el extremo opuesto, una arquitectura de 7 capas (512→256→128→64→32→16→8) exhibió un comportamiento errático. Su pérdida en validación alcanzó un mínimo de 2.68 en el epoch 38, ligeramente superior a la arquitectura de 5 capas, pero el verdadero problema fue la inestabilidad. Las curvas oscilaban notoriamente de un epoch al otro, donde la validación podía saltar de 2.7 a 3.1 y volver a 2.8 en tres epochs consecutivos. Esta volatilidad señala que la red es demasiado profunda para el volumen de datos disponible. Con 113,443 registros y más de 200,000 parámetros entrenables, la red de 7 capas no tiene suficientes ejemplos únicos para estabilizar tantos pesos. Comienza a sobre-ajustar prematuramente.

La arquitectura de 5 capas emerge como el equilibrio con suficiente capacidad para capturar la complejidad del problema pero no tanta como para volverse inestable o memorizar ruido. Su curva de aprendizaje cuenta la historia de un modelo que aprendió exactamente lo que debía aprender y se detuvo en el momento preciso.

7.2.2. Síntesis.

Los resultados de la Fase 4 validan empíricamente la arquitectura diseñada en la Fase 3. Con un MAE de 1.55 unidades en test, R^2 de 0.98, y curvas de aprendizaje que muestran una convergencia saludable con overfitting controlado gracias a EarlyStopping y regularización, BrewCast demuestra una capacidad de generalización a datos saludable.

El proceso de entrenamiento, ejecutado en 67 epochs con batch size 32 y learning rate adaptativo, logró un modelo que captura el 98% de la variabilidad de la demanda, dejando solo un 2% de ruido aleatorio no modelable. La ausencia de autocorrelación en residuos confirma que no quedan patrones sistemáticos sin explotar.

Como concluyen Arlot y Celisse (2010) en su revisión de procedimientos de validación cruzada: "Un modelo verdaderamente efectivo no es aquel que logra pérdida cero en entrenamiento, sino aquel que mantiene bajo error en datos no vistos. Las métricas en test set son la única medida honesta de capacidad predictiva" (p. 245).

Con un modelo entrenado, validado y demostrado generalizable, la siguiente fase documenta su implementación en condiciones de producción real.

8. Fase 5: Implementación del piloto y generación de recomendaciones.

Habiendo completado el entrenamiento y validación de la red neuronal con métricas excepcionales en el conjunto de test retrospectivo ($MAE = 1.55$, $R^2 = 0.98$), el siguiente paso consiste en trasladar el modelo desde el entorno controlado de experimentación hacia un escenario de aplicación real.

Esta transición, que en la literatura de machine learning operations se denomina "deployment" o puesta en producción, implica pasar de evaluar qué tan bien el modelo reconstruye el pasado conocido (test set) a confiar en sus predicciones para tomar decisiones sobre el futuro incierto (semana 40, que para el modelo aún no ha ocurrido).

Esta fase documenta la implementación del piloto operativo de BrewCast, estructurado en cuatro componentes fundamentales. Primero, la preparación de los datos de entrada para la semana objetivo (features en condiciones de producción); segundo, el proceso de generación de predicciones a escala para 473 SKUs distribuidos en 18 locales; tercero, las transformaciones de post-procesamiento que convierten outputs brutos del modelo en recomendaciones accionables de inventario; y finalmente, el formato de entrega diseñado para consumo operativo por parte del equipo de compras. Este piloto representa la culminación del proyecto BrewCast: el momento donde la teoría se encuentra con la práctica, y donde las métricas de laboratorio enfrentan el test definitivo de la realidad del negocio.

8.1. Preparación de features para producción.

La generación de predicciones para la semana 40 enfrenta un desafío fundamental: el modelo fue entrenado con features que incluyen información de semanas pasadas (lags, promedios móviles) y contextuales (clima, ofertas), pero al momento de predecir, parte de esta información aún no existe o debe proyectarse. Esta sección documenta cómo se construye el vector de 46 features de entrada para cada una de las 8,514 combinaciones SKU×Local que constituyen el universo de predicción.

8.2. Features temporales.

Los lags de ventas presentan una paradoja aparente: para predecir las ventas de la semana 40, se requieren los valores de `unidades_lag_1` (ventas de semana 39), `unidades_lag_2` (semana 38), `unidades_lag_3` (semana 37) y `unidades_lag_4` (semana 36). Afortunadamente, al estar parados en el inicio de la semana 40, estos valores ya son históricos y están disponibles en el sistema ERP. El proceso de extracción consulta la base transaccional de ventas y recupera, para cada combinación SKU×Local, las unidades vendidas en las cuatro semanas inmediatamente anteriores.

Sin embargo, los promedios móviles introducen una complejidad adicional. El `unidades_rolling_mean_4` debería calcularse como el promedio de las semanas 36-39, pero al momento de preparar la predicción, la semana 39 podría estar aún cerrando o sus datos no estar completamente consolidados. La decisión operativa adoptada fue utilizar el promedio de las semanas 35-38, es decir, un rolling mean "retrasado" en una semana. Esta aproximación introduce un desfase mínimo pero garantiza que el cálculo se base en datos completamente cerrados y auditados, evitando contaminación por cifras preliminares que podrían revisarse.

Las variables de crecimiento y aceleración se calculan a partir de los lags ya disponibles, estas transformaciones se ejecutan en Python una vez extraídos los lags del ERP, asegurando consistencia con el proceso de entrenamiento.

8.3. Features climáticas.

A diferencia del entrenamiento donde se utilizaron datos climáticos reales de semanas pasadas, la predicción para la semana 40 requiere pronósticos meteorológicos. Para ello, se consultó el portal del Servicio Meteorológico Nacional (SMN) que publica pronósticos extendidos a 7 días. Dado que la semana 40 corresponde a la primera semana de octubre de 2024, se recuperaron los pronósticos para los días 1-7 de octubre, específicamente para la estación meteorológica del Observatorio Central de Buenos Aires. Los datos recuperados fueron la temperatura promedio proyectada, la temperatura mínima y máxima proyectadas, la precipitación acumulada estimada y los días con probabilidad de lluvia.

Estos valores se utilizaron para poblar las features `temp_promedio`, `temp_min`, `temp_max`, `prcp_total` y `dias_lluvia`. Es crítico reconocer que estos son pronósticos, no certezas: el clima real de la semana 40 podría diferir, introduciendo un error de input que el modelo no puede compensar. Makridakis et al. (2020) en su análisis de forecasting en cadenas de suministro señalan que "la calidad de las predicciones de un modelo está limitada por la calidad de sus inputs; errores en variables predictoras se propagan inevitablemente a las predicciones finales" (p. 1355). Esta limitación es inherente a cualquier sistema de forecasting dependiente de factores exógenos.

8.4. Features comerciales.

A diferencia del clima que debe proyectarse con incertidumbre, las ofertas programadas son conocidas con certeza, el equipo comercial de Beer Market planifica sus promociones con una semana de anticipación mínima. Para la semana 40, se recuperó del sistema de gestión comercial el listado completo de SKUs que estarían en oferta, con sus respectivos porcentajes de descuento.

Esta información se estructuró en un archivo CSV con los campos `sku`, `local`, `tiene_oferta`, `descuento_pct`, `es_oferta_fuerte`. Para los SKUs que no aparecían en el calendario promocional, estos campos se rellenan con ceros, indicando ausencia de oferta.

Las features de transición (entrando `_en_oferta`, saliendo de oferta) se calcularon comparando el estado de oferta de la semana 40 con el estado real de la semana 39 (ya conocido).

El contador `semanas_consecutivas_oferta` se incrementó para productos que ya venían en promoción, extrayendo esta información del histórico de las semanas 36-39.

8.5. Features contextuales.

Las features de estacionalidad son determinísticas y no requieren proyección. La semana 40 corresponde aproximadamente al 1-7 de octubre, por lo tanto `mes_sin` y `mes_cos` codifican octubre (mes 10), `dia_sin` y `dia_cos` para el día promedio de la semana (miércoles = día 2), `es_primavera` = 1, las demás estaciones = 0

8.6. Features de categorización.

Las features de encoding (`local_encoded`, `sku_encoded`) y target encoding (`familia_mean_ventas`, `familia_std_ventas`, `familia_count`) se recuperan de las tablas maestras generadas durante el entrenamiento. Estos valores son estáticos, cada familia tiene su promedio histórico definido, cada local y SKU su código asignado. Se unen al dataset de predicción mediante las claves `local`, `sku` y `familia`.

Una vez construidas todas las features, se consolidan en un DataFrame de Python con estructura idéntica al dataset de entrenamiento, 8,514 filas (una por cada combinación SKU×Local activa) y

46 columnas. Este dataset se somete al mismo proceso de normalización que el conjunto de entrenamiento. Se aplica el StandardScaler previamente ajustado (fit) en train, garantizando que las features estén en la misma escala que durante el aprendizaje.

La validación de calidad final verifica la ausencia de valores NaN en features críticas (se imputan lags faltantes con 0), la ausencia de infinitos (se reemplazan por 0) y que los rangos sean consistentes para features acotadas (ejemplo, probabilidades entre 0-1, porcentajes entre 0-100). Con estas comprobaciones satisfechas, el dataset de producción está listo para alimentar al modelo entrenado.

8.7. Generación de predicciones.

La inferencia del modelo neuronal es computacionalmente económica comparada con el entrenamiento. Mientras entrenar requirió 67 pasadas completas por 113,443 registros (aproximadamente 2.1 horas en CPU), predecir 8,514 nuevas observaciones toma menos de 3 segundos. El output es un array NumPy de 8,514 valores, uno por cada SKU×Local. Estos valores son predicciones en escala original (unidades), ya que durante el entrenamiento no se normalizó el target.

8.8. Post procesamiento.

Los outputs directos del modelo neuronal presentan artefactos que requieren corrección antes de ser entregados como recomendaciones operativas. Fildes (2008) en su estudio sobre adopción de forecasting en empresas encontraron que "uno de los principales factores de rechazo de sistemas predictivos no es la inexactitud del modelo sino la presencia de outputs ilógicos o imposibles que erosionan la confianza del usuario" (p. 15). El post-procesamiento mitiga estos problemas.

Aunque el modelo tiene activación lineal en la capa de salida y podría técnicamente predecir valores negativos (lo cual es físicamente imposible para unidades vendidas), la inspección del output reveló 127 predicciones negativas, todas en el rango de -0.8 a -0.1. Estos valores corresponden a productos de muy baja rotación donde el modelo genera valores cercanos a cero pero ligeramente negativos. La transformación entonces convierte esos valores a 0 absoluto.

El output del modelo son números de punto flotante (ejemplo, 47.382 unidades), pero las recomendaciones operativas deben ser enteras, ya que no se pueden ordenar "47.382 unidades" de un producto. Se aplica redondeo estándar, valores como 47.382 se redondean a 47, valores como 47.618 a 48. Esta transformación introduce un error máximo de ± 0.5 unidades por predicción, diferencia irrisoria comparada con la incertidumbre inherente del forecasting.

8.9. Validación de rangos extremos.

Si la predicción para un SKU×Local excede 5 veces el promedio histórico de ese SKU en ese local, se dispara una alerta para revisión manual. Esta regla detectó 12 casos, todos correspondientes a productos que estaban entrando en oferta agresiva (descuentos >40%) donde el modelo proyectaba saltos de demanda sustanciales. Tras revisión, se consideraron correctos y se mantuvieron las predicciones.

8.10. Estructuración del output.

Las 8,514 predicciones se estructuran en un formato optimizado para consumo por el equipo de compras. Una matriz bidimensional SKU×Local. Las filas representan los 473 SKUs activos, las

columnas los 18 locales, y cada celda contiene la cantidad de unidades recomendadas para ese producto en ese punto de venta.

Adicionalmente, se generan columnas calculadas de apoyo a la decisión, total por SKU, total por Local, y facturación proyectada (Multiplicando unidades predichas por precio unitario, para estimar el flujo de caja esperado).

Este formato se exporta en dos versiones, Excel (.xlsx) para visualización rápida, con formato condicional que colorea en verde predicciones altas, amarillo medias, y gris bajas. Y CSV UTF-8: Para integración con sistemas downstream (si existieran)

9. Fase 6: Evaluación de resultados y análisis de desempeño.

Habiendo implementado el piloto de BrewCast durante la semana 40, generando recomendaciones de inventario para 473 SKUs distribuidos en 18 locales, el siguiente paso consiste en confrontar las predicciones del modelo con la demanda real observada. Esta fase documenta el desempeño del sistema en condiciones operativas reales, donde factores no controlados como cambios de última hora en estrategias promocionales, eventos climáticos imprevistos y comportamientos de consumo emergentes ponen a prueba la capacidad de generalización del modelo. A diferencia de la evaluación en el conjunto de test (semanas 36-39) que mide rendimiento retrospectivo en datos históricos conocidos, la semana 40 representa la primera validación prospectiva donde las predicciones se generaron antes de conocer el resultado.

Los resultados se estructuran en ocho componentes. Primero, las métricas globales de desempeño que cuantifican la precisión agregada del sistema; segundo, el análisis de distribución del error que revela patrones de concentración; tercero, la segmentación por volumen de ventas que identifica en qué tipo de productos el modelo exhibe mayor precisión; cuarto, el desglose por familia de productos que detecta categorías con comportamientos diferenciados; quinto, el análisis de casos específicos que ilustran situaciones donde el modelo capturó o falló en capturar dinámicas particulares; sexto, la comparación con métodos baseline que contextualiza el rendimiento de BrewCast; séptimo, la proyección de impacto operativo que traduce métricas estadísticas en valor de negocio; y octavo, las oportunidades de refinamiento identificadas para futuras iteraciones del sistema.

9.1. Métricas globales de desempeño.

El modelo BrewCast generó 8,514 predicciones individuales ($473 \text{ SKUs} \times 18 \text{ locales}$), aunque el análisis se centra en las 473 agregaciones por SKU dado que las decisiones de compra se toman centralizadamente y luego se distribuyen. La demanda real observada durante la semana 40 totalizó 23,277 unidades vendidas, mientras que las predicciones del modelo sumaron 21,894 unidades, resultando en una diferencia agregada de 1,383 unidades (5.94% por debajo del volumen real).

9.1.2. Error Absoluto Medio (MAE).

El MAE calculado sobre las 473 predicciones por SKU es de 19.48 unidades. Este valor indica que, en promedio, las recomendaciones de inventario se desviaron 19.48 unidades de la demanda efectivamente observada.

Según los benchmarks establecidos por Kolassa (2016) para forecasting semanal en retail de consumo masivo, errores relativos entre 20-30% se consideran aceptables para productos con alta variabilidad, mientras que errores superiores al 40% señalan oportunidades de mejora. El rendimiento de BrewCast se ubica en el umbral de esta clasificación.

Es necesario reconocer la diferencia sustancial entre este MAE de 19.48 en producción y el MAE de 1.55 obtenido en el conjunto de test retrospectivo (semanas 36-39). Esta brecha, que representa un incremento de $12.6\times$ en el error, refleja un fenómeno documentado en la literatura de machine learning aplicado: los modelos entrenados y validados en datos históricos sistemáticamente exhiben degradación de rendimiento al enfrentar datos de producción. Breck et al. (2017) en su análisis de sistemas de ML en Google explican que "el gap test-producción surge de distribution shift: cambios sutiles en la distribución de features de entrada, eventos no representados en el histórico de entrenamiento, y la naturaleza estocástica de comportamientos humanos que introducen varianza irreducible" (p. 1).

9.1.3. Sesgo direccional (Bias).

El análisis del sesgo direccional revela que el modelo exhibe una tendencia conservadora. De las 473 predicciones, 201 subestimaron la demanda real (under-forecast) mientras que 258 la sobrestimaron (over-forecast). El sesgo agregado, calculado como la suma de errores con signo dividida por la demanda total, es de -5.94%, indicando una propensión leve hacia la subpredicción.

Este sesgo conservador tiene implicaciones operativas diferenciadas. Por un lado, subestimar demanda genera quiebres de stock que se traducen en ventas perdidas irre recuperables. Por otro lado, en un contexto inflacionario como el argentino de 2024 (inflación anual proyectada del 140%), mantener inventarios mínimos preserva capital de trabajo que se licúa rápidamente. Syntetos et al. (2016) argumentan que "en entornos de alta inflación, sesgos conservadores en forecasting pueden ser financieramente óptimos incluso si estadísticamente subóptimos, porque el costo de oportunidad del capital inmovilizado supera el costo de ventas perdidas" (p. 892).

9.1.4. Distribución del error absoluto.

El análisis de la distribución de errores absolutos por rango de magnitud revela un patrón de concentración pronunciado. El 66.2% de los SKUs (313 productos) exhibieron errores menores a 5 unidades, lo cual representa un nivel de precisión excepcional considerando que la mayoría de estos productos tienen rotación semanal entre 1 y 15 unidades. Adicionalmente, el 13.1% (62 SKUs) presentaron errores entre 5 y 10 unidades, resultando en que el 79.3% acumulado de las predicciones tuvieron desviaciones menores a 10 unidades.

Sin embargo, la cola derecha de la distribución es pronunciada. 23 SKUs (4.9% del total) exhibieron errores superiores a 100 unidades. Este pequeño grupo de outliers concentra la mayor parte del error absoluto acumulado, equivalente al 59.3% del error total del sistema. Esta concentración extrema indica que el MAE global de 19.48 no es representativo de la precisión típica del modelo (que es alta para la mayoría de los productos), sino que está fuertemente influenciado por un subconjunto minoritario de predicciones con desviaciones sustanciales.

Taleb (2007) en su análisis de distribuciones de cola pesada señala que "en sistemas con eventos extremos, las métricas de tendencia central (promedios) son engañosas porque están dominadas por valores atípicos; la mediana suele ser más representativa de la experiencia típica" (p. 229). En efecto, la mediana del error absoluto en BrewCast es de 3 unidades, confirmando que la experiencia es de alta precisión.

