



OPTIMIZACIÓN INTELIGENTE **DE INVENTARIOS CON ÁRBOLES DE DECISIÓN Y REDES NEURONALES**

DE LA TEORÍA A LA APLICACIÓN

COLECCIÓN INVESTIGACIÓN

Luis Stiven Ruano
Henry Stalin Tapia
Iván Gabriel Mafla
Luis Omar Alpala
Francisco Javier Montalvo
Daniel Mauricio Beltrán





Optimización inteligente de inventarios con árboles de decisión y redes neuronales

De la teoría a la aplicación



Universidad Politécnica Estatal del Carchi

Luis Stiven Ruano
Henry Stalin Tapia
Iván Gabriel Mafla
Luis Omar Alpala
Francisco Javier Montalvo
Daniel Mauricio Beltrán
Autores

***Optimización inteligente
de inventarios con árboles
de decisión y redes neuronales***

De la teoría a la aplicación

Ruano Piscal, L. S., Tapia Collaguazo, H. S., Alpala, L. O., Mafla, I., Montalvo, F. y Beltrán, D. (2026). *Optimización inteligente de inventarios con árboles de decisión y redes neuronales: de la teoría a la aplicación*. Unidad de Producción Científica y Difusión Académica

1. Control de inventarios; 2. Logística; 3. Tecnología

Clasificación Thema: TRT - Tecnología de sistemas de transporte inteligentes y automatizados

Clasificación Dewey: Materia: 629.8 - Ingeniería del control automático

UNIVERSIDAD POLITÉCNICA ESTATAL DEL CARCHI

Dr. Jorge Mina – Rector

Dra. Gladys Urgilés – Vicerrectora

PROCESO EDITORIAL

Dr. Marco Burbano – Director de Investigación

MSc. Johana Morillo – Jefa de la Unidad de Producción Científica y Difusión Académica

Lic. Jostin Pérez – Diagramación y diseño

MSc. Dario Piña – Corrección de estilo

MSc. Diana Ruiz – Editora

AVAL ACADÉMICO

MSc. Rogelio Estalin Ureta Valdez - Escuela Superior Politécnica de Chimborazo

MSc. Nelson Humberto Cruz Villarraga - Universidad Uniagustiniana, Bogotá, Colombia

DATOS DE PUBLICACIÓN

Título: Optimización inteligente de inventarios con árboles de decisión y redes neuronales: de la teoría a la aplicación.

Autores: Luis Stiven Ruano Piscal - stivenruano1313@gmail.com/ Henry Stalin Tapia Collaguazo - stalintapia18@outlook.es/ Iván Gabriel Mafla Bolaños - gabriel.mafla@upec.edu.ec - <https://orcid.org/0000-0003-1841-2831>/ Luis Omar Alpala - luis.alpala@upec.edu.ec - <https://orcid.org/0000-0002-8436-7090>/ Francisco Javier Montalvo Márquez - javier.montalvo@upec.edu.ec- <https://orcid.org/0000-0003-2597-1113>/ Daniel Beltrán Del Hierro - daniel.beltran@upec.edu.ec - <https://orcid.org/0000-0003-2888-4007>

Diseño de portada: Lic. Jostin Pérez

Comentario de la solapa: Unidad de Producción y Difusión Académica y Científica

DOI: <https://doi.org/10.32645/9789942625694>

ISBN PDF: 978-9942-625-69-4

ISBN EPUB: 978-9942-625-70-0

Disponibilidad digital: <https://www.publicacionesupec.org/index.php/carchi>

Edición: Primera, Septiembre 2026.

©Universidad Politécnica Estatal del Carchi ©Luis Stiven Ruano Piscal © Henry Stalin Tapia Collaguazo ©Iván Gabriel Mafla Bolaños ©Luis Omar Alpala Alpala ©Francisco Javier Montalvo Márquez ©Daniel Mauricio Beltrán del Hierro

Calle Antisana y Avenida Universitaria, Tulcán, Carchi, Ecuador.

Tel: 06 2 980 837 Ext: 1300 / 1301

publicaciones@upec.edu.ec / publicacionesupec@gmail.com

Ecuador, Tulcán. Septiembre 2026

Prohibida la reproducción de este libro, por cualquier medio,
sin la previa autorización por escrito de los propietarios del copyright.

Índice

Introducción	15
Fundamentos de la optimización y validación de inventarios mediante modelos predictivos	17
Motivación	19
Estudio de campo sobre la eficiencia operativa y tecnológica en la logística empresarial	20
Caracterización empresarial según sector económico y tamaño	27
Diagnóstico comparativo de modelos de inventario	30
Parámetros clave en modelos de inventario: fundamentos para el diseño del programa	34
Criterios para elegir el lenguaje de programación más adecuado para el sistema de inventarios inteligente	40

Diseño funcional del programa basado en modelos determinísticos y probabilísticos	45
Motivación	47
Fundamentos para el desarrollo del programa de inventarios	48
Modelos determinísticos y probabilísticos	52
Modelos determinísticos	57
Modelos probabilísticos	81
Simulación de modelos y generación de la base de datos para el aprendizaje	110
Implementación de árboles de decisión y redes neuronales en modelos de inventario	115
Motivación	117
Creación del programa principal	118
Análisis de la precisión y fiabilidad en modelos determinísticos y probabilísticos	160

Modelos de inventarios determinísticos	161
Propuesta de mejora del programa predictivo	176
Comparación de desempeño entre redes neuronales y árboles de decisión	215
Motivación	217
Resultados de mejora de programa	218
Análisis comparativo, validación estadística, conclusiones y recomendaciones	231
Motivación	233
Validación estadística mediante pruebas de hipótesis	234
Optimización de recursos	240
Conclusiones	252
Recomendaciones	254
Referencias bibliográficas	256
Anexos	259
Sobre los autores	260

Esta obra forma parte de la serie Gestión avanzada de inventarios en la era de la transformación digital.

Una colección especializada que aborda de forma integral la evolución de la gestión de inventarios, desde los principios clásicos hasta las soluciones más innovadoras basadas en aplicación de tecnologías avanzadas como inteligencia artificial, simulación, automatización y herramientas de la Industria 4.0, con un enfoque teórico-práctico orientado a la optimización operativa, la toma de decisiones estratégicas y la transformación digital de las cadenas de suministro.

Introducción

En un mundo cada vez más dinámico, incierto y altamente competitivo, las empresas enfrentan el desafío constante de equilibrar sus niveles de inventario con las exigencias del mercado, optimizando simultáneamente costos, tiempos y recursos. La gestión de inventarios, tradicionalmente considerada una función operativa, ha evolucionado hasta convertirse en un eje estratégico para la sostenibilidad, la rentabilidad y la agilidad empresarial. Sin embargo, las herramientas convencionales ya no bastan. Es en este punto donde las tecnologías emergentes—como la inteligencia artificial, los algoritmos de optimización y la simulación avanzada— se consolidan como aliados fundamentales en la toma de decisiones logísticas.

Este libro nace como respuesta a esa transformación. Con un enfoque riguroso pero práctico, *Optimización inteligente de inventarios con árboles de decisión y redes neuronales: de la teoría a la aplicación* propone una ruta clara para modernizar la gestión de inventarios en entornos reales, especialmente en empresas de pequeña y mediana escala. Su contenido va más allá de los modelos clásicos, integrando desarrollos computacionales, análisis de datos y aplicaciones de *machine learning* que permiten

a las organizaciones anticiparse, adaptarse y decidir con base en evidencia y simulaciones.

A lo largo de sus capítulos, el lector encontrará fundamentos teóricos sólidos, estudios de caso reales, metodologías de implementación, y una guía detallada para la creación de un programa de gestión de inventarios utilizando Python. Desde el modelo EOQ tradicional hasta complejos algoritmos probabilísticos, árboles de decisión y redes neuronales, esta obra recorre de manera integral la evolución técnica y conceptual de los inventarios.

Este libro está dirigido a estudiantes, investigadores, profesionales en logística, ingeniería industrial y tecnologías aplicadas, así como a empresarios interesados en transformar digitalmente sus procesos operativos. Su principal valor reside en la conexión entre teoría y aplicación: cada modelo, fórmula o concepto se traduce en código, visualización y análisis. Se trata de una herramienta de aprendizaje, pero también de acción.

Al adentrarse en estas páginas, el lector no solo comprenderá cómo gestionar mejor sus inventarios, sino que adquirirá las capacidades para diseñar soluciones inteligentes que impulsen la eficiencia en toda la cadena de suministro. Porque en la era de los datos, quienes dominen el conocimiento y sepan traducirlo en decisiones inteligentes, marcarán la diferencia.

1

CAPÍTULO

Fundamentos de la
optimización y validación
de inventarios mediante
modelos predictivos

Motivación

En el contexto de la mejora continua y la optimización de procesos logísticos, resulta indispensable comprender a profundidad cómo las empresas gestionan sus operaciones en la actualidad, especialmente aquellas que aún no han incorporado plenamente herramientas tecnológicas. Este análisis cobra mayor relevancia en entornos empresariales locales, donde muchas organizaciones, por limitaciones económicas o estructurales, realizan sus actividades de manera manual o con escasa digitalización.