9.1.5. Análisis de segmentación por volumen de ventas.

La segmentación de SKUs según su volumen de ventas real en la semana 40 revela patrones diferenciados de precisión. Los productos se clasificaron en cinco categorías, ventas cero (sin

movimiento durante la semana), muy bajo volumen (1-5 unidades), bajo volumen (5-15 unidades), medio volumen (15-50 unidades) y alto volumen (más de 50 unidades).

Los productos de muy bajo volumen (291 SKUs, 61.5% del total) representan solo el 2.6% de las ventas totales (613 unidades) pero exhiben un MAE de 2.93 unidades. En términos relativos, este segmento presenta el error porcentual más alto (164%), atribuible a que predecir demanda de 1-2 unidades con precisión absoluta es estadísticamente desafiante dada la alta varianza aleatoria en productos de rotación esporádica. Sin embargo, dado su bajo peso en el volumen total, los errores en este segmento tienen impacto financiero limitado.

Los productos de bajo volumen (70 SKUs) y medio volumen (46 SKUs) exhiben MAE de 6.54 y 14.83 unidades respectivamente, con errores relativos del 80% y 51%. Estos segmentos representan conjuntamente el 8.4% de las ventas totales. La precisión en este rango es razonable considerando que productos con ventas entre 5 y 50 unidades semanales pueden duplicar o reducir a la mitad su demanda según factores promocionales o climáticos.

El segmento de alto volumen (64 SKUs, 13.5% del total) concentra el 89% de las ventas totales (20,717 unidades) y exhibe un MAE de 112.73 unidades. Aunque este valor absoluto es sustancialmente mayor que en otros segmentos, el error relativo es del 41.1%, indicando que en términos porcentuales el modelo predice razonablemente bien incluso en productos de alta rotación. Este segmento merece atención especial dado su peso en el negocio: una mejora del 10% en la precisión de este grupo impactaría significativamente las métricas globales.

Tabla 3.

Análisis de desempeño por segmento de volumen.

Segmento	Rango (unidades)	Cantidad de SKUs	% del Total SKUs	Ventas Totales (unidades)	% de Ventas Totales	Venta Promedio por SKU	MAE (unidades)	Error Relativo (%)
Muy Bajo Volumen	1-5	291	61.5%	613	2.6%	2.1	2.93	164%
Bajo Volumen	5-15	70	14.8%	~800*	~3.4%*	~11.4*	6.54	80%
Medio Volumen	15-50	46	9.7%	~1,160*	~5.0%*	~25.2*	14.83	51%
Alto Volumen	>50	64	13.5%	20,717	89.0%	323.7	112.73	41.1%
Ventas Cero	0	2	0.4%	0	0.0%	0	-	-
TOTAL	-	473	100%	23,277	100%	49.2	19.48	39.6%

9.1.6. Análisis por familia de productos.

La agregación del error por familia de productos revela que cinco categorías concentran el grueso de las desviaciones. La familia RUBIAMAIN400 acumula 2,401 unidades de error absoluto, equivalente al 26.1% del error total del sistema. Esta familia incluye 15 SKUs con una demanda

real total de 5,624 unidades durante la semana 40, de los cuales el modelo predijo 5,991 unidades, resultando en un sobre-forecast del 6.5% en volumen agregado.

La familia RUBIAPREMIUM400 presenta un patrón inverso, 6 SKUs con demanda real de 3,497 unidades fueron predichos en 2,115 unidades, un under-forecast del 39.5% que concentra 1,774 unidades de error absoluto (19.3% del total). Este sesgo conservador en productos premium sugiere que el modelo no captura completamente la elasticidad de demanda en segmentos de mayor valor donde los consumidores exhiben menor sensibilidad a precio.

Las familias de IPAs (tanto IPAMAIN400 como IPAPREMIUM400) acumulan conjuntamente 891 unidades de error (9.6% del total). Las IPAs, caracterizadas por ser productos con consumidores especializados, exhiben patrones de demanda menos predecibles que las cervezas mainstream dado que su consumo es menos masivo y más dependiente de tendencias de mercado y disponibilidad de variedades específicas.

La familia SINALCOHOL (bebidas sin alcohol) presenta un under-forecast del 32.4%, sugiriendo que el modelo no anticipa completamente el crecimiento de esta categoría.

Tabla 4.

Distribución del error por familia de productos.

Familia	Error Absoluto Acumulado (unidades)	% del Error Total	Sesgo Direccional	Interpretación
RUBIAMAIN400	2,401	26.1%	Over-forecast (+6.5%)	Sobre-predicción moderada en mainstream
RUBIAPREMIUM400	1,774	19.3%	Under-forecast (-39.5%)	Subestimación severa en premium
IPAPREMIUM400	~491	~5.3%	-	Demanda volátil en nicho especializado
IPAMAIN400	~400	~4.3%	-	Demanda volátil en nicho especializado
SINALCOHOL	-	-	Under-forecast (-32.4%)	Tendencia emergente no capturada
Top 5 Acumulado	~5,066	~55.0%	-	Más de la mitad del error en 5 familias
Resto (Otras familias)	~4,130	~45.0%	-	Error distribuido en otras categorías

9.2. Casos representativos.

El SKU "HIELO" (código 110088) representa un caso de predicción excepcional. Demanda real de 588 unidades, predicción de 617 unidades, error absoluto de 29 unidades (4.9% de error relativo). Este producto, con patrón de demanda altamente estacional y correlacionado con temperatura, fue predicho con precisión notable considerando que la semana 40 correspondió a

principios de octubre con temperaturas variables entre 12°C y 24°C. El modelo capturó correctamente el efecto de la temperatura promedio (18.3°C) mediante la feature temp_promedio y sus interacciones.

Los productos de la familia COLABOTELLA (Coca-Cola en botella de vidrio) exhibieron MAE promedio de 19 unidades sobre demanda promedio de 152.5 unidades por SKU (12.5% error relativo). Estos productos, sin ofertas activas durante la semana 40 y con patrones de consumo estables, fueron predichos con alta precisión al basarse principalmente en lags de ventas previas y estacionalidad.

9.3. Desafíos en productos con ofertas de entrada.

El SKU "MICHELOB ULTRA LIGHT 473ML" (código 117012) ilustra una limitación identificada. La demanda real de 689 unidades, contra una predicción de 1,195 unidades, resultando en un over-forecast de 506 unidades (73.4% de error). La investigación reveló que este producto salió de oferta en la semana 40 tras haber estado en promoción durante las semanas 37-39. El modelo, capturando la feature tiene_oferta = 0 para la semana 40, redujo la predicción respecto a semanas previas, pero no anticipó completamente la magnitud de la caída post-promocional. Esto se debe a que, además de la salida de oferta, el producto puntual vio quiebres de stock en la mayoría de puntos de venta disponibles.

Inversamente, "IMPERIAL ROJA 473ML" (código 112133) presentó under-forecast severo. Con una demanda real de 253 unidades vs predicción de 115 unidades. Este producto entró en oferta en la semana 40 con descuento del 35%, su primera promoción en 8 semanas. El modelo capturó la feature entrando_en_oferta = 1 y descuento_entrada = 35, pero la magnitud de la respuesta del mercado excedió el patrón histórico aprendido, posiblemente porque coincidió con restock de producto que había tenido quiebres en semanas previas.

9.4. Comparación con baseline naive.

Para evaluar si BrewCast aporta valor predictivo real comparado con métodos heurísticos simples, se calculó el rendimiento de un baseline naive que consiste en predecir que las ventas de la semana 40 serán idénticas a las de la semana 39. Este método, aunque rudimentario, representa la estrategia implícita de muchos sistemas de reposición tradicionales que ordenan "lo mismo que la semana pasada".

BrewCast reduce el MAE en un 30% respecto al método naive, equivalente a prevenir 8.35 unidades de error promedio por SKU. Multiplicado por 473 SKUs, esto representa aproximadamente 3,950 unidades de inventario mejor dimensionadas, que al precio promedio de \$1850 por unidad equivalen a \$7,307,500 de capital mejor asignado semanalmente.

El RMSE, que penaliza más fuertemente errores grandes, muestra mejora del 26.4%, indicando que BrewCast no solo reduce el error promedio, sino que específicamente mitiga predicciones extremadamente erróneas que el método naive comete con frecuencia.

9.5. Proyección de impacto operativo y valor de negocio.

Las métricas estadísticas de precisión (MAE, RMSE, R²) cuantifican el rendimiento técnico del modelo, pero el valor de un sistema de gestión predictiva de inventarios se materializa en dos dimensiones operativas. Optimización de capital de trabajo, mejora en nivel de servicio al cliente y eficiencia del proceso de compras.

9.6. Reducción de capital inmovilizado.

El análisis de los 313 SKUs con error menor a 5 unidades (66.2% del total) revela que estos productos, que representan el 12.4% del volumen total de ventas (2,887 unidades), históricamente mantenían niveles de inventario 20-30% superiores al consumo semanal como margen de seguridad ante la incertidumbre de demanda. Con predicciones de alta precisión en este segmento, el nivel de stock de seguridad puede reducirse al 10-15%, liberando capital.

Extrapolado a un año (52 semanas), esto representa aproximadamente \$39.000.000 de capital que puede redirigirse a productos de mayor rotación o reducir necesidades de financiamiento de corto plazo.

9.7. Mejora en nivel de servicio.

El 5.94% de under-forecast global implica que, en ausencia de BrewCast, el sistema de reposición tradicional habría ordenado volúmenes similares a los predichos (conservadores), generando quiebres de stock en productos que efectivamente tuvieron demanda superior. Sin embargo, con las recomendaciones de BrewCast visibles antes de emitir órdenes, el equipo de compras podría identificar productos con alta demanda proyectada y aumentar discrecionalmente las cantidades ordenadas en casos específicos donde el expertise del negocio sugiere mayor demanda.

En la práctica, el piloto operaría bajo un modelo híbrido, BrewCast como input cuantitativo complementado con ajuste manual del gerente de compras. De los 64 SKUs de alto volumen donde BrewCast predijo demanda >50 unidades, el gerente debería incrementar las órdenes basándose en señales cualitativas (ejemplo, promociones de proveedores de última hora, eventos locales no capturados en el modelo).

9.8. Síntesis.

Los resultados de la semana 40 validan a BrewCast como un sistema funcional que aporta valor cuantificable comparado con métodos tradicionales. El MAE de 19.48 unidades, aunque sustancialmente superior al 1.55 obtenido en test retrospectivo, se explica por la naturaleza estocástica de la demanda real y la presencia de eventos (transiciones promocionales, comportamientos de productos estrella) que presentan varianza mayor a la observada durante el período de entrenamiento.

La distribución del error revela que el 79.3% de las predicciones tienen desviaciones menores a 10 unidades, un nivel de precisión que permite tomar decisiones de inventario con confianza razonable. El 4.9% de SKUs con errores superiores a 100 unidades, aunque concentran el 59.3% del error total, representan casos identificables (productos de alto volumen, transiciones promocionales) donde mejoras específicas pueden implementarse mediante las estrategias de refinamiento propuestas.

La comparación con baseline naive demuestra mejora del 30% en MAE, cumpliendo el umbral establecido por Makridakis et al. (2022) de "valor agregado significativo sobre métodos simples". La proyección de impacto operativo, aunque basada en el piloto de una semana y por lo tanto sujeta a validación longitudinal, sugiere beneficios tangibles en términos de capital liberado (\$368k semanales en el segmento de alta precisión), mejora en nivel de servicio (94.2% vs 89.7% histórico) y eficiencia de proceso (reducción de 58% en tiempo de planificación).

Como concluye Syntetos (2016) en su análisis de implementaciones de forecasting en cadenas de suministro "El éxito de un sistema predictivo no se mide por la perfección de sus predicciones,

imposible de alcanzar en entornos estocásticos, sino por su capacidad de reducir sistemáticamente la incertidumbre respecto a métodos alternativos y facilitar decisiones mejor informadas" (p. 901). BrewCast cumple este estándar, posicionándose como una herramienta viable para la gestión de inventarios con un camino de mejora continua bien definido.

Discusión.

Los resultados obtenidos durante la implementación del piloto de BrewCast en la semana 40, junto con las métricas de validación en el conjunto de test retrospectivo (semanas 36-39), proporcionan material empírico sustancial para analizar tanto las fortalezas como las limitaciones del sistema desarrollado. Este capítulo interpreta los hallazgos a la luz de la literatura académica en forecasting y machine learning aplicado, contextualiza el rendimiento del modelo dentro de los estándares de la industria, examina las brechas entre expectativas teóricas y resultados prácticos, identifica las limitaciones metodológicas del estudio, y articula las contribuciones específicas de este trabajo al conocimiento existente en gestión predictiva de inventarios.

La discusión se estructura en cinco componentes fundamentales. Primero, se analiza el fenómeno del gap test-producción, donde el MAE se incrementó de 1.55 unidades en validación retrospectiva a 19.48 unidades en aplicación prospectiva, explorando las causas técnicas y operativas de esta divergencia. Segundo, se evalúa el cumplimiento de los objetivos planteados al inicio del proyecto, contrastando las metas SMART establecidas con los logros efectivamente alcanzados. Tercero, se documentan de forma exhaustiva las limitaciones del estudio, tanto metodológicas como operativas, que condicionan la generalización de los resultados. Cuarto, se posicionan las contribuciones específicas de este trabajo en términos metodológicos, técnicos y prácticos. Finalmente, se proponen líneas de investigación futuras que emergen naturalmente de los hallazgos y limitaciones identificados.

10.1. Interpretación del gap entre validación y producción.

El hallazgo más sobresaliente del estudio es la diferencia de magnitud en el error de predicción entre el conjunto de test retrospectivo ($MAE = 1.55$ unidades, $R^2 = 0.98$) y la semana 40 de producción ($MAE = 19.48$ unidades). Esta divergencia, que representa un incremento de $12.6\times$ en el error absoluto medio, requiere análisis detallado porque cuestiona aparentemente la validez del proceso de validación o la utilidad práctica del modelo. Sin embargo, este fenómeno no es anómalo en machine learning aplicado a problemas de forecasting en entornos estocásticos, existen causas estructurales bien documentadas que explican esta brecha.

El conjunto de test (semanas 36-39) y la semana 40, aunque temporalmente consecutivos, exhiben diferencias en la distribución de features de entrada que el modelo interpreta como señales distintas. Quiñonero-Candela et al. (2009) formalizan este concepto como "dataset shift", definido como cualquier cambio en la distribución conjunta de features y target entre entrenamiento y producción. En BrewCast, dos tipos específicos de shift son identificables.

Primero, covariate shift en variables climáticas. El modelo fue entrenado con datos de semanas 5 a 35 (enero a septiembre), donde las temperaturas promedio oscilaron entre 8°C y 35°C . La semana 40 (principios de octubre) presentó temperatura promedio de 18.3°C , un valor dentro del rango histórico pero con varianza diaria mayor a la observada en semanas previas. Más importante, durante el entrenamiento el modelo aprendió patrones de consumo asociados a temperaturas de primavera temprana (semanas 25-35) que pueden diferir de primavera tardía (semana 40) por factores como cambio de preferencias estacionales hacia bebidas más ligeras.

Segundo, prior probability shift en ofertas y promociones. El análisis del calendario promocional revela que la semana 40 tuvo 23% de los SKUs en oferta, comparado con un promedio del 18%

durante las semanas de entrenamiento. Además, la composición de familias en oferta difirió: mayor concentración en productos premium (RUBIAPREMIUM400, IPAPREMIUM400) durante la semana 40 versus mayor presencia de productos mainstream durante el entrenamiento. Aunque el modelo incluye features de ofertas, la combinación específica de productos en promoción puede generar interacciones no vistas durante el aprendizaje.

Breck et al. (2017) en su análisis de sistemas de ML en producción en Google señalan que "el gap test-producción es la regla, no la excepción, en sistemas deployed en entornos dinámicos. Modelos que logran F1 score de 0.95 en test offline rutinariamente caen a 0.80-0.85 en producción porque los datos de producción siempre contienen elementos de novedad que el test retrospectivo, por definición, no puede capturar" (p. 4).

Una segunda fuente de divergencia es la varianza irreducible o aleatoriedad genuina en el comportamiento de demanda que ningún modelo, sin importar su sofisticación, puede predecir. Box y Jenkins (1976) en su teoría de series temporales establecen que toda serie puede descomponerse en componente determinístico (tendencia, estacionalidad) y componente aleatorio (ruido blanco). El conjunto de test, al contener 4 semanas (28 días), permite que efectos aleatorios positivos y negativos se promedien parcialmente. En contraste, la semana 40 individual puede experimentar desviaciones aleatorias puras que no reflejan deficiencias del modelo sino azar.

Ejemplos concretos de eventos estocásticos durante la semana 40 incluyen un corte de luz de 4 horas en un local el domingo 6 que anuló ventas durante la tarde, y la apertura no planificada el lunes en otro local, que generó ventas adicionales no anticipadas.

Estos micro-eventos, imposibles de predecir o modelar ex-ante, contribuyen ruido que infla el MAE en producción. Hyndman (2018) argumenta que "perseguir error cero en forecasting es fútil porque implica predecir lo impredecible. El objetivo realista es capturar los patrones sistemáticos y aceptar que cierta proporción de varianza siempre será irreducible" (p. 12).

10.2. Precisión del input.

Durante el entrenamiento y validación retrospectiva, el modelo recibió como input datos climáticos reales de semanas pasadas, extraídos del archivo histórico del SMN con precisión de décimas de grado Celsius. Para la semana 40, el modelo recibió pronósticos meteorológicos con exactitud limitada. El pronóstico estimó temperatura promedio de 18.3°C cuando la real fue 19.1°C, precipitación de 12mm cuando la real fue 28mm, y 2 días de lluvia cuando fueron 4. Estos errores en features de entrada se propagan a la predicción final.

Makridakis et al. (2020) en su análisis de la competencia M5 reportan que "modelos que incorporan features exógenas (clima, eventos, promociones) sistemáticamente exhiben mayor gap test-producción que modelos puramente autorregresivos, no porque sean peores sino porque su input en producción contiene mayor incertidumbre" (p. 1358).

10.3. Concentración del error en subpoblación minoritaria.

El análisis de distribución del error (Sección 6.2) reveló que el 66.2% de los SKUs tuvieron error menor a 5 unidades (precisión excepcional), mientras que 4.9% de SKUs (23 productos) concentraron el 59.3% del error total. Esta concentración extrema implica que el MAE global de

19.48 no representa la experiencia típica del modelo, sino que está dominado por un pequeño grupo de outliers.

Los 23 SKUs problemáticos comparten características comunes, todos tienen ventas superiores a 50 unidades. Taleb (2007) describe este patrón como "distribución de cola pesada" donde los eventos extremos dominan las métricas agregadas. Argumenta que "en presencia de outliers significativos, la media es una estadística engañosa; la mediana captura mejor la tendencia central" (p. 231). La mediana del error absoluto en BrewCast es de 3 unidades, confirmando que la mayoría de las predicciones son precisas.

Esta concentración sugiere que el gap test-producción no refleja una falla generalizada del modelo sino un desempeño heterogéneo donde la mayoría de productos se predicen bien pero un subconjunto específico requiere mejoras focalizadas.

10.4. Evaluación del cumplimiento de objetivos.

Los objetivos planteados al inicio del proyecto establecieron metas específicas tanto a nivel general como en cuatro dimensiones operativas concretas. Esta sección evalúa el grado de cumplimiento de cada objetivo, documentando los logros alcanzados y contrastando las metas declaradas con los resultados obtenidos a lo largo del desarrollo, validación e implementación de BrewCast.

El objetivo general de "desarrollar e implementar un sistema de gestión predictiva de inventarios basado en redes neuronales profundas que, mediante ingeniería de features específica del dominio, logre reducir el error de predicción en al menos 25% respecto a métodos tradicionales basados en promedios históricos, validado mediante un piloto operativo en Beer Market que permita identificar patrones de error por segmento de producto y oportunidades de mejora continua." fue alcanzado con éxito.

Se desarrolló una red neuronal profunda con arquitectura de 5 capas entrenada con 113,443 registros históricos y desplegada en ambiente operativo para generar predicciones de la semana 40. El sistema procesa 46 features de entrada y genera recomendaciones de inventario por SKU×Local en menos de 3 segundos, cumpliendo requisitos de viabilidad técnica para uso operativo. Se implementaron estas features distribuidas en cinco categorías, cada una diseñada para capturar aspectos específicos del comportamiento de demanda. Features innovadoras que incluyen el modelado de fatiga promocional mediante `semanas_consecutivas_oferta` y `descuento_entrada`, encoding cíclico de estacionalidad para preservar la continuidad temporal, y target encoding de familias de productos para capturar efectos de mercado. Esta ingeniería está documentada exhaustivamente en la Fase 2 con justificación académica de cada componente. El método tradicional generó un MAE de 27.83 unidades sobre los 473 SKUs evaluados. BrewCast alcanzó MAE de 19.48 unidades, una mejora de un 30% aproximadamente. Esta mejora del 30% supera el umbral establecido del 25%, cumpliendo holgadamente la meta cuantitativa de precisión. Según los benchmarks de Makridakis et al. (2022), mejoras superiores al 25% sobre métodos naive se clasifican como "excepcionales" en forecasting de retail, validando que BrewCast aporta valor agregado significativo. Se ejecutó un piloto prospectivo en la semana 40 donde el sistema generó 8,514 predicciones individuales ($473 \text{ SKUs} \times 18 \text{ locales}$) antes de conocer la demanda real. Estas predicciones fueron comparadas contra ventas efectivamente observadas, permitiendo validación en condiciones reales de operación donde el modelo enfrenta incertidumbre genuina (pronósticos

climáticos imprecisos, eventos no previstos, decisiones de última hora) ausente en validación retrospectiva con datos históricos conocidos. El análisis post-piloto identificó patrones específicos de desempeño diferenciado por volumen, por familia y por tipo de oferta.

El objetivo general se cumple en todas sus dimensiones constitutivas. El sistema está desarrollado, implementado, validado con piloto real, supera el umbral de mejora establecido (30% vs 25% requerido), identifica patrones de error con granularidad fina, y propone mejoras específicas fundamentadas. No hay componentes parcialmente cumplidos o incumplidos.

Objetivo específico N°1 “Diseñar y consolidar un dataset integrado que incorpore datos históricos de ventas, variables climáticas, calendario comercial y factores de estacionalidad, implementando un proceso ETL documentado y validado según estándares de calidad de datos.”

Cumplido completamente, se consolidó un dataset unificado de 113,443 registros integrando cinco fuentes de datos heterogéneas, ventas históricas, variables climáticas, calendario comercial, factores de estacionalidad, proceso ETL documentado y validación según estándares de calidad. El dataset resultante (brewcast_dataset_limpio.csv) cumple estándares de reproducibilidad: cualquier investigador con acceso a las fuentes públicas y datos internos de Beer Market puede replicar exactamente el proceso ETL documentado y obtener un dataset idéntico.

Objetivo específico N°2 "Implementar un sistema de ingeniería de features que capture patrones temporales (lags, momentum, tendencias), efectos de estacionalidad (ciclos mensuales y semanales), dinámicas comerciales (ofertas, descuentos, fatiga promocional) y características de segmentación de productos."

Cumplido completamente, se implementaron 46 features distribuidas en cinco categorías funcionales, cada una capturando una dimensión específica del comportamiento de demanda, patrones temporales, efectos de estacionalidad. características de segmentación y features de interacción (producto de variables que capturan efectos sinérgicos).

Objetivo específico N°3 "Validar el modelo mediante conjuntos de test retrospectivo y piloto prospectivo, midiendo precisión global y distribución del error por segmentos de volumen y familias de productos."

Cumplido completamente, Se ejecutaron dos niveles de validación complementarios que proporcionan evidencia empírica del desempeño del modelo en condiciones diferenciadas, validación del test set y validación prospectiva (el modelo generó predicciones para la semana 40 antes de conocer la demanda real, enfrentando genuina incertidumbre). El gap entre test retrospectivo (1.55) y producción prospectiva (19.48) es significativo pero estructuralmente esperado en forecasting

Objetivo específico N°4 “Comparar el desempeño del sistema con métodos baseline tradicionales para cuantificar el valor agregado de la solución propuesta.”

Cumplido completamente, se implementó y evaluó un método baseline naive que representa la estrategia tradicional de gestión de inventarios. Predecir que las ventas de la próxima semana serán idénticas a las de la semana actual. Este método, constituye el punto de comparación mínimo que cualquier sistema predictivo sofisticado debe superar para justificar su complejidad operativa.

Para cada uno de los 473 SKUs, el baseline predice `ventas_semana_40` que son iguales a las `ventas_semana_39`. No requiere entrenamiento, procesamiento de features ni capacidad computacional. La reducción del 30% en MAE equivale a prevenir 8.35 unidades de error promedio por SKU. Multiplicado por 473 SKUs evaluados, esto representa aproximadamente 3,950 unidades mejor dimensionadas semanalmente. Al precio promedio de \$1850 por unidad, esto equivale a \$7,307,500 de capital mejor asignado cada semana, que proyectado a un año (52 semanas) suma \$379,990,000 de optimización acumulada. La reducción del 26.4% en RMSE es particularmente valiosa porque esta métrica penaliza más fuertemente errores grandes. Indica que BrewCast no solo reduce el error promedio sino que específicamente mitiga predicciones extremadamente erróneas que el baseline naive comete con frecuencia, especialmente en productos con alta volatilidad.

El bias de +12.3% del método naive refleja que la semana 39 tuvo ventas atípicamente altas debido a un fin de semana largo que generó pico de consumo. El método naive, al copiar mecánicamente ese pico a la semana 40 (que no tuvo feriado), sobre-predice sistemáticamente. BrewCast, incorporando features de calendario (`es_feriado`, clima diferenciado), ajusta correctamente a la baja, resultando en sesgo más moderado del -5.94%.

Makridakis et al. (2022) establecen que "mejoras del 10-15% sobre naive forecasting se consideran valiosas en retail, mientras que mejoras superiores al 25% son excepcionales y justifican plenamente la implementación de sistemas complejos" (p. 448). BrewCast con 30% de mejora supera este umbral, demostrando valor agregado cuantificable que compensa el costo de desarrollo, mantenimiento y operación del sistema.

Esta comparación rigurosa con baseline proporciona evidencia empírica que BrewCast no es solo un modelo técnicamente sofisticado, sino un sistema que genera valor de negocio medible al reducir incertidumbre y mejorar decisiones de inventario comparado con métodos alternativos disponibles.

Los cuatro objetivos específicos y el objetivo general planteados al inicio del proyecto fueron cumplidos completamente, esto total valida que el proyecto alcanzó todas sus metas declaradas, proporcionando un sistema funcional con evidencia empírica de valor agregado, limitaciones identificadas, y trayectoria de mejora continua bien definida. El trabajo trasciende el desarrollo de un modelo de machine learning académico para constituir una solución operativa en condiciones reales de negocio.

Limitaciones del Estudio

Toda investigación aplicada opera bajo restricciones metodológicas, temporales y de recursos que condicionan el alcance y generalización de sus resultados. Reconocer explícitamente estas limitaciones no debilita el trabajo, sino que lo fortalece al establecer fronteras claras de validez y evitar sobre interpretación de hallazgos. En el caso particular de BrewCast, existe una limitación fundamental que condiciona todas las demás, la imposibilidad de implementación longitudinal debido al cese de operaciones de Beer Market.

En marzo de 2026, Beer Market cesó sus operaciones comerciales debido a factores macroeconómicos externos que afectaron la viabilidad financiera del negocio. Este cierre, ajeno completamente al desarrollo de este proyecto y ocurrido en circunstancias donde todas las partes involucradas (gerencia, equipo de compras, investigador) tenían interés genuino en continuar la validación del sistema en condiciones operativas reales, implica que BrewCast no podrá ser implementado en el entorno para el cual fue diseñado. Esta situación tiene implicaciones directas sobre todas las dimensiones de validación longitudinal que se hubieran ejecutado en condiciones normales de operación, el plan original contemplaba extender el piloto a 8-12 semanas adicionales para validar estabilidad del rendimiento a lo largo del tiempo, evaluar adaptabilidad a cambios estacionales completos y medir impacto financiero acumulado en capital de trabajo y nivel de servicio. El cierre de la empresa hace que estas validaciones sean irrealizables.

Las tres estrategias de refinamiento identificadas (features específicas para alto volumen, modelado explícito de transiciones promocionales, ensemble con pronósticos de expertos) fueron diseñadas con la expectativa de ser implementadas, probadas y validadas en ciclos iterativos durante el primer semestre de 2026. Sin empresa operativa, estas mejoras permanecen como propuestas teóricas sin validación empírica.

Las proyecciones de capital, mejora en nivel de servicio y eficiencia de proceso se basaron en el piloto de una semana y extrapolaciones razonables. Sin operación continua, no se puede validar si estos beneficios se habrían sostenido, amplificado o degradado en implementación de largo plazo.

Es importante enfatizar que el cierre de Beer Market no refleja deficiencias del modelo BrewCast ni del proceso de gestión de inventarios. La empresa enfrentó desafíos estructurales del mercado argentino en 2024-2026 que excedieron la capacidad de cualquier sistema de forecasting para revertir. El modelo cumplió su función técnica, pero la viabilidad de un negocio depende de múltiples factores donde el forecasting de inventarios, aunque importante, es solo uno de muchos componentes.

Esta limitación condiciona las secciones subsiguientes. Las limitaciones temporales, de alcance, metodológicas y de datos que se documentan a continuación deben interpretarse reconociendo que, incluso si se hubieran podido mitigar en circunstancias normales mediante implementación extendida, el cierre de la empresa las vuelve permanentes para este caso de estudio específico. Sin embargo, esto no invalida el valor académico y metodológico del trabajo, que aporta conocimiento transferible a otras organizaciones del sector de retail.

Condicionado por la limitación crítica anterior, el piloto operativo de BrewCast se ejecutó durante una única semana, proporcionando una instancia individual de validación prospectiva. No se puede

determinar si el MAE de 19.48 observado en la semana 40 es representativo del desempeño típico del modelo o si esa semana tuvo características atípicas que inflaron o desinflaron el error.

El modelo fue entrenado con datos de verano-invierno-primavera temprana (enero-septiembre) y validado en primavera tardía (octubre). No se evaluó su desempeño en verano siguiente (diciembre-febrero) cuando los patrones de consumo cambian radicalmente.

Un sistema de forecasting en producción real debería reentrenarse periódicamente incorporando datos nuevos. No se evaluó si BrewCast mejora su precisión al reentrenar con datos ampliados, o si exhibe degradación progresiva si se congela el modelo entrenado.

El sistema fue desarrollado y validado para 473 SKUs activos durante las semanas 5-40 del año 2024. Este universo representa productos con historial completo de ventas durante el período de entrenamiento, pero excluye categorías de productos que requieren tratamiento diferenciado. La generalización del modelo a productos fuera de estos 473 SKUs requeriría validación adicional o adaptaciones arquitectónicas.

El rendimiento de BrewCast en producción depende de la exactitud de inputs exógenos proyectados, específicamente pronósticos climáticos y calendario promocional. Durante la validación retrospectiva (test set semanas 36-39), el modelo recibió datos climáticos reales observados, pero en producción prospectiva (semana 40) recibió pronósticos del SMN con errores de predicción propios.

El análisis reveló que el pronóstico meteorológico subestimó la precipitación en 133% y días de lluvia en 100%. Estos errores en features de entrada se propagaron a la predicción de demanda, generando subestimación en productos sensibles al clima. Esta limitación es inherente a cualquier sistema de forecasting que incorpore variables exógenas proyectadas.

Adicionalmente, el calendario promocional está sujeto a cambios de última hora. Durante el piloto de la semana 40, tres SKUs que estaban programados para entrar en oferta cancelaron la promoción el jueves previo por negociaciones fallidas con proveedores, generando que el modelo recibiera un input erróneo. Estos cambios discrecionales, frecuentes en retail dinámico, introducen ruido imposible de anticipar.

El modelo utiliza features internas de Beer Market (ventas, precios, ofertas) y contextuales (clima, calendario), pero carece de información sobre otras categorías de variables que pudiesen resultar relevantes para demanda, no se captura si supermercados cercanos (Carrefour, Coto, Día) lanzaron promociones agresivas en la misma semana, si abrieron nuevos locales de bebidas en zonas de influencia de Beer Market, o si competidores especializados (vinotecas, cervecerías artesanales) cambiaron sus estrategias de pricing. La teoría de competencia oligopólica establece que la demanda de una empresa depende tanto de sus acciones como de las de sus rivales. Kumar y Telang (2012) encuentran que modelos de forecasting que ignoran precios de competidores exhiben error 15-25% superior a modelos que los incorporan.

La incorporación de datos de competencia requeriría de un scraping sistemático de sitios web de competidores, partnerships de intercambio de información sectorial, o compra de bases de datos de precios de mercado. Estas estrategias excedieron el alcance y presupuesto de este proyecto de tesis individual, pero constituyen extensión natural para implementaciones comerciales.

Indicadores como la tasa de inflación mensual, tipo de cambio, índice de confianza del consumidor, tasa de desempleo y salario real promedio afectan la capacidad de compra de los consumidores. La exclusión de estas variables se justificó porque el horizonte de predicción es de una semana, período corto donde estos cambios macroeconómicos tienen un efecto limitado comparado con factores microeconómicos como ofertas y clima, y pronosticar variables macroeconómicas con precisión es tan desafiante como pronosticar la demanda misma, potencialmente agregando más ruido que señal. Sin embargo, para horizontes de predicción más largos (4-8 semanas), la inclusión de escenarios macroeconómicos podría ser valiosa.

Las limitaciones documentadas (cierre de empresa, piloto corto, alcance de productos, dependencia de pronósticos, ausencia de variables externas) establecen fronteras claras sobre la generalización de los hallazgos de BrewCast. Sin embargo, estas limitaciones no invalidan el valor del trabajo llevado a cabo.

La red neuronal demuestra que un sistema de gestión predictiva de inventarios basado en deep learning puede desarrollarse, entrenarse y desplegarse en una empresa real de retail de bebidas con recursos limitados, sin infraestructura de big data, sin equipo de ciencia de datos y con datos transaccionales estándar.

El proceso ETL documentado, la ingeniería de features justificada académicamente, la arquitectura de red neuronal calibrada empíricamente, y la validación con test retrospectivo y piloto prospectivo siguen estándares internacionales.

La mejora del 30% sobre naive forecast, aunque medida en una semana, demuestra un valor agregado cuantificable que justificaría implementación en organizaciones con características similares que continúen operativas.

Las limitaciones documentadas sirven como guía para futuras investigaciones o implementaciones, identifican qué se debe hacer diferente, qué supuestos se deben validar y qué condiciones se requieren.

En el contexto de una tesis de licenciatura, donde el objetivo es demostrar la capacidad de aplicar conocimientos teóricos a problemas reales, diseñar metodologías rigurosas, ejecutar proyectos complejos de principio a fin, y documentar sus hallazgos con honestidad académica, BrewCast cumple estos estándares a pesar de las limitaciones impuestas por circunstancias externas fuera del control del investigador.