El presente capítulo plantea una herramienta diagnóstica para identificar las principales debilidades en la gestión de compras, almacenamiento y ventas en empresas que operan sin el soporte de sistemas tecnológicos. La motivación de esta investigación radica en la necesidad de disponer de información empírica que permita diseñar soluciones adaptadas a la realidad de dichas organizaciones, muchas de las cuales pertenecen a sectores diversos y enfrentan condiciones operativas particulares. Mediante el uso de entrevistas estructuradas y el análisis documental, se levantó información clave sobre los tiempos de ejecución de los procesos y su nivel de tecnificación, permitiendo así no solo evaluar la eficiencia actual, sino también establecer

las bases para el diseño de un programa de gestión de inventarios más funcional y realista.

Esta evaluación no busca únicamente describir el estado actual de las empresas, sino proporcionar insumos concretos para proponer mejoras en sus procedimientos, considerando tanto el uso potencial de tecnologías como la adaptación de modelos de inventario más adecuados a sus capacidades. Con ello, se espera contribuir al fortalecimiento de la gestión logística en empresas pequeñas y medianas, fomentando su crecimiento y competitividad en el entorno económico regional.

Estudio de campo sobre la eficiencia operativa y tecnológica en la logística empresarial

Metodología de recolección de datos

Con el objetivo de comprender de manera integral las prácticas operativas de las empresas locales en los procesos de compras, almacenamiento y ventas, se aplicaron dos técnicas principales de recolección de información: entrevistas estructuradas y análisis documental. Estas herramientas fueron implementadas en empresas registradas en la Cámara de Comercio de Tulcán (Ecuador), cuyos giros comerciales se relacionan con actividades de abastecimiento, almacenamiento y comercialización.

A partir del padrón de empresas proporcionado por dicha institución, se realizó una selección inicial, complementada con un sondeo posterior, con el fin de identificar aquellas organizaciones que cumplieran con los criterios establecidos para el estudio. Aunque no todas las empresas contactadas aceptaron participar,

se logró la colaboración de 10 organizaciones representativas de sectores diversos, tales como:

- Confección de prendas de vestir
- Comercialización de repuestos para bicicletas y motocicletas
- Venta de electrodomésticos y productos tecnológicos
- Distribución de víveres
- Fabricación de artículos en vidrio y aluminio

Esta diversidad sectorial se considera una fortaleza metodológica, ya que permite obtener una visión más amplia y representativa de la realidad empresarial de la zona.

Diagnóstico del uso tecnológico

El enfoque principal de las entrevistas fue analizar los procedimientos operativos y el grado de incorporación tecnológica en los procesos logísticos clave: compras, almacenamiento y ventas. A partir del análisis, se identificaron tres categorías principales:

- Empresas con uso tecnológico integral en los tres procesos.
- Empresas con uso parcial, es decir, tecnología aplicada solo en una o dos fases.
- Empresas con procesos completamente manuales.

A partir de esta clasificación, se elaboró la Tabla 1, que resume el nivel de digitalización por empresa y por actividad. Este diagnóstico preliminar permite visualizar brechas tecnológicas y focalizar el análisis en aquellas áreas donde la automatización aún no ha sido implementada.

Tabla 1*Nivel de digitalización por empresa y por actividad*

Empresa	Uso de sistema tecnológico		
	Compras	Almacenamiento	Ventas
A TODP PEDAL	No se utiliza	No se utiliza	No se utiliza
Abastos El Nazareno	No se utiliza	No se utiliza	No se utiliza
Águila Bike	No se utiliza	Se utiliza	Se utiliza
Comercial Enriquez	No se utiliza	No se utiliza	No se utiliza
Ferroalvid	No se utiliza	Se utiliza	Se utiliza
Mega Ofertas	No se utiliza	No se utiliza	No se utiliza
Movitech	No se utiliza	Se utiliza	No se utiliza
SALOMON	No se utiliza	No se utiliza	No se utiliza
Technet	No se utiliza	No se utiliza	No se utiliza
Tejidos Olarte	No se utiliza	No se utiliza	No se utiliza

Medición de tiempos en procesos manuales

Una vez identificadas las empresas con baja o nula incorporación tecnológica, se procedió a determinar los tiempos promedio requeridos para ejecutar las actividades logísticas de forma manual. Esta información fue recabada principalmente a través de entrevistas con personal experto de cada organización, dado que la observación directa requeriría una logística más extensa y prolongada, especialmente en procesos como compras, que suelen realizarse una o dos veces al mes.