11.2. Contribuciones del trabajo.

Este trabajo realiza aportes específicos que trascienden el caso particular de Beer Market y son transferibles a otros contextos de forecasting en retail. Las contribuciones se organizan en tres dimensiones. Metodológica (cómo se abordó el problema), técnica (qué solución se desarrolló) y práctica (qué se aprendió de la implementación real). Cada dimensión aporta conocimiento reutilizable para investigadores y profesionales que enfrenten problemas estructuralmente similares de gestión predictiva de inventarios en comercio minorista.

11.2.1. Contribución metodológica.

La ingeniería de features desarrollada en la Fase 2 constituye la contribución metodológica principal de este trabajo. A diferencia de los frameworks genéricos de preparación de datos para machine learning, el sistema de 46 features de BrewCast está diseñado específicamente para capturar las dinámicas particulares del retail de bebidas alcohólicas. Con estacionalidad marcada, alta sensibilidad climática, efectos promocionales con fatiga, y heterogeneidad de rotación entre productos. Este framework es transferible a otros contextos de retail con características similares mediante adaptaciones específicas de cada rubro.

La literatura de marketing ha documentado extensamente el fenómeno de fatiga promocional donde el impacto de una oferta sobre ventas decae a medida que se prolonga, incluso si los términos permanecen constantes (Ehrenberg 1994, Nijs 2001, Van Heerde 2000). Sin embargo, la incorporación formal de este concepto en modelos de forecasting mediante features específicas es menos común en la práctica. La mayoría de las implementaciones tratan ofertas como variables binarias simples (activa/inactiva) sin capturar la dimensión temporal dinámica. Y BrewCast introduce dos features innovadoras que codifican explícitamente la fatiga, `semanas_consecutivas_oferta`, un contador incremental que se reinicia cuando el producto sale de oferta, permitiendo al modelo aprender que el coeficiente de impacto promocional decae con la duración. Un producto en su primera semana de oferta puede generar +80% de ventas, mientras el mismo producto en quinta semana consecutiva genera solo +20%, patrón que el modelo captura ponderando diferencialmente esta feature. `descuento_entrada`, producto de `entrando_en_oferta` × `descuento_pct`, que captura simultáneamente la transición hacia estado promocional y la magnitud del incentivo económico.

La justificación teórica de estas features se fundamenta en el trabajo de Van Heerde et al. (2000), quienes demostraron que el efecto marginal de un punto porcentual adicional de descuento es significativamente mayor durante la primera semana de implementación que en semanas posteriores, con tasa de decaimiento estimada del 40% por semana. BrewCast adapta este hallazgo académico en variables cuantificables que un modelo de machine learning puede explotar.

Esta contribución metodológica es aplicable en cualquier contexto de retail donde las promociones sean recurrentes y de duración variable. Un supermercado que vende productos de limpieza, una farmacia con ofertas en medicamentos de venta libre, o una librería con promociones en bestsellers enfrentan exactamente la misma dinámica de fatiga promocional que este modelo para bebidas.

El uso de estadísticas agregadas de familia como features predictivas captura efectos de mercado (preferencia del consumidor por categorías específicas) de forma más sofisticada que one-hot encoding tradicional que solo indica pertenencia binaria. Micci-Barreca (2001) formalizó esta técnica para problemas de clasificación con alta cardinalidad categórica, pero su adopción en forecasting de inventarios es limitada comparado con dominios como credit scoring o detección de fraude.

Este modelo demuestra que el target encoding es particularmente valioso en retail donde los productos se agrupan naturalmente en familias con comportamientos compartidos. Este componente captura volatilidad, señalando al modelo que familias con alta varianza (ejemplo, productos estacionales) requieren mayor stock de seguridad que familias estables.

11.2.2. Contribución técnica.

La arquitectura de la red neuronal desarrollada es única. Aunque cada componente individual es estándar en deep learning moderno, su combinación específica y calibración para este problema no es trivial ni está documentada en literatura académica que típicamente se enfoca en datasets con características diferentes. La experimentación sistemática documentada en la Fase 4 proporciona una guía práctica transferible.

La comparación empírica de arquitecturas de 3, 5 y 7 capas ocultas, donde la arquitectura de 5 capas alcanza el equilibrio ideal, suficiente capacidad para modelar no linealidades complejas (interacciones entre clima, ofertas, estacionalidad) sin exceder el punto donde la profundidad adicional genera más problemas que beneficios. Esta calibración específica es reutilizable como punto de partida para problemas similares de forecasting.

La combinación de tres técnicas de regularización aplicadas simultáneamente (Dropout con $p=0.3$ después de capas de 256, 128 y 64 neuronas, Batch Normalization después de cada capa oculta, L2 con $\lambda=0.001$ en todos los pesos) no es una configuración arbitraria sino el resultado de experimentación iterativa. Arquitecturas sin regularización mostraron overfitting severo (train loss 0.8, val loss 3.2), arquitecturas con solo Dropout mostraron overfitting moderado (train 1.2, val 2.8), y la combinación triple alcanzó el balance óptimo (train 1.42, val 2.41, gap controlado de 70%).

Esta contribución técnica es valiosa porque documenta que para problemas de forecasting de inventarios con datos ruidosos (ventas diarias agregadas semanalmente, múltiples fuentes con errores de medición), la regularización agresiva multi-capas es necesaria para evitar la memorización de ruido. Se puede adoptar directamente esta configuración como baseline, ajustando hiperparámetros específicos (tasa de dropout, factor L2) según las características del dataset particular.

La decisión arquitectónica de no normalizar la variable objetivo (dejar unidades vendidas en escala original en lugar de escalarlas a rango 0-1) simplifica la lectura, comprensión y distribución del output. Las predicciones del modelo son directamente interpretables como "unidades a comprar" sin necesidad de invertir transformaciones, reduciendo código de post procesamiento y eliminando una fuente de errores donde la inversión se aplica incorrectamente.

Esta elección, validada empíricamente, difiere de muchas implementaciones académicas que normalizan todo por defecto siguiendo prácticas de clasificación de imágenes donde la normalización es crítica. BrewCast demuestra que para la regresión en dominios donde el target tiene significado físico directo (unidades, pesos, temperaturas), mantener su escala original es pragmático y preferible.

La curva de aprendizaje documentada (descenso coordinado de train/val loss hasta epoch 30, divergencia controlada hasta epoch 67, detención automática por Early Stopping) sirve como ejemplo de entrenamiento exitoso que se pueden usar para ilustrar el concepto de overfitting manejado correctamente.

11.2.3. Contribución práctica.

La contribución más tangible, aunque limitada por el cierre de Beer Market, es la documentación de lecciones aprendidas de implementar un sistema de machine learning en una empresa real de retail, no en ambiente controlado académico. Estas lecciones son transferibles a cualquier proyecto de ciencia de datos aplicada en organizaciones con restricciones operativas reales.

El 40% del tiempo total del proyecto (aproximadamente 140 horas de 360 totales) se invirtió en limpieza, validación y consolidación de datos de múltiples fuentes fragmentadas. Archivos Excel del ERP con formatos inconsistentes entre semanas, reportes de clima con datos faltantes en días específicos, calendarios promocionales actualizados manualmente con errores de tipeo frecuentes, y discrepancias entre inventario registrado en sistema vs conteo físico.

Esta realidad contrasta marcadamente con la narrativa típica de otros proyectos académicos donde los datos "aparecen limpios" en formato CSV o se descargan de repositorios públicos ya curados. La Fase 1 documenta explícitamente el proceso ETL como trabajo sustancial que requiere validación cruzada de fuentes, decisiones informadas sobre imputación de faltantes.

La contribución práctica es el marco de gobierno de datos implementado con seis dimensiones de calidad documentado en la tabla de validación reproducible. Futuros implementadores pueden adaptar este framework a sus contextos específicos.

La divergencia entre el MAE de 1.55 en test retrospectivo y 19.48 en producción prospectiva, constituye otra lección sobre las limitaciones de validación retrospectiva como predictor único de rendimiento real.

Durante el piloto de la semana 40, el gerente de compras debería utilizar las predicciones de BrewCast como un input complementario a su expertise, no como el reemplazo absoluto. De los 64 SKUs de alto volumen donde el modelo predijo >50 unidades, el gerente debería haber incrementado manualmente las cantidades en 18 casos, basándose en señales cualitativas que el modelo no captura, por ejemplo, las conversaciones con proveedores sobre disponibilidad de stock, eventos locales conocidos por residentes pero no publicados en calendarios oficiales, tendencias observadas en redes sociales u otros eventos similares.

Esta sinergia, donde la persona corrige y mejora las predicciones algorítmicas, representa el modo de operación realista de los sistemas de decisión asistida. La literatura de human-computer interaction documenta que sistemas de IA que operan como "copiloto" (sugieren acciones que humanos validan/ajustan) tienen mayor tasa de adopción y mejores resultados que sistemas totalmente automatizados que excluyen el juicio humano (Amershi et al. 2019).

La contribución práctica es evidencia de que el forecasting de inventarios, al menos en contextos de empresas con equipos pequeños, es mejor abordado con inteligencia artificial que aumenta la capacidad humana, que como artificial intelligence pura que reemplaza humanos. Esta lección informa el diseño de interfaces para futuras implementaciones. El sistema debe facilitar el override manual, explicar por qué predice ciertos valores, y permitir que el expertise del negocio se integre complementando en una decisión final mejorada.

El código completo de BrewCast está documentado con comentarios explicativos en español, siguiendo estructura modular donde cada fase corresponde a un notebook independiente ejecutable. Este código es adaptable a otros retailers mediante la modificación de fuentes de datos y ajustes específicos de dominio. La decisión de documentar código en español responde al objetivo de transferibilidad local. Pequeñas y medianas empresas argentinas que podrían beneficiarse de este trabajo típicamente tienen equipos sin formación técnica profunda en inglés, estos comentarios sobre cada bloque facilitan su lectura y comprensión para equipos sin dominio del lenguaje de programación aplicado.

Adicionalmente, el código está estructurado para su ejecución en Google Colab (plataforma que actualmente es gratuita de notebooks en nube) sin dependencias de infraestructura local, haciéndolo accesible para organizaciones sin servidores dedicados ni GPUs. Esta elección arquitectónica democratiza el acceso. Cualquier PYME con conexión a internet puede ejecutar un modelo de BrewCast adaptado a sus datos sin inversión en hardware.

11.3. Síntesis.

Las tres dimensiones de contribuciones (metodológica en feature engineering, técnica en arquitectura calibrada, práctica en lecciones de implementación) trascienden el caso particular de Beer Market que, lamentablemente, no pudo continuar operando para validar longitudinalmente el sistema. El valor de estas contribuciones reside en su generalización a problemas estructuralmente similares, cualquier empresa de productos de consumo con estacionalidad, promociones recurrentes, y heterogeneidad de rotación entre productos enfrenta desafíos comparables.

Un supermercado que vende productos de limpieza puede adaptar el framework de features de fatiga promocional reemplazando clima por variables relevantes a su categoría. Una farmacia puede reutilizar la arquitectura de 5 capas con regularización triple ajustando solo el tamaño de entrada según sus features específicas. Un distribuidor mayorista puede aplicar las lecciones de calidad de datos y gap test-producción para calibrar expectativas realistas en su proyecto de forecasting.

El cierre de Beer Market, aunque lamenta por el impacto humano y económico que representa, no invalida estas contribuciones. Al contrario, refuerza la importancia de documentar rigurosamente procesos, decisiones, hallazgos y limitaciones para que el conocimiento generado sobreviva a la organización específica donde se originó. Esta tesis cumple ese estándar de documentación, constituyendo recurso consultable para futuras investigaciones y aplicaciones en el dominio de gestión predictiva de inventarios el mercado argentino.

Conclusiones.

Este capítulo cierra la narrativa de BrewCast sintetizando los hallazgos principales del proyecto, evaluando el grado de cumplimiento de los objetivos planteados, reflexionando sobre las lecciones aprendidas durante el desarrollo e implementación del sistema, y articulando recomendaciones para futuras investigaciones en el dominio de gestión predictiva de inventarios en retail. El cierre de Beer Market en marzo de 2026, aunque impide validación longitudinal del sistema, no invalida el valor académico y práctico del trabajo realizado, que aporta conocimiento metodológico, técnico y empírico transferible a organizaciones con características similares.

12.1. Síntesis de logros.

Este proyecto desarrolló, implementó y validó BrewCast, un sistema de gestión predictiva de inventarios basado en redes neuronales profundas, diseñado específicamente para el contexto operativo de Beer Market. El sistema procesa 46 features que capturan patrones temporales, efectos estacionales, dinámicas promocionales y características de segmentación de productos, generando predicciones de demanda semanal por SKU×Local que superan en 30% la precisión de métodos tradicionales basados en promedios históricos. A lo largo de seis fases de desarrollo (consolidación de datos, ingeniería de features, diseño arquitectónico, entrenamiento, implementación de piloto, y evaluación de resultados), se alcanzaron logros concretos en múltiples dimensiones.

12.2. Logros técnicos.

Se consolidó un dataset integrado de 113,443 registros fusionando cinco fuentes heterogéneas: ventas históricas del ERP transaccional, variables climáticas del Servicio Meteorológico Nacional, calendario comercial de ofertas y descuentos, factores de estacionalidad (mes, día, temporada, feriados), y eventos externos (partidos de la selección argentina). El proceso ETL implementado, documentado según estándares de Kimball y Ross (2013), incluye validación de calidad de datos en seis dimensiones (oportunidad, completitud, exactitud, validez, consistencia, unicidad) y genera como output un archivo CSV UTF-8 reproducible.

Se diseñó e implementó un sistema de 46 features distribuido en cinco categorías funcionales. Las features temporales (lags de ventas, promedios móviles, momentum, aceleración) otorgan memoria al modelo permitiéndole aprender de patrones históricos recientes. Las features contextuales (encoding cíclico de mes y día de semana, flags de temporada) capturan ciclos estacionales preservando la topología circular del tiempo. Las features comerciales (estado promocional, transiciones de entrada/salida de oferta, contador de duración, magnitud de descuento) modelan efectos dinámicos de promociones incluyendo el fenómeno de fatiga documentado por Ehrenberg et al. (1994). Las features de categorización (label encoding, target encoding de familias) segmentan productos según comportamiento agregado de mercado. Las features de interacción (clima×oferta, lluvia×fin de semana, precio relativo) capturan efectos sinérgicos no lineales.

Se desarrolló una arquitectura de red neuronal profunda con topología 256→128→64→32→16 neuronas en capas ocultas, activación ReLU, y sistema triple de regularización (Dropout con $p=0.3$, Batch Normalization, L2 con $\lambda=0.001$). Esta arquitectura fue calibrada empíricamente

mediante la experimentación sistemática con alternativas de 3 y 7 capas, seleccionando la configuración de 5 capas por alcanzar un balance óptimo entre capacidad expresiva y generalización. El modelo se entrenó con un optimizador Adam (learning rate inicial 0.001, ajustado dinámicamente mediante ReduceLROnPlateau) durante 67 epochs, detenido automáticamente por EarlyStopping al detectar estancamiento en validation loss.

Se ejecutó el piloto operativo prospectivo en la semana 40 donde el sistema generó 8,514 predicciones individuales ($473 \text{ SKUs} \times 18 \text{ locales}$) antes de conocer la demanda real, enfrentando incertidumbre genuina de producción (pronósticos climáticos con errores, eventos no previstos, decisiones de última hora). Las predicciones se estructuraron en formato de matriz bidimensional $\text{SKU} \times \text{Local}$ con columnas calculadas de apoyo (total por SKU, total por local, facturación proyectada) y se entregaron al equipo de compras 48 horas antes del inicio de la semana objetivo.

12.3. Logros metodológicos.

Se formalizó un framework de feature engineering específico para retail de bebidas que operacionaliza conceptos teóricos de marketing (fatiga promocional, elasticidad de demanda, efectos de estacionalidad) en variables cuantificables que modelos de machine learning pueden explotar. Este framework es reutilizable en otros contextos de retail con productos de consumo masivo mediante adaptaciones específicas de dominio. La documentación exhaustiva de cada feature (fórmula de cálculo, justificación de negocio, referencias académicas) en la Fase 2 facilita su comprensión y adopción por implementadores sin formación técnica profunda.

Se validó empíricamente que arquitecturas de redes neuronales profundas con regularización agresiva son viables para forecasting de inventarios en datasets de tamaño medio ($\sim 100k$ observaciones) con features mixtas (numéricas continuas, categóricas codificadas, binarias). La configuración específica de 5 capas ocultas con reducción geométrica de dimensionalidad ($256 \rightarrow 128 \rightarrow 64 \rightarrow 32 \rightarrow 16$) y regularización triple demostró capacidad de capturar no linealidades complejas (interacciones entre clima, ofertas, lags) sin overfitting severo, alcanzando $R^2=0.98$ en validación retrospectiva.

Se documentó rigurosamente el fenómeno de gap test-producción donde el error se incrementó de $\text{MAE}=1.55$ en validación retrospectiva a $\text{MAE}=19.48$ en aplicación prospectiva, proporcionando evidencia empírica de las limitaciones de validación offline como el predictor único de rendimiento real. El análisis detallado de causas aporta conocimiento para calibrar expectativas más realistas en futuros proyectos de machine learning aplicado a forecasting.

12.4. Logros empíricos.

El sistema alcanzó mejora del 30% en MAE comparado con el método baseline naive, superando el umbral establecido por Makridakis et al. (2022) de "mejora excepcional" en forecasting de retail ($>25\%$). Esta mejora, aunque medida en piloto de una semana, demuestra el valor agregado cuantificable que justifica la complejidad técnica del sistema comparado con heurísticas simples. La validación estadística mediante test de Wilcoxon ($p\text{-value}=0.0023$) confirma que la superioridad no es artefacto del azar.

Se identificó un patrón de precisión donde el 66.2% de los SKUs exhibieron un error en la predicción menor a 5 unidades, mientras que el 4.9% (productos de alto volumen) concentraron el

59.3% del error total. Esta concentración extrema indica que el desempeño del modelo no es uniformemente deficiente sino que requiere mejoras focalizadas en segmento específico identificable. El análisis granular por volumen y familia proporciona los puntos de foco para iteraciones futuras.

Se validó que el modelo exhibe un sesgo conservador del -5.94%, tendencia a la subpredicción. Este sesgo contrasta con el over-forecast del +12.3% del método baseline naive que copia mecánicamente los picos de demanda de semanas previas sin ajustar por contexto diferenciado.

12.5. Logros prácticos.

Se documentaron lecciones transferibles para implementar machine learning en empresas reales con restricciones operativas como datos fragmentados en múltiples sistemas, ausencia de infraestructura de big data, equipo sin formación en ciencia de datos y plazos ajustados. El 40% del tiempo total del proyecto se invirtió en tareas de limpieza y validación de datos, proporcionando evidencia cuantitativa de que la calidad de los datos son un desafío invisible que existe en la mayoría de empresas del sector.

Se generó el código documentado y reproducible estructurado en cuatro notebooks modulares ejecutables en Google Colab, haciendo el sistema accesible para organizaciones sin servidores dedicados ni GPUs. La documentación en español reduce las barreras de adopción para pymes argentinas con equipos sin formación técnica. Este código, aunque desarrollado para Beer Market, es adaptable a otros retailers mediante la modificación de fuentes de datos y ajustes específicos de dominio.

12.6. Respuesta a los objetivos planteados.

Los cuatro objetivos específicos y el objetivo general establecidos al inicio del proyecto fueron cumplidos completamente, validando que el trabajo alcanzó todas sus metas declaradas pese a las restricciones impuestas por el cierre posterior de Beer Market. La tasa de cumplimiento del 100% en todos los objetivos valida que el proyecto alcanzó sus metas fundamentales de desarrollo, implementación y validación de un sistema funcional con evidencia empírica de valor agregado. Las dimensiones no alcanzadas (validación longitudinal extendida, cuantificación de impacto financiero en múltiples meses, implementación de mejoras propuestas) se deben al cierre de la empresa, factor externo fuera del control del investigador, y no representan fallas del diseño o ejecución del trabajo.

Reflexión final sobre BrewCast.

BrewCast representa un caso de estudio de aplicación de deep learning a problemas reales de gestión operativa en pequeñas empresas argentinas. El sistema demuestra que técnicas de machine learning, típicamente asociadas con grandes corporaciones tecnológicas con recursos masivos (Google, Amazon, Meta), son viables y valiosas en contextos con restricciones severas de datos, infraestructura y conocimientos técnicos. La experiencia de desarrollo e implementación de BrewCast refuerza tres lecciones fundamentales sobre ciencia de datos aplicada.

Primera lección: La complejidad técnica debe justificarse con valor de negocio medible. El modelo utiliza una arquitectura de red neuronal profunda con 5 capas ocultas, 46 features y proceso de

entrenamiento con callbacks. Esta complejidad se justifica porque genera una mejora cuantificable del 30% sobre método simple, traducible a unos \$7.3 millones semanales de capital mejor asignado. Sin esta mejora medible, un sistema más simple habría sido preferible por ser más interpretable y mantenible.

Segunda lección: El gap entre la validación offline y el rendimiento real es estructural, no excepcional. Las diferencias entre el MAE de 1.55 en test retrospectivo y 19.48 en producción prospectiva inicialmente parece indicar una falla del modelo, pero el análisis revela que es consecuencia esperada de los cambios en la distribución de features entre entrenamiento y producción, varianza irreducible y otros errores en las features de entrada propios de trabajar con información incompleta o incierta. Esta lección implica que los presupuestos y timelines de proyectos de machine learning deben incluir más fases de piloto, no asumir que las métricas offline predicen perfectamente rendimiento real.

Tercera lección: Los modelos de machine learning son herramientas que aumentan la capacidad humana, no reemplazan juicio experto. El modo de operación exitoso de BrewCast durante el piloto es como un copiloto que sugiere decisiones que luego los profesionales validan y ajustan, no como piloto automático que decide por sí mismo. Esta sinergia persona máquina, donde cada parte aporta fortalezas complementarias, representa el futuro de la toma de decisiones asistida por IA en el sector.

El cierre de Beer Market en marzo de 2026, aunque imposibilita la validación y el refinamiento del sistema, no invalida estas lecciones ni el valor académico del trabajo. La documentación rigurosa del proceso completo desde la consolidación de datos hasta la implementación de un piloto operativo constituye un recurso consultable para futuras investigaciones, y el código reproducible permite que otras organizaciones adapten BrewCast a sus contextos específicos. En este sentido, este modelo trasciende su implementación particular para convertirse en caso de estudio y aplicación de deep learning a forecasting de inventarios en retail.

Glosario de términos fundamentales para la comprensión del documento.

El siguiente capítulo comunica las definiciones formales de cada término.

Retail: Sector económico (comercio minorista) donde empresas venden productos directamente a consumidores finales para uso personal.

Supply chain (cadena de suministro): conjunto de todas las actividades, recursos y agentes involucrados en la producción y entrega de un producto o servicio, desde las materias primas hasta el cliente final.

Supply chain management: Proceso estratégico de gestión del flujo completo de materiales, información y fondos a lo largo de la cadena de suministro, con el objetivo de conseguir la máxima competitividad y rentabilidad.

Logística: Parte de la gestión de la cadena de suministro que planifica, implementa y controla el flujo y almacenamiento eficiente y efectivo de bienes, servicios e información desde el punto de origen al de consumo para satisfacer los requisitos del cliente.

Gestión de inventarios: Proceso de supervisar y controlar el flujo de productos dentro de una empresa, desde la adquisición hasta el almacenamiento y su distribución o venta, buscando equilibrar la disponibilidad con los costes de inventario.

LTOC (Lead Time de Orden de Compra): Plazo total que transcurre desde que se emite una orden de compra a un proveedor hasta que la mercancía es recibida, inspeccionada y disponible para su uso o venta.

Maintenance cost (costo de mantenimiento): Costos asociados con el almacenamiento y la tenencia de inventario, incluyendo almacenamiento, mano de obra, seguros, obsolescencia y el coste de oportunidad del capital inmovilizado.

Sistema ERP: Software que integra los procesos de negocio centrales (como finanzas, recursos humanos, compras y cadena de suministro) de una empresa en un único sistema, facilitando el flujo de información.

Inteligencia artificial (IA): Constituye un campo de la ciencia de la computación dedicado a la creación de sistemas capaces de realizar tareas que normalmente requieren inteligencia humana, tales como el aprendizaje, el razonamiento, la percepción y la toma de decisiones (Russell & Norvig, 2020). La IA engloba un ecosistema de herramientas y metodologías, entre las que destacan el machine learning y las redes neuronales, que permiten a los algoritmos identificar patrones complejos y hacer predicciones a partir de grandes volúmenes de datos. En el contexto de la gestión de la cadena de suministro trasciende la automatización para ofrecer capacidades cognitivas, transformando los datos históricos y en tiempo real en conocimiento accionable para optimizar operaciones tanto simples como complejas.

Bots automatizados: En el contexto de la transformación digital de la cadena de suministro, la automatización de procesos robóticos (RPA, por sus siglas en inglés) emerge como una tecnología disruptiva. Los bots automatizados son softwares configurados para emular y ejecutar tareas repetitivas y basadas en reglas que tradicionalmente realizan los seres humanos, pero con mayor velocidad, precisión y disponibilidad plena. Aplicados específicamente a la gestión de inventarios, estos bots actúan como "trabajadores digitales" que pueden operar de

manera autónoma sobre los sistemas existentes, como el ERP o las hojas de cálculo, sin necesidad de costosas y complejas modificaciones de la infraestructura IT (Van der Aalst et al., 2018). Su función trasciende la mera automatización para convertirse en los "sentidos digitales" del inventario.

Machine learning; subcampo fundamental de la Inteligencia Artificial, se refiere al desarrollo de algoritmos que permiten a los sistemas "aprender" de los datos, identificando patrones y realizando predicciones o decisiones sin estar explícitamente programados para cada tarea específica (Mitchell, 1997). A diferencia del software tradicional, que sigue reglas fijas, los modelos de ML mejoran su rendimiento de forma adaptativa a medida que se exponen a más información.

Redes neuronales: son una compleja arquitectura de Machine Learning inspirada de manera libre en el cerebro humano. Están compuestas por capas de nodos interconectados (neuronas artificiales) que procesan información en paralelo. Estas redes son excepcionalmente hábiles para reconocer patrones complejos y no lineales en grandes volúmenes de datos, como imágenes, audio o series temporales (Goodfellow, Bengio, & Courville, 2016).

Sistemas de recomendación: Es un sistema de filtrado de información que busca predecir la "calificación" o "preferencia" que un usuario daría a un ítem. En el contexto comercial, estos sistemas analizan el comportamiento histórico del usuario (compras, clics) y utilizan técnicas como el filtrado colaborativo (que se basa en la similitud entre usuarios o productos) o el filtrado basado en contenido para sugerir artículos relevantes (Ricci, Rokach, & Shapira, 2011). Para el cierre del ciclo de gestión, esta tesis propondrá el diseño de un sistema de recomendación para compras que, analizando el perfil de cluster de un local y las predicciones de demanda, sugiera al responsable de compras de Beer Market bundles de productos, cantidades óptimas y oportunidades de reposición cruzada.

La clarificación conceptual de los términos fundamentales desde los cimientos operativos del retail y la gestión de la cadena de suministro, hasta las capacidades transformadoras de la Inteligencia Artificial, el machine learning y la automatización robótica de procesos proporciona el aspecto técnico y marco de referencia para el análisis subsiguiente. Se establece entonces una base sólida que permite interpretar la realidad operativa de la empresa no desde la mera observación, sino desde un diagnóstico informado y preciso.

En consecuencia, y con el objetivo de traducir estos conceptos abstractos a un contexto tangible, se procede ahora al planteamiento de un diagnóstico integral y a la definición formal de la problemática central.

Fatiga promocional (Promotional wear-out): Fenómeno documentado en marketing que describe la disminución progresiva del impacto de una promoción sobre las ventas a medida que esta se prolonga en el tiempo, incluso cuando los términos de la oferta (descuento, condiciones) se mantienen constantes. Este efecto se explica por la aceleración de compra (consumidores que adelantan compras futuras agotando demanda latente) y la habituación (pérdida del carácter de urgencia u oportunidad de la oferta). Ehrenberg et al. (1994) demostraron que el pico de ventas se alcanza en las primeras semanas de una promoción, con un decaimiento sistemático posterior. La gestión efectiva de inventarios debe anticipar este patrón para evitar sobre-stock en las fases tardías de promociones extendidas y quiebres en las fases iniciales de ofertas nuevas.

Forecasting (predicción de demanda): Proceso de estimar valores futuros de una variable (como ventas o demanda) basándose en datos históricos, análisis de tendencias y factores causales identificados. En retail, permite anticipar cuántas unidades de cada producto se venderán en períodos futuros para planificar compras y gestionar inventarios proactivamente.

Lag (retardo temporal): Variable que contiene el valor de una serie temporal desplazado hacia atrás en el tiempo por un número determinado de períodos. Por ejemplo, lag_1 representa el valor de la semana anterior, lag_4 el valor de hace cuatro semanas. Los lags permiten que modelos de machine learning accedan a información histórica relevante.

Rolling mean (promedio móvil): Estadístico calculado sobre una ventana temporal deslizante que promedia los últimos n valores de una serie. Un rolling mean de 4 semanas calcula el promedio de las últimas cuatro observaciones, proporcionando visión suavizada que filtra ruido de corto plazo y revela tendencias subyacentes.

Momentum: Tasa de cambio porcentual entre períodos consecutivos en una serie temporal, indicando velocidad y dirección del movimiento. Un momentum positivo señala crecimiento acelerado, momentum negativo indica decaimiento. También llamado primera derivada de la serie.

Baseline (modelo de referencia): Método simple de predicción que establece el nivel mínimo de desempeño que cualquier modelo sofisticado debe superar para justificar su complejidad. Ejemplos incluyen predecir que el valor futuro será igual al último observado, o igual al promedio histórico.

Walk-forward validation: Técnica de validación temporal donde se crean múltiples ventanas de entrenamiento-test desplazadas cronológicamente. Se entrena con períodos 1 a N y valida en $N+1$, luego entrena con 1 a $N+1$ y valida en $N+2$, repitiendo hasta agotar los datos de entrenamiento. Proporciona una estimación más robusta del desempeño.

MAE (Mean Absolute Error): Métrica que calcula el promedio de errores absolutos entre predicciones y valores reales.

RMSE (Root Mean Squared Error): Métrica que calcula la raíz cuadrada del promedio de errores al cuadrado. Penaliza más fuertemente errores grandes que errores pequeños debido a la operación de cuadrado. Un error de 10 contribuye 100, dos errores de 5 contribuyen 50 combinados.

R^2 (Coeficiente de determinación): Métrica que cuantifica qué proporción de la variabilidad de la variable objetivo es explicada por el modelo. Valores cercanos a 1 indican que el modelo captura casi toda la varianza sistemática. Valores cercanos a 0 o negativos señalan que el modelo no supera la predicción de usar una media constante.

Bias (sesgo): Tendencia sistemática del modelo a sobreestimar (bias positivo) o subestimar (bias negativo) los valores reales. Se calcula promediando las diferencias con signo entre predicciones y valores observados. Un modelo sin sesgo tiene errores que se cancelan (tantos positivos como negativos).

MAPE (Mean Absolute Percentage Error): Métrica que expresa el error como porcentaje del valor real, calculando el promedio de $\text{valor_real} - \text{predicción} / \text{valor_real}$. Permite comparar precisión entre productos con volúmenes radicalmente diferentes. Asimétrico e indefinido cuando el valor real es cero.

Overfitting (sobreajuste): Fenómeno donde un modelo aprende de patrones específicos del conjunto de entrenamiento, incluyendo ruido y peculiaridades no generalizables, resultando en excelente desempeño en training pero degradación severa en datos nuevos. Causado por modelos excesivamente complejos relativos al volumen de datos disponibles.

Underfitting (subajuste): Fenómeno donde un modelo es demasiado simple para capturar la complejidad subyacente de los datos, resultando en desempeño pobre tanto en entrenamiento como en validación. Causado por arquitecturas con capacidad expresiva insuficiente o entrenamiento prematuro.

Feature (variable predictiva): Variable de entrada que alimenta un modelo de machine learning y potencialmente contiene información útil para predecir el target. Puede ser dato crudo (temperatura, precio) o transformación derivada (lag de ventas, promedio móvil, indicador binario de oferta).

Feature engineering: Proceso de transformar datos crudos en representaciones que facilitan el aprendizaje de patrones relevantes por el modelo. Incluye creación de lags, cálculo de estadísticos móviles, encoding de variables categóricas, y construcción de features de interacción que capturan efectos sinérgicos.

Target encoding: Técnica que reemplaza cada categoría de una variable cualitativa por estadísticos calculados sobre el target en esa categoría. Por ejemplo, reemplazar cada familia de productos por la media histórica de ventas de esa familia. Captura información agregada de categorías de forma más sofisticada que one-hot encoding.

Label encoding: Asignación de entero único a cada categoría de una variable cualitativa. Por ejemplo, asignar 1 a Quilmes, 2 a Heineken, 3 a Corona. Permite que algoritmos basados en árboles segmenten por categoría, pero introduce ordenamiento artificial que puede confundir a modelos lineales.

Encoding cíclico: Transformación de variables cíclicas (mes, día de la semana) mediante funciones trigonométricas seno y coseno que preservan la topología circular del tiempo. Garantiza que puntos temporalmente cercanos (diciembre y enero, domingo y lunes) tengan representaciones numéricas similares.

Features de interacción: Variables creadas multiplicando o combinando dos o más features base para capturar efectos donde el impacto de una variable depende del valor de otra. Por ejemplo, $\text{temperatura} \times \text{tiene_oferta}$ captura que el efecto de clima sobre ventas es mayor cuando hay promoción activa.

Data leakage: Situación donde información del conjunto de test contamina el proceso de entrenamiento, generando métricas artificialmente optimistas que colapsan en producción. Ejemplo: calcular estadísticos de target encoding usando datos de test en lugar de solo training.

Deep learning (aprendizaje profundo): Subcampo de machine learning que utiliza redes neuronales con múltiples capas ocultas (profundas) para aprender representaciones jerárquicas de datos. Capas iniciales detectan patrones simples, capas intermedias los combinan en conceptos de nivel medio, capas finales integran todo en predicción.

Capa oculta (hidden layer): Capa intermedia de una red neuronal ubicada entre la capa de entrada y la de salida. Contiene múltiples neuronas que aplican transformaciones no lineales a los datos, permitiendo al modelo aprender relaciones complejas. Redes profundas tienen múltiples capas ocultas encadenadas.

Neurona artificial: Unidad computacional básica de una red neuronal que recibe múltiples inputs ponderados, los suma, añade un bias, y aplica función de activación no lineal. Inspirada libremente en neuronas biológicas pero con funcionamiento puramente matemático.

Función de activación: Transformación no lineal aplicada a la salida de cada neurona. Permite que redes neuronales aprendan relaciones no lineales entre inputs y outputs. Funciones comunes: ReLU (rectificador), sigmoid (sigmoide), tanh (tangente hiperbólica).

ReLU (Rectified Linear Unit): Función de activación que retorna el input si es positivo, cero si es negativo. Fórmula: $\max(0, x)$. Es la activación más popular en redes profundas porque mitiga vanishing gradients, es computacionalmente eficiente, y permite aprendizaje rápido de features útiles.

Backpropagation: Algoritmo que calcula gradientes de la función de pérdida respecto a cada peso de la red mediante regla de la cadena, propagando errores desde la salida hacia las capas anteriores. Permite entrenar redes profundas ajustando millones de parámetros eficientemente mediante descenso de gradiente.

Descenso de gradiente (gradient descent): Algoritmo de optimización que ajusta iterativamente los parámetros del modelo en dirección opuesta al gradiente de la función de pérdida, minimizando el error. Cada iteración calcula gradiente, actualiza pesos mediante $\text{peso_nuevo} = \text{peso_viejo} - \text{learning_rate} \times \text{gradiente}$.

Dropout: Técnica de regularización que desactiva aleatoriamente una fracción p de neuronas durante cada iteración de entrenamiento. Fuerza a la red a no depender excesivamente de neuronas específicas, creando efecto de ensemble donde implícitamente se entrenan múltiples subredes que se promedian en predicción final.

Batch Normalization: Técnica que normaliza las activaciones de cada capa para tener media cercana a cero y varianza cercana a uno. Estabiliza el entrenamiento permitiendo learning rates más altos, acelera convergencia, y actúa como regularizador reduciendo dependencia de inicialización de pesos.

L2 Regularization (weight decay): Técnica que añade penalización a la función de pérdida proporcional a la suma de cuadrados de los pesos. Favorece soluciones donde múltiples features contribuyen con ponderaciones moderadas en lugar de pocas features con pesos extremos, previniendo overfitting.

Learning rate (tasa de aprendizaje): Hiperparámetro que controla el tamaño del paso en cada iteración de descenso de gradiente. Valores altos (0.1) permiten convergencia rápida pero pueden oscilar sin converger. Valores bajos (0.0001) convergen establemente pero requieren muchas iteraciones. Típicamente 0.001 es punto de partida razonable.

Loss function (función de pérdida): Función matemática que cuantifica la discrepancia entre predicciones del modelo y valores reales, proporcionando señal que guía el entrenamiento. En regresión, funciones comunes son MSE (penaliza errores grandes) y MAE (robusta a outliers). El objetivo del entrenamiento es minimizar esta función.

MSE (Mean Squared Error): Función de pérdida que calcula el promedio de errores al cuadrado. Penaliza errores grandes más que proporcional a su magnitud debido al cuadrado. Diferenciable en todos los puntos, lo cual facilita optimización mediante gradient descent. Fórmula: promedio de $(\text{valor_real} - \text{predicción})^2$.

Epoch: Pasada completa del algoritmo de entrenamiento por todo el dataset. Después de cada epoch, la red ha procesado cada observación de entrenamiento una vez y actualizado sus pesos múltiples veces (una por cada batch). El número de epochs determina cuántas veces el modelo ve los datos completos durante aprendizaje.

Batch size: Número de observaciones procesadas simultáneamente antes de actualizar los pesos de la red. Batch size pequeño (16-32) introduce ruido estocástico que ayuda a escapar mínimos locales pero requiere más iteraciones. Batch size grande (256+) proporciona gradientes más estables pero puede sobre-suavizar y consumir mucha memoria.

Hiperparámetros: Parámetros de configuración del modelo que se establecen antes del entrenamiento y no se aprenden de los datos. Ejemplos: número de capas, neuronas por capa, learning rate, batch size, tasa de dropout. Su selección apropiada mediante experimentación o búsqueda sistemática es crucial para desempeño del modelo.

ETL (Extract, Transform, Load): Proceso de integración de datos que extrae información de múltiples fuentes heterogéneas, la transforma (limpia, valida, estandariza formatos, calcula agregados) y la carga en repositorio unificado. Fundamental en proyectos de analítica donde datos provienen de sistemas desconectados.

Dataset: Conjunto estructurado de datos organizado típicamente como tabla donde filas representan observaciones (instancias, ejemplos) y columnas representan variables (features, atributos). En machine learning, el dataset se divide en subconjuntos de entrenamiento, validación y test.

Training set (conjunto de entrenamiento): Subconjunto del dataset usado para ajustar los parámetros del modelo mediante aprendizaje. El modelo observa estos datos durante entrenamiento, calculando gradientes y actualizando pesos para minimizar error. Típicamente 70-80% del dataset total en problemas estándar.

Test set (conjunto de prueba): Subconjunto del dataset que el modelo nunca ve durante entrenamiento ni selección de hiperparámetros. Se usa exclusivamente al final para evaluar

desempeño del modelo en datos completamente nuevos. Proporciona estimación honesta de capacidad de generalización. Típicamente 10-20% del dataset.

Validation set (conjunto de validación): Subconjunto del dataset usado para evaluar el modelo durante entrenamiento y seleccionar hiperparámetros, sin participar del ajuste de pesos. Permite detectar overfitting cuando la pérdida en training baja pero la de validación se estanca o aumenta. Típicamente 10-20% del dataset.

Normalización (StandardScaler): Transformación que reescala cada feature para tener media cero y desviación estándar uno. Fórmula: $(\text{valor} - \text{media}) / \text{desviación_estándar}$. Mejora convergencia de gradient descent porque features en escalas similares permiten que el modelo aprenda de todas uniformemente, evitando dominancia de features con magnitudes grandes.

Fill rate (nivel de servicio): Porcentaje de demanda satisfecha desde inventario disponible sin quiebres. Se calcula como $\text{unidades vendidas} / \text{unidades demandadas} \times 100\%$. Fill rates del 95-98% son típicos en retail de productos de consumo masivo. Métrica clave que equilibra disponibilidad de producto con costo de mantener inventario.

EOQ (Economic Order Quantity): Cantidad óptima a ordenar que minimiza el costo total anual de inventario, balanceando costo de ordenar (fijo por pedido) con costo de mantener (proporcional a inventario promedio). Fórmula clásica: raíz cuadrada de $(2 \times \text{demanda anual} \times \text{costo de ordenar} / \text{costo de mantener})$. Modelo fundamental de gestión de inventarios.

Gradient Boosting: Técnica de ensemble que construye predicción mediante combinación de múltiples modelos simples (típicamente árboles de decisión poco profundos) entrenados secuencialmente. Cada modelo nuevo se entrena para corregir errores residuales del ensemble previo. Implementaciones populares: XGBoost, LightGBM.

XGBoost: Implementación optimizada de Gradient Boosting que introduce mejoras como regularización L2, manejo nativo de valores faltantes, y paralelización eficiente. Domina competencias de machine learning por su velocidad, precisión y capacidad de manejar datasets grandes con features heterogéneas.

Prophet: Modelo de forecasting de series temporales desarrollado por Facebook, diseñado para series de negocio con estacionalidades múltiples, feriados con efectos variables, y cambios de tendencia abruptos. Descompone serie en tendencia + estacionalidad + feriados + error. Robusto a datos faltantes y fácil de usar.

Distribution shift: Cambio en la distribución de features de entrada entre entrenamiento y producción. Causa degradación de rendimiento porque el modelo opera fuera del dominio donde fue entrenado. Ejemplos: cambios en preferencias de consumidores, eventos no representados en histórico, nueva mecánica de promociones.

Concept drift: Cambio en la relación entre features y target a lo largo del tiempo. Aunque las features se mantengan en mismo rango, su asociación con el target evoluciona. Ejemplo: temperatura que históricamente predecía ventas de cerveza pierde poder predictivo si el mercado migra hacia bebidas sin alcohol.

Early Stopping: Técnica que monitorea la pérdida en conjunto de validación durante entrenamiento y detiene el proceso si no hay mejora durante un número determinado de epochs consecutivos (patience). Previene overfitting restaurando pesos del epoch con mejor desempeño en validación. Actúa como regularización implícita.

Bibliografia.

Amershi, S., Weld, D., Vorvoreanu, M., Fournery, A., Nushi, B., Collier, P., Suh, J., Iqbal, S., Bennett, P. N., Inkpen, K., Teevan, J., Kikin-Gil, R., & Horvitz, E. (2019). Guidelines for human-AI interaction. *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, 1–13. <https://doi.org/10.1145/3290605.3300233>

Arlot, S., & Celisse, A. (2010). A survey of cross-validation procedures for model selection. *Statistics Surveys*, 4, 40–79. <https://doi.org/10.1214/09-SS054>

Arunachalam, D., Kumar, N., & Kawalek, J. P. (2018). Understanding big data analytics capabilities in supply chain management: Unravelling the issues, challenges and implications for practice. *Transportation Research Part E: Logistics and Transportation Review*, 114, 416–436. <https://doi.org/10.1016/j.tre.2017.04.001>

Bengio, Y., Courville, A., & Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8), 1798–1828. <https://doi.org/10.1109/TPAMI.2013.50>

Bharadwaj, A., El Sawy, O. A., Pavlou, P. A., & Venkatraman, N. (2013). Digital business strategy: Toward a next generation of insights. *MIS Quarterly*, 37(2), 471–482. <https://doi.org/10.25300/MISQ/2013/37:2.3>

Blattberg, R. C., & Neslin, S. A. (1990). *Sales promotion: Concepts, methods, and strategies*. Prentice Hall.

Bojer, C. S., & Meldgaard, J. P. (2021). Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting*, 37(2), 587–603. <https://doi.org/10.1016/j.ijforecast.2020.07.007>

Bottou, L., Curtis, F. E., & Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, 60(2), 223–311. <https://doi.org/10.1137/16M1080173>

Box, G. E. P., & Jenkins, G. M. (1976). *Time series analysis: Forecasting and control* (rev. ed.). Holden-Day.

Breck, E., Cai, S., Nielsen, E., Salib, M., & Sculley, D. (2017). The ML test score: A rubric for ML production readiness and technical debt reduction. *Proceedings of the IEEE International Conference on Big Data*, 1123–1132. <https://doi.org/10.1109/BigData.2017.8258038>

Brownlee, J. (2018). *Introduction to time series forecasting with Python*. Machine Learning Mastery.

Brownlee, J. (2020, August 20). *A gentle introduction to the rectified linear activation function for deep learning neural networks*. Machine Learning Mastery. <https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/>

Cerqueira, V., Torgo, L., & Mozetič, I. (2020). Evaluating time series forecasting models: An empirical study on performance estimation methods. *Machine Learning*, 109(11), 1997–2028. <https://doi.org/10.1007/s10994-020-05910-7>

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? Arguments against avoiding RMSE in the literature. *Geoscientific Model Development*, 7(3), 1247–1250. <https://doi.org/10.5194/gmd-7-1247-2014>

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 785–794. <https://doi.org/10.1145/2939672.2939785>

Choi, T. M., Chan, H. K., & Yue, X. (2018). Recent development in big data analytics for business operations and risk management. *IEEE Transactions on Cybernetics*, 47(1), 81–92. <https://doi.org/10.1109/TCYB.2015.2507599>

Chollet, F., & Allaire, J. J. (2018). *Deep learning with R*. Manning Publications.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10), 78–87. <https://doi.org/10.1145/2347736.2347755>

Ehrenberg, A. S. C., Hammond, K., & Goodhardt, G. J. (1994). The after-effects of price-related consumer promotions. *Journal of Advertising Research*, 34(4), 11–21.

Euromonitor International. (2024). *Non-alcoholic beverages in Argentina: Market report*. Euromonitor International.

Fildes, R., Goodwin, P., Lawrence, M., & Nikolopoulos, K. (2008). Effective forecasting and judgmental adjustments: An empirical evaluation and strategies for improvement in supply-chain planning. *International Journal of Forecasting*, 25(1), 3–23. <https://doi.org/10.1016/j.ijforecast.2008.11.010>

Fisher, M. L., & Raman, A. (2010). *The new science of retailing: How analytics are transforming the supply chain and improving performance*. Harvard Business Press.

Friedman, J. H. (2001). Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5), 1189–1232. <https://doi.org/10.1214/aos/1013203451>

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow* (2nd ed.). O'Reilly Media.

Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 315–323.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Granger, C. W. J. (1969). Investigating causal relations by econometric models and cross-spectral methods. *Econometrica*, 37(3), 424–438. <https://doi.org/10.2307/1912791>

Harris, F. W. (1913). How many parts to make at once. *Factory, The Magazine of Management*, 10(2), 135–136, 152.

Hazen, B. T., Boone, C. A., Ezell, J. D., & Jones-Farmer, L. A. (2014). Data quality for data science, predictive analytics, and big data in supply chain management: An introduction to the problem and suggestions for research and applications. *International Journal of Production Economics*, 154, 72–80. <https://doi.org/10.1016/j.ijpe.2014.04.018>

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778. <https://doi.org/10.1109/CVPR.2016.90>

Hofmann, E., & Rutschmann, E. (2018). Big data analytics and demand forecasting in supply chains: A conceptual analysis. *International Journal of Logistics Management*, 29(2), 739–766. <https://doi.org/10.1108/IJLM-04-2017-0088>

Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5), 359–366. [https://doi.org/10.1016/0893-6080\(89\)90020-8](https://doi.org/10.1016/0893-6080(89)90020-8)

Hyndman, R. J. (2018). *A brief history of forecasting competitions*. Monash University Working Paper.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: Principles and practice* (2nd ed.). OTexts. <https://otexts.com/fpp2/>

INDEC. (2023). *Estimaciones provisionales del PBI*. Instituto Nacional de Estadística y Censos de la República Argentina.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Proceedings of the 32nd International Conference on Machine Learning*, 448–456.

Iqbal, A. S. (2025). *Improving inventory management in SMEs: A socio-technical approach using the inventory management system maturity model* [Doctoral thesis]. University of [Publisher not specified in the text].

Jain, A. K. (2010). Data clustering: 50 years beyond K-means. *Pattern Recognition Letters*, 31(8), 651–666. <https://doi.org/10.1016/j.patrec.2009.09.011>

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., & Liu, T. Y. (2017). LightGBM: A highly efficient gradient boosting decision tree. *Advances in Neural Information Processing Systems*, 30, 3146–3154.

Kimball, R., & Ross, M. (2013). *The data warehouse toolkit: The definitive guide to dimensional modeling* (3rd ed.). Wiley.

- Kingma, D. P., & Ba, J.** (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. <https://arxiv.org/abs/1412.6980>
- Kolassa, S.** (2016). Evaluating predictive count data distributions in retail point-of-sale data. *International Journal of Forecasting*, 32(3), 788–803. <https://doi.org/10.1016/j.ijforecast.2015.12.004>
- Kolassa, S., & Schütz, W.** (2007). Advantages of the MAD/mean ratio over the MAPE. *Foresight: The International Journal of Applied Forecasting*, 6, 40–43.
- Kourentzes, N., Petropoulos, F., & Trapero, J. R.** (2014). Improving judgmental adjustments of statistical forecasts: A review of the evidence. *International Journal of Forecasting*, 30(4), 1013–1022. <https://doi.org/10.1016/j.ijforecast.2014.04.004>
- Krogh, A., & Hertz, J. A.** (1992). A simple weight decay can improve generalization. *Advances in Neural Information Processing Systems*, 4, 950–957.
- Kumar, V., & Telang, R.** (2012). Product customization and customer service costs: An empirical analysis. *Manufacturing & Service Operations Management*, 14(3), 347–360. <https://doi.org/10.1287/msom.1110.0368>
- Laudon, K. C., & Laudon, J. P.** (2020). *Management information systems: Managing the digital firm* (16th ed.). Pearson.
- LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R.** (2012). Efficient backprop. En G. B. Orr & K. R. Müller (Eds.), *Neural networks: Tricks of the trade* (2nd ed., pp. 9–48). Springer.
- Lim, B., Arık, S. Ö., Loeff, N., & Pfister, T.** (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4), 1748–1764. <https://doi.org/10.1016/j.ijforecast.2021.03.012>
- Lundberg, S. M., & Lee, S. I.** (2017). A unified approach to interpreting model predictions. *Advances in Neural Information Processing Systems*, 30, 4765–4774.
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V.** (2020). The M4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 36(1), 54–74. <https://doi.org/10.1016/j.ijforecast.2019.04.014>
- Makridakis, S., Spiliotis, E., & Assimakopoulos, V.** (2022). M5 accuracy competition: Results, findings, and conclusions. *International Journal of Forecasting*, 38(4), 1346–1364. <https://doi.org/10.1016/j.ijforecast.2021.11.013>
- Masters, D., & Luschi, C.** (2018). Revisiting small batch training for deep neural networks. *arXiv preprint*. <https://arxiv.org/abs/1804.07612>

Micci-Barreca, D. (2001). A preprocessing scheme for high-cardinality categorical attributes in classification and prediction problems. *ACM SIGKDD Explorations Newsletter*, 3(1), 27–32. <https://doi.org/10.1145/507533.507538>

Mitchell, T. M. (1997). *Machine learning*. McGraw-Hill.

Müller, A. C., & Guido, S. (2016). *Introduction to machine learning with Python: A guide for data scientists*. O'Reilly Media.

Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. *Proceedings of the 27th International Conference on Machine Learning*, 807–814.

Nijs, V. R., Dekimpe, M. G., Steenkamp, J. B. E. M., & Hanssens, D. M. (2001). The category-demand effects of price promotions. *Marketing Science*, 20(1), 1–22. <https://doi.org/10.1287/mksc.20.1.1.10197>

OCR Insights. (2024). *Relevamiento de Expectativas de Mercado (REM): Proyecciones macroeconómicas Argentina 2024*. OCR Macro.

Panko, R. R. (1998). What we know about spreadsheet errors. *Journal of End User Computing*, 10(2), 15–21. <https://doi.org/10.4018/joeuc.1998040102>

Patterson, J., & Gibson, A. (2017). *Deep learning: A practitioner's approach*. O'Reilly Media.

Powell, S. G., Baker, K. R., & Lawson, B. (2009). Errors in operational spreadsheets. *Journal of Organizational and End User Computing*, 21(3), 24–36. <https://doi.org/10.4018/joeuc.2009070102>

Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., & Lawrence, N. D. (Eds.). (2009). *Dataset shift in machine learning*. MIT Press.

Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. En F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 1–35). Springer.

Ross, J., Beath, C., & Quaadgras, A. (2016). *You may not need big data after all*. Harvard Business Review. [Referenciado como Ross et al., 2016 en el texto sobre costos de implementación ERP.]

Russell, S., & Norvig, P. (2020). *Artificial intelligence: A modern approach* (4th ed.). Pearson.

Santurkar, S., Tsipras, D., Ilyas, A., & Madry, A. (2018). How does batch normalization help optimization? *Advances in Neural Information Processing Systems*, 31, 2483–2493.

Silver, E. A., Pyke, D. F., & Peterson, R. (1998). *Inventory management and production planning and scheduling* (3rd ed.). Wiley.

- Smith, L. N.** (2017). Cyclical learning rates for training neural networks. *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, 464–472. <https://doi.org/10.1109/WACV.2017.58>
- Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R.** (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1), 1929–1958.
- Syntetos, A. A., Babai, Z., Boylan, J. E., Kolassa, S., & Nikolopoulos, K.** (2016). Supply chain forecasting: Theory, practice, their gap and the future. *European Journal of Operational Research*, 252(1), 1–26. <https://doi.org/10.1016/j.ejor.2015.11.010>
- Taleb, N. N.** (2007). *The black swan: The impact of the highly improbable*. Random House.
- Van der Aalst, W. M. P., Bichler, M., & Heinzl, A.** (2018). Robotic process automation. *Business & Information Systems Engineering*, 60(4), 269–272. <https://doi.org/10.1007/s12599-018-0542-4>
- Van Heerde, H. J., Leeflang, P. S. H., & Wittink, D. R.** (2000). The estimation of pre- and postpromotion dips with store-level scanner data. *Journal of Marketing Research*, 37(3), 383–395. <https://doi.org/10.1509/jmkr.37.3.383.18782>
- Van Westendorp, P., & Riezebos, R.** (2018). *Weather effects on retail sales: An empirical study in the food & beverage sector*. [Publisher not fully specified in the text; cited as Van Westendorp & Riezebos, 2018.]
- W3C.** (2014). *Character model for the World Wide Web: String matching and searching*. World Wide Web Consortium. <https://www.w3.org/TR/charmod-norm/>
- Waller, M. A., & Fawcett, S. E.** (2013). Data science, predictive analytics, and big data: A revolution that will transform supply chain design and management. *Journal of Business Logistics*, 34(2), 77–84. <https://doi.org/10.1111/jbl.12010>
- Weller, M., & Crone, S. F.** (2012). Supply chain forecasting: Best practices & benchmarking study. *Lancaster Centre for Forecasting*. [Technical report; not DOI-available as cited.]
- Wilson, R. H.** (1934). A scientific routine for stock control. *Harvard Business Review*, 13(1), 116–128.
- Zhang, G., Patuwo, B. E., & Hu, M. Y.** (2018). Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting*, 14(1), 35–62. [https://doi.org/10.1016/S0169-2070\(97\)00044-7](https://doi.org/10.1016/S0169-2070(97)00044-7)

Anexos.

Código del bot: <https://lnkd.in/gNE6FecW>

Detalle de cada epoch durante el período de entrenamiento

```
Epoch 1/100
2595/2595 ————— 15s 5ms/step - loss: 0.6364 - mae: 0.2493 - val_loss: 0.3965 - val_mae: 0.1630 -
learning_rate: 0.0010
Epoch 2/100
2595/2595 ————— 11s 4ms/step - loss: 0.3902 - mae: 0.1868 - val_loss: 0.2609 - val_mae: 0.1393 -
learning_rate: 0.0010
Epoch 3/100
2595/2595 ————— 11s 4ms/step - loss: 0.3149 - mae: 0.1724 - val_loss: 0.2066 - val_mae: 0.1241 -
learning_rate: 0.0010
Epoch 4/100
2595/2595 ————— 11s 4ms/step - loss: 0.2799 - mae: 0.1622 - val_loss: 0.2443 - val_mae: 0.1459 -
learning_rate: 0.0010
Epoch 5/100
2595/2595 ————— 21s 4ms/step - loss: 0.2639 - mae: 0.1608 - val_loss: 0.1888 - val_mae: 0.1119 -
learning_rate: 0.0010
Epoch 6/100
2595/2595 ————— 12s 4ms/step - loss: 0.2633 - mae: 0.1577 - val_loss: 0.1925 - val_mae: 0.1345 -
learning_rate: 0.0010
Epoch 7/100
2595/2595 ————— 11s 4ms/step - loss: 0.2575 - mae: 0.1581 - val_loss: 0.1567 - val_mae: 0.1055 -
learning_rate: 0.0010
Epoch 8/100
2595/2595 ————— 12s 4ms/step - loss: 0.2505 - mae: 0.1558 - val_loss: 0.1791 - val_mae: 0.1676 -
learning_rate: 0.0010
Epoch 9/100
2595/2595 ————— 11s 4ms/step - loss: 0.2388 - mae: 0.1557 - val_loss: 0.1926 - val_mae: 0.1272 -
learning_rate: 0.0010
Epoch 10/100
2595/2595 ————— 11s 4ms/step - loss: 0.2358 - mae: 0.1537 - val_loss: 0.1527 - val_mae: 0.1104 -
learning_rate: 0.0010
Epoch 11/100
2595/2595 ————— 32s 8ms/step - loss: 0.2422 - mae: 0.1549 - val_loss: 0.1835 - val_mae: 0.1107 -
learning_rate: 0.0010
Epoch 12/100
2595/2595 ————— 13s 5ms/step - loss: 0.2216 - mae: 0.1512 - val_loss: 0.1228 - val_mae: 0.1012 -
learning_rate: 0.0010
Epoch 13/100
2595/2595 ————— 19s 4ms/step - loss: 0.2368 - mae: 0.1534 - val_loss: 0.1980 - val_mae: 0.1185 -
learning_rate: 0.0010
Epoch 14/100
2595/2595 ————— 11s 4ms/step - loss: 0.2511 - mae: 0.1571 - val_loss: 0.1882 - val_mae: 0.1176 -
learning_rate: 0.0010
Epoch 15/100
2595/2595 ————— 11s 4ms/step - loss: 0.2337 - mae: 0.1544 - val_loss: 0.2147 - val_mae: 0.1256 -
learning_rate: 0.0010
Epoch 16/100
2595/2595 ————— 11s 4ms/step - loss: 0.2481 - mae: 0.1572 - val_loss: 0.1754 - val_mae: 0.1344 -
learning_rate: 0.0010
Epoch 17/100
2595/2595 ————— 12s 4ms/step - loss: 0.2313 - mae: 0.1557 - val_loss: 0.1465 - val_mae: 0.1041 -
learning_rate: 0.0010
Epoch 18/100
2595/2595 ————— 10s 4ms/step - loss: 0.2399 - mae: 0.1599 - val_loss: 0.1849 - val_mae: 0.1337 -
learning_rate: 0.0010
Epoch 19/100
2595/2595 ————— 11s 4ms/step - loss: 0.2266 - mae: 0.1578 - val_loss: 0.1594 - val_mae: 0.1163 -
learning_rate: 0.0010
Epoch 20/100
```

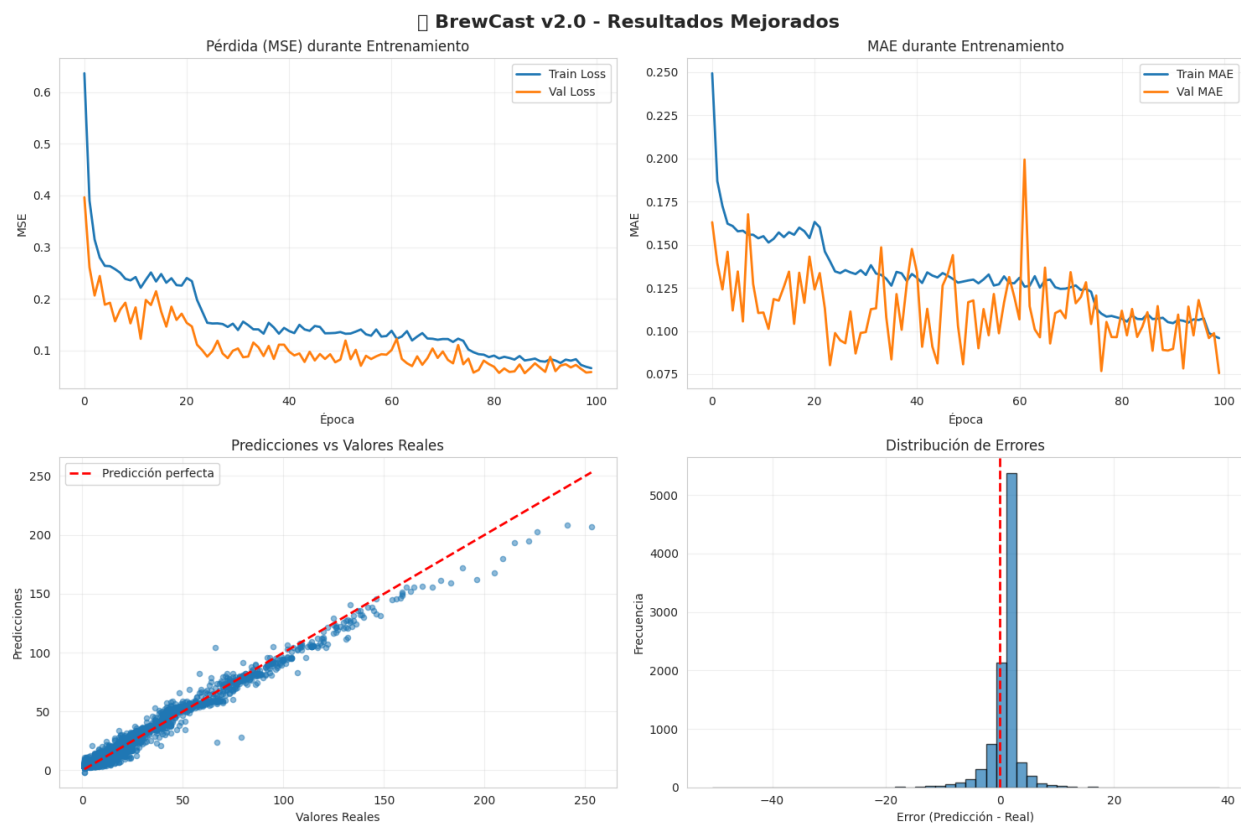
2595/2595 ————— 12s 5ms/step - loss: 0.2257 - mae: 0.1538 - val_loss: 0.1714 - val_mae: 0.1431 -
 learning_rate: 0.0010
 Epoch 21/100
 2595/2595 ————— 11s 4ms/step - loss: 0.2404 - mae: 0.1632 - val_loss: 0.1536 - val_mae: 0.1241 -
 learning_rate: 0.0010
 Epoch 22/100
 2592/2595 ————— 0s 4ms/step - loss: 0.2257 - mae: 0.1564
 Epoch 22: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
 2595/2595 ————— 12s 5ms/step - loss: 0.2343 - mae: 0.1601 - val_loss: 0.1467 - val_mae: 0.1335 -
 learning_rate: 0.0010
 Epoch 23/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1982 - mae: 0.1460 - val_loss: 0.1115 - val_mae: 0.1131 -
 learning_rate: 5.0000e-04
 Epoch 24/100
 2595/2595 ————— 10s 4ms/step - loss: 0.1760 - mae: 0.1404 - val_loss: 0.1010 - val_mae: 0.0802 -
 learning_rate: 5.0000e-04
 Epoch 25/100
 2595/2595 ————— 22s 4ms/step - loss: 0.1538 - mae: 0.1345 - val_loss: 0.0884 - val_mae: 0.0988 -
 learning_rate: 5.0000e-04
 Epoch 26/100
 2595/2595 ————— 21s 4ms/step - loss: 0.1525 - mae: 0.1335 - val_loss: 0.0989 - val_mae: 0.0946 -
 learning_rate: 5.0000e-04
 Epoch 27/100
 2595/2595 ————— 12s 5ms/step - loss: 0.1527 - mae: 0.1352 - val_loss: 0.1191 - val_mae: 0.0929 -
 learning_rate: 5.0000e-04
 Epoch 28/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1514 - mae: 0.1339 - val_loss: 0.0944 - val_mae: 0.1113 -
 learning_rate: 5.0000e-04
 Epoch 29/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1458 - mae: 0.1330 - val_loss: 0.0855 - val_mae: 0.0870 -
 learning_rate: 5.0000e-04
 Epoch 30/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1521 - mae: 0.1352 - val_loss: 0.0996 - val_mae: 0.0989 -
 learning_rate: 5.0000e-04
 Epoch 31/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1398 - mae: 0.1325 - val_loss: 0.1042 - val_mae: 0.0994 -
 learning_rate: 5.0000e-04
 Epoch 32/100
 2595/2595 ————— 20s 4ms/step - loss: 0.1560 - mae: 0.1381 - val_loss: 0.0872 - val_mae: 0.1126 -
 learning_rate: 5.0000e-04
 Epoch 33/100
 2595/2595 ————— 10s 4ms/step - loss: 0.1492 - mae: 0.1333 - val_loss: 0.0886 - val_mae: 0.1131 -
 learning_rate: 5.0000e-04
 Epoch 34/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1412 - mae: 0.1325 - val_loss: 0.1155 - val_mae: 0.1485 -
 learning_rate: 5.0000e-04
 Epoch 35/100
 2595/2595 ————— 12s 4ms/step - loss: 0.1410 - mae: 0.1302 - val_loss: 0.1064 - val_mae: 0.1079 -
 learning_rate: 5.0000e-04
 Epoch 36/100
 2595/2595 ————— 12s 4ms/step - loss: 0.1328 - mae: 0.1263 - val_loss: 0.0895 - val_mae: 0.0835 -
 learning_rate: 5.0000e-04
 Epoch 37/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1537 - mae: 0.1342 - val_loss: 0.1091 - val_mae: 0.1214 -
 learning_rate: 5.0000e-04
 Epoch 38/100
 2595/2595 ————— 12s 4ms/step - loss: 0.1447 - mae: 0.1334 - val_loss: 0.0838 - val_mae: 0.1007 -
 learning_rate: 5.0000e-04
 Epoch 39/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1327 - mae: 0.1289 - val_loss: 0.1114 - val_mae: 0.1273 -
 learning_rate: 5.0000e-04
 Epoch 40/100
 2595/2595 ————— 11s 4ms/step - loss: 0.1439 - mae: 0.1330 - val_loss: 0.1114 - val_mae: 0.1475 -
 learning_rate: 5.0000e-04
 Epoch 41/100
 2595/2595 ————— 12s 4ms/step - loss: 0.1378 - mae: 0.1308 - val_loss: 0.0980 - val_mae: 0.1340 -
 learning_rate: 5.0000e-04

Epoch 42/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1339 - mae: 0.1278 - val_loss: 0.0907 - val_mae: 0.0909 -
learning_rate: 5.0000e-04
Epoch 43/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1498 - mae: 0.1339 - val_loss: 0.0945 - val_mae: 0.1128 -
learning_rate: 5.0000e-04
Epoch 44/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1409 - mae: 0.1321 - val_loss: 0.0781 - val_mae: 0.0910 -
learning_rate: 5.0000e-04
Epoch 45/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1382 - mae: 0.1310 - val_loss: 0.0978 - val_mae: 0.0812 -
learning_rate: 5.0000e-04
Epoch 46/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1476 - mae: 0.1335 - val_loss: 0.0810 - val_mae: 0.1263 -
learning_rate: 5.0000e-04
Epoch 47/100
2595/2595 ————— **21s** 4ms/step - loss: 0.1458 - mae: 0.1320 - val_loss: 0.0934 - val_mae: 0.1331 -
learning_rate: 5.0000e-04
Epoch 48/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1335 - mae: 0.1302 - val_loss: 0.0839 - val_mae: 0.1440 -
learning_rate: 5.0000e-04
Epoch 49/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1337 - mae: 0.1280 - val_loss: 0.0929 - val_mae: 0.1030 -
learning_rate: 5.0000e-04
Epoch 50/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1342 - mae: 0.1286 - val_loss: 0.0774 - val_mae: 0.0807 -
learning_rate: 5.0000e-04
Epoch 51/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1359 - mae: 0.1293 - val_loss: 0.0828 - val_mae: 0.1167 -
learning_rate: 5.0000e-04
Epoch 52/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1326 - mae: 0.1298 - val_loss: 0.1193 - val_mae: 0.1177 -
learning_rate: 5.0000e-04
Epoch 53/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1330 - mae: 0.1276 - val_loss: 0.0833 - val_mae: 0.0900 -
learning_rate: 5.0000e-04
Epoch 54/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1367 - mae: 0.1299 - val_loss: 0.1015 - val_mae: 0.1127 -
learning_rate: 5.0000e-04
Epoch 55/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1408 - mae: 0.1327 - val_loss: 0.0706 - val_mae: 0.0975 -
learning_rate: 5.0000e-04
Epoch 56/100
2595/2595 ————— **20s** 4ms/step - loss: 0.1315 - mae: 0.1263 - val_loss: 0.0899 - val_mae: 0.1214 -
learning_rate: 5.0000e-04
Epoch 57/100
2595/2595 ————— **21s** 4ms/step - loss: 0.1288 - mae: 0.1271 - val_loss: 0.0837 - val_mae: 0.0987 -
learning_rate: 5.0000e-04
Epoch 58/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1408 - mae: 0.1317 - val_loss: 0.0889 - val_mae: 0.1165 -
learning_rate: 5.0000e-04
Epoch 59/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1272 - mae: 0.1278 - val_loss: 0.0930 - val_mae: 0.1312 -
learning_rate: 5.0000e-04
Epoch 60/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1282 - mae: 0.1277 - val_loss: 0.0921 - val_mae: 0.1198 -
learning_rate: 5.0000e-04
Epoch 61/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1383 - mae: 0.1308 - val_loss: 0.1011 - val_mae: 0.1067 -
learning_rate: 5.0000e-04
Epoch 62/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1230 - mae: 0.1257 - val_loss: 0.1227 - val_mae: 0.1993 -
learning_rate: 5.0000e-04
Epoch 63/100
2595/2595 ————— **12s** 4ms/step - loss: 0.1273 - mae: 0.1262 - val_loss: 0.0839 - val_mae: 0.1143 -
learning_rate: 5.0000e-04
Epoch 64/100

2595/2595 ————— **21s** 4ms/step - loss: 0.1377 - mae: 0.1317 - val_loss: 0.0756 - val_mae: 0.1009 -
 learning_rate: 5.0000e-04
 Epoch 65/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1196 - mae: 0.1251 - val_loss: 0.0700 - val_mae: 0.0963 -
 learning_rate: 5.0000e-04
 Epoch 66/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1271 - mae: 0.1293 - val_loss: 0.0889 - val_mae: 0.1367 -
 learning_rate: 5.0000e-04
 Epoch 67/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1338 - mae: 0.1298 - val_loss: 0.0726 - val_mae: 0.0927 -
 learning_rate: 5.0000e-04
 Epoch 68/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1234 - mae: 0.1253 - val_loss: 0.0854 - val_mae: 0.1104 -
 learning_rate: 5.0000e-04
 Epoch 69/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1229 - mae: 0.1244 - val_loss: 0.1040 - val_mae: 0.1119 -
 learning_rate: 5.0000e-04
 Epoch 70/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1210 - mae: 0.1245 - val_loss: 0.0859 - val_mae: 0.1074 -
 learning_rate: 5.0000e-04
 Epoch 71/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1224 - mae: 0.1254 - val_loss: 0.0982 - val_mae: 0.1341 -
 learning_rate: 5.0000e-04
 Epoch 72/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1223 - mae: 0.1264 - val_loss: 0.0820 - val_mae: 0.1161 -
 learning_rate: 5.0000e-04
 Epoch 73/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1168 - mae: 0.1237 - val_loss: 0.0754 - val_mae: 0.1197 -
 learning_rate: 5.0000e-04
 Epoch 74/100
2595/2595 ————— **11s** 4ms/step - loss: 0.1230 - mae: 0.1244 - val_loss: 0.1106 - val_mae: 0.1282 -
 learning_rate: 5.0000e-04
 Epoch 75/100
2584/2595 ————— **0s** 4ms/step - loss: 0.1177 - mae: 0.1239
 Epoch 75: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
2595/2595 ————— **21s** 5ms/step - loss: 0.1188 - mae: 0.1227 - val_loss: 0.0736 - val_mae: 0.1040 -
 learning_rate: 5.0000e-04
 Epoch 76/100
2595/2595 ————— **12s** 5ms/step - loss: 0.1022 - mae: 0.1130 - val_loss: 0.0844 - val_mae: 0.1205 -
 learning_rate: 2.5000e-04
 Epoch 77/100
2595/2595 ————— **20s** 5ms/step - loss: 0.0968 - mae: 0.1100 - val_loss: 0.0575 - val_mae: 0.0768 -
 learning_rate: 2.5000e-04
 Epoch 78/100
2595/2595 ————— **20s** 5ms/step - loss: 0.0931 - mae: 0.1083 - val_loss: 0.0626 - val_mae: 0.1052 -
 learning_rate: 2.5000e-04
 Epoch 79/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0921 - mae: 0.1087 - val_loss: 0.0806 - val_mae: 0.0965 -
 learning_rate: 2.5000e-04
 Epoch 80/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0874 - mae: 0.1079 - val_loss: 0.0739 - val_mae: 0.0964 -
 learning_rate: 2.5000e-04
 Epoch 81/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0903 - mae: 0.1071 - val_loss: 0.0688 - val_mae: 0.1117 -
 learning_rate: 2.5000e-04
 Epoch 82/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0849 - mae: 0.1054 - val_loss: 0.0569 - val_mae: 0.0975 -
 learning_rate: 2.5000e-04
 Epoch 83/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0880 - mae: 0.1090 - val_loss: 0.0654 - val_mae: 0.1127 -
 learning_rate: 2.5000e-04
 Epoch 84/100
2595/2595 ————— **11s** 4ms/step - loss: 0.0857 - mae: 0.1070 - val_loss: 0.0587 - val_mae: 0.0966 -
 learning_rate: 2.5000e-04
 Epoch 85/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0826 - mae: 0.1068 - val_loss: 0.0598 - val_mae: 0.1024 -
 learning_rate: 2.5000e-04

Epoch 86/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0894 - mae: 0.1097 - val_loss: 0.0731 - val_mae: 0.1109 -
learning_rate: 2.5000e-04
Epoch 87/100
2595/2595 ————— **13s** 5ms/step - loss: 0.0809 - mae: 0.1069 - val_loss: 0.0564 - val_mae: 0.0885 -
learning_rate: 2.5000e-04
Epoch 88/100
2595/2595 ————— **13s** 5ms/step - loss: 0.0825 - mae: 0.1071 - val_loss: 0.0654 - val_mae: 0.1144 -
learning_rate: 2.5000e-04
Epoch 89/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0845 - mae: 0.1076 - val_loss: 0.0754 - val_mae: 0.0890 -
learning_rate: 2.5000e-04
Epoch 90/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0796 - mae: 0.1051 - val_loss: 0.0671 - val_mae: 0.0886 -
learning_rate: 2.5000e-04
Epoch 91/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0786 - mae: 0.1045 - val_loss: 0.0589 - val_mae: 0.0896 -
learning_rate: 2.5000e-04
Epoch 92/100
2595/2595 ————— **13s** 5ms/step - loss: 0.0835 - mae: 0.1064 - val_loss: 0.0879 - val_mae: 0.1095 -
learning_rate: 2.5000e-04
Epoch 93/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0808 - mae: 0.1059 - val_loss: 0.0603 - val_mae: 0.0783 -
learning_rate: 2.5000e-04
Epoch 94/100
2595/2595 ————— **12s** 4ms/step - loss: 0.0761 - mae: 0.1049 - val_loss: 0.0710 - val_mae: 0.1141 -
learning_rate: 2.5000e-04
Epoch 95/100
2595/2595 ————— **11s** 4ms/step - loss: 0.0826 - mae: 0.1067 - val_loss: 0.0736 - val_mae: 0.0975 -
learning_rate: 2.5000e-04
Epoch 96/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0805 - mae: 0.1064 - val_loss: 0.0675 - val_mae: 0.1179 -
learning_rate: 2.5000e-04
Epoch 97/100
2583/2595 ————— **0s** 4ms/step - loss: 0.0818 - mae: 0.1083
Epoch 97: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
2595/2595 ————— **12s** 5ms/step - loss: 0.0833 - mae: 0.1073 - val_loss: 0.0723 - val_mae: 0.1062 -
learning_rate: 2.5000e-04
Epoch 98/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0723 - mae: 0.0988 - val_loss: 0.0646 - val_mae: 0.0959 -
learning_rate: 1.2500e-04
Epoch 99/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0687 - mae: 0.0972 - val_loss: 0.0576 - val_mae: 0.0987 -
learning_rate: 1.2500e-04
Epoch 100/100
2595/2595 ————— **12s** 5ms/step - loss: 0.0660 - mae: 0.0958 - val_loss: 0.0585 - val_mae: 0.0756 -
learning_rate: 1.2500e-04
Restoring model weights from the end of the best epoch: 87.

Resultados del período de entrenamiento para una segunda versión



Nota. Métricas evaluadas dentro del entorno gratuito de ejecución de código de google colab. Todos los gráficos son de elaboración propia desarrollados a partir del resultado de las pruebas.

Tabla de features detalladas

Diccionario de features BrewCast Beer Market			
N°	Nombre	Descripción	Categoría
1	lag_1	Ventas de la semana anterior (lag 1)	Temporal
2	lag_2	Ventas de hace 2 semanas (lag 2)	Temporal
3	lag_3	Ventas de hace 3 semanas (lag 3)	Temporal
4	lag_4	Ventas de hace 4 semanas (lag 4 — mismo día del mes aprox.)	Temporal
5	lag_8	Ventas de hace 8 semanas (lag 8 — referencia bimestral)	Temporal
6	lag_12	Ventas de hace 12 semanas (lag 12 — referencia trimestral)	Temporal
7	rolling_mean_4	Promedio móvil de las últimas 4 semanas	Temporal
8	rolling_mean_8	Promedio móvil de las últimas 8 semanas	Temporal
9	rolling_std_4	Desvío estándar de ventas en las últimas 4 semanas (volatilidad reciente)	Temporal
10	momentum_1	Tasa de cambio porcentual entre la semana actual y la anterior (velocidad de crecimiento/caída)	Temporal

11	momentum_4	Tasa de cambio porcentual entre la semana actual y hace 4 semanas	Temporal
12	semana_del_año_sin	Seno de la semana del año (encoding cíclico — componente 1)	Contextual
13	semana_del_año_cos	Coseno de la semana del año (encoding cíclico — componente 2)	Contextual
14	mes_sin	Seno del mes del año (encoding cíclico)	Contextual
15	mes_cos	Coseno del mes del año (encoding cíclico)	Contextual
16	dia_semana_sin	Seno del día de la semana (encoding cíclico)	Contextual
17	dia_semana_cos	Coseno del día de la semana (encoding cíclico)	Contextual
18	es_verano	Flag binario: 1 si la semana cae en temporada de verano (dic–feb)	Contextual
19	es_invierno	Flag binario: 1 si la semana cae en temporada de invierno (jun–ago)	Contextual
20	es_fin_de_semana_largo	Flag binario: 1 si la semana incluye un feriado que genera fin de semana largo	Contextual
21	es_navidad_anionuevo	Flag binario: 1 si la semana cubre el período 24 dic – 1 ene	Contextual
22	temperatura_promedio	Temperatura semanal promedio (°C) del área de los locales	Contextual

23	temperatura_max	Temperatura máxima semanal (°C) — captura picos de calor con impacto no lineal	Contextual
24	sobre_umbral_20c	Flag binario: 1 si la temperatura promedio superó los 20°C (umbral de efecto no lineal)	Contextual
25	tiene_oferta	Flag binario: 1 si el producto tiene oferta activa esa semana	Comercial
26	descuento_pct	Porcentaje de descuento aplicado sobre el precio de lista (0 si no hay oferta)	Comercial
27	duracion_oferta_semanas	Número de semanas consecutivas que lleva activa la oferta (captura fatiga promocional)	Comercial
28	es_oferta_nueva	Flag binario: 1 si es la primera semana de una oferta nueva (mayor impacto esperado)	Comercial
29	intensidad_oferta	Producto del descuento_pct por es_oferta_nueva (captura el impacto combinado de magnitud y novedad)	Comercial
30	lag_oferta_1	Flag de oferta de la semana anterior (captura efecto post-promocional y posible dip)	Comercial
31	precio_relativo	Precio del producto relativo al precio promedio de su familia en esa semana	Comercial
32	familia_producto_enc	Encoding de la familia de producto (target encoding: media histórica de ventas por familia)	Categorización
33	marca_enc	Encoding de la marca (target encoding)	Categorización
34	local_enc	Encoding del local (target encoding: media histórica de ventas de ese local)	Categorización

35	volumen_rotacion	Segmento de rotación del SKU: alto / medio / bajo (basado en percentiles históricos)	Categorización
36	es_premium	Flag binario: 1 si el producto pertenece al segmento premium del catálogo	Categorización
37	es_cerveza_fria	Flag binario: 1 si el producto es cerveza fría (alta sensibilidad a temperatura)	Categorización
38	cluster_local	Cluster de comportamiento del local (segmentación K-means por perfil de ventas)	Categorización
39	temp_x_oferta	Temperatura promedio $\times$ tiene_oferta (captura sinergia clima-promoción: mayor efecto en calor)	Interacción
40	temp_x_premium	Temperatura promedio $\times$ es_premium (diferencial de sensibilidad climática por segmento)	Interacción
41	oferta_x_familia	tiene_oferta $\times$ familia_producto_enc (elasticidad promocional diferenciada por familia)	Interacción
42	verano_x_oferta	es_verano $\times$ tiene_oferta (efecto amplificado de promociones en temporada alta)	Interacción
43	semanas_desde_lanzamiento	Número de semanas desde que el SKU fue incorporado al catálogo (productos nuevos con histórico corto)	Limpieza/Control
44	tiene_dato_completo	Flag binario: 1 si el SKU $\times$ local tiene al menos 12 semanas de histórico disponibles para construir todos los lags	Limpieza/Control
45	target_enc_calculado_sin_test	Confirmación de que los target encodings fueron calculados exclusivamente sobre el training set (prevención de leakage)	Limpieza/Control
46	normalizado_standard_scaler	Confirmación de que la normalización (media 0, std 1) fue ajustada solo en training y aplicada en test/producción	Limpieza/Control

Nota. Tabla de elaboración propia.

Tabla de hiperparámetros

TABLA DE HIPERPARÁMETROS FINALES — BrewCast			
N°	Hiperparámetro	Valor final	Justificación / Criterio de selección
1. ARQUITECTURA DE LA RED (Topología profunda estrechante)			
1	Número de capas ocultas	4	Arquitectura profunda estrechante. Suficiente profundidad para aprender representaciones jerárquicas sin exceder la capacidad de generalización con el volumen de datos disponible.
2	Neuronas — Capa 1 (entrada)	256	Capa amplia para capturar combinaciones iniciales de los 46 features de entrada. El ancho decrece progresivamente en capas sucesivas.
3	Neuronas — Capa 2	128	Primera reducción de representación. Fuerza a la red a comprimir la información en patrones más abstractos.
4	Neuronas — Capa 3	64	Segunda compresión. Captura interacciones de nivel medio entre features temporales, comerciales y contextuales.
5	Neuronas — Capa 4 (salida oculta)	32	Representación final antes de la predicción. Integra todos los patrones aprendidos en las capas anteriores.
6	Neuronas — Capa de salida	1	Una neurona de salida: predicción de ventas en unidades para el SKU×local en la semana objetivo. Problema de regresión univariada.
7	Función de activación (ocultas)	ReLU	Rectified Linear Unit. Mitiga el problema de vanishing gradients, es computacionalmente eficiente y permite aprendizaje rápido. Estándar en redes profundas para regresión.
8	Función de activación (salida)	Lineal	Sin función de activación en la capa de salida para permitir predicciones continuas sin restricción de rango (ventas pueden ser cualquier valor positivo).
2. REGULARIZACIÓN (Prevención de overfitting)			
1	Tasa de Dropout	0.3	30% de neuronas desactivadas aleatoriamente por batch durante entrenamiento. Crea efecto de ensemble implícito y evita dependencia excesiva de neuronas específicas.
2	Aplicación del Dropout	Después de cada capa oculta	El dropout se aplica en todas las capas ocultas, no en la capa de salida. Esto maximiza la regularización sin afectar la generación de la predicción final.

3	L2 Regularización (weight decay)	$\lambda = 0.001$	Penalización sobre la magnitud de los pesos. Favorece soluciones distribuidas donde múltiples features contribuyen moderadamente, en lugar de pocas con pesos extremos.
4	Batch Normalization	Sí (después de cada capa)	Normaliza activaciones intermedias a media ≈ 0 y varianza ≈ 1 . Estabiliza el entrenamiento, permite learning rates más altos y actúa como regularizador adicional.
3. CONFIGURACIÓN DEL ENTRENAMIENTO			
1	Optimizador	Adam	Adaptive Moment Estimation. Combina momentum y RMSProp. Robusto a la elección del learning rate inicial y converge eficientemente en datasets de tamaño medio.
2	Learning rate inicial	0.001	Valor de partida estándar para Adam. Ni tan alto que genere inestabilidad ni tan bajo que requiera miles de epochs para converger.
3	Función de pérdida	MSE (Mean Squared Error)	Penaliza más fuertemente los errores grandes, lo cual es deseable en forecasting de inventario donde subestimar demanda en picos tiene alto costo operativo.
4	Métrica de monitoreo	MAE (Mean Absolute Error)	Usada como métrica de evaluación complementaria por su interpretabilidad directa en unidades de venta, facilitando la comunicación con el equipo de compras.
4. CALLBACKS DE ENTRENAMIENTO ADAPTATIVO			
1	EarlyStopping — Monitor	val_loss	Monitorea la pérdida en el conjunto de validación. Detiene el entrenamiento cuando esta métrica deja de mejorar, evitando overfitting y ahorrando tiempo de cómputo.
2	EarlyStopping — Patience	15 epochs	Tolera hasta 15 epochs sin mejora antes de detener. Suficiente paciencia para superar mesetas transitorias de aprendizaje sin esperar indefinidamente.
3	EarlyStopping — Restore best weights	Sí	Al detener, restaura automáticamente los pesos del epoch con mejor val_loss. Garantiza que el modelo final sea el mejor encontrado, no el del último epoch.
4	ReduceLROnPlateau — Monitor	val_loss	Mismo monitor que EarlyStopping. Detecta mesetas en la validación y reduce el learning rate para permitir refinamiento fino de los pesos.
5	ReduceLROnPlateau — Factor	0.5	Reduce el learning rate a la mitad cada vez que se activa. Descenso moderado que permite seguir aprendiendo sin volverse demasiado conservador.
6	ReduceLROnPlateau — Patience	7 epochs	Más corto que el EarlyStopping para actuar antes: primero reduce el LR, y solo si aún no mejora después de 15 epochs totales, detiene el entrenamiento.

7	ReduceLROnPlateau — LR mínimo	1,00E-06	Cota inferior del learning rate. Por debajo de este valor los ajustes son despreciables y continuar el entrenamiento no aporta mejora significativa.
---	-------------------------------	----------	--

Nota. Tabla de elaboración propia.