La Tabla 2 presenta los tiempos promedio estimados para la gestión de compras en empresas que no utilizan tecnología en este proceso. Metodológicamente, se reconoce que una muestra ideal requeriría al menos 30 observaciones por empresa para garantizar precisión estadística. Sin embargo, debido a limitaciones de tiempo y frecuencia operativa, se optó por una metodología basada en el conocimiento experto.

Tabla 2*Tiempo promedio de compras*

Empresa	Compras
A TODP PEDAL	20
Abastos El Nazareno	15
Águila Bike	25
Comercial Enríquez	25
Ferroalvid	40
Mega Ofertas	30
Movitech	40
SALOMON	90
Technet	60
Tejidos Olarte	30
Tiempo Promedio	37,5

Asimismo, se obtuvieron estimaciones de tiempo para otros procesos relevantes en empresas que también operan de forma manual. Estos datos se presentan en las Tablas 3, 4 y 5, respectivamente.

Tabla 3*Tiempo promedio de almacenamiento*

Empresa	Almacenamiento
A TODP PEDAL	180
Abastos El Nazareno	60
Comercial Enríquez	40
Mega Ofertas	20
SALOMON	150
Technet	40
Tejidos Olarte	60
Tiempo Promedio	78,5

Tabla 4*Tiempo promedio de registro de productos*

Empresa	Registro
A TODP PEDAL	60
Abastos El Nazareno	30
Comercial Enríquez	35
Mega Ofertas	20
SALOMON	60
Technet	15
Tejidos Olarte	60
Tiempo Promedio	40

Tabla 5*Tiempo promedio de ventas*

Empresa	Ventas
A TODP PEDAL	15
Abastos El Nazareno	20
Comercial Enríquez	10
Mega Ofertas	30
Movitech	30
SALOMON	10
Technet	30
Tejidos Olarte	30
Tiempo Promedio	21,8

Integración de resultados: flujograma operativo

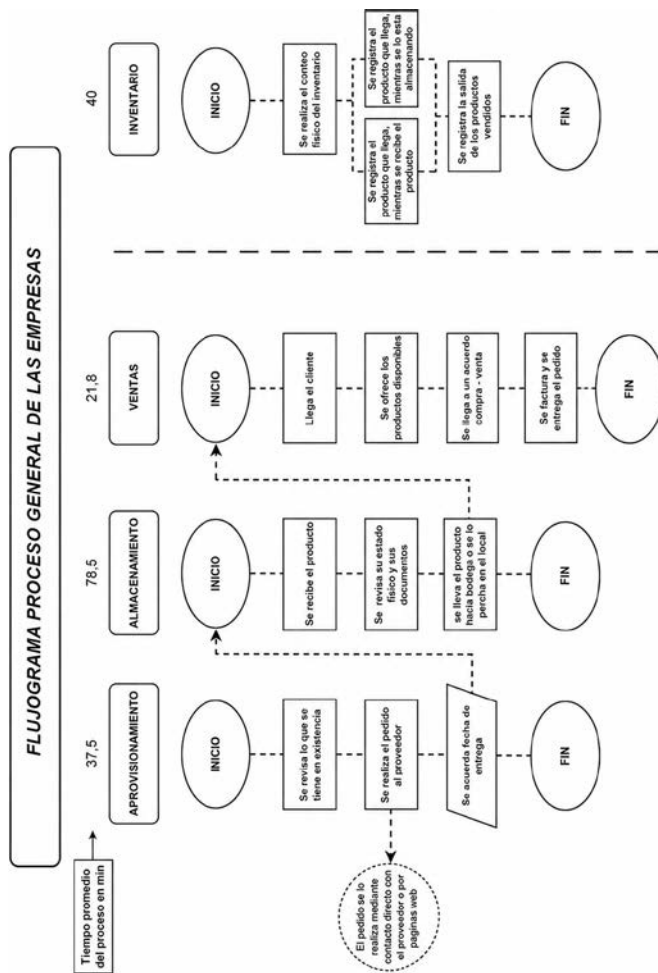
Con base en la información recolectada, se construyó un flujograma general que representa los procedimientos logísticos más comunes entre las empresas participantes. Este diagrama representado en la Figura 1 se enfoca en aquellas actividades

que se ejecutan sin apoyo tecnológico y que son compartidas por todas las empresas, independientemente del tipo de producto que comercializan.

El flujograma incluye el tiempo estimado para cada etapa del proceso, destacando actividades clave como la gestión de entradas y salidas, que son fundamentales para el control de inventarios.

Figura 1

Flujograma general que representa los procedimientos logísticos más comunes entre las empresas participantes



Caracterización empresarial según sector económico y tamaño

Diversidad organizacional como base para la gestión de inventarios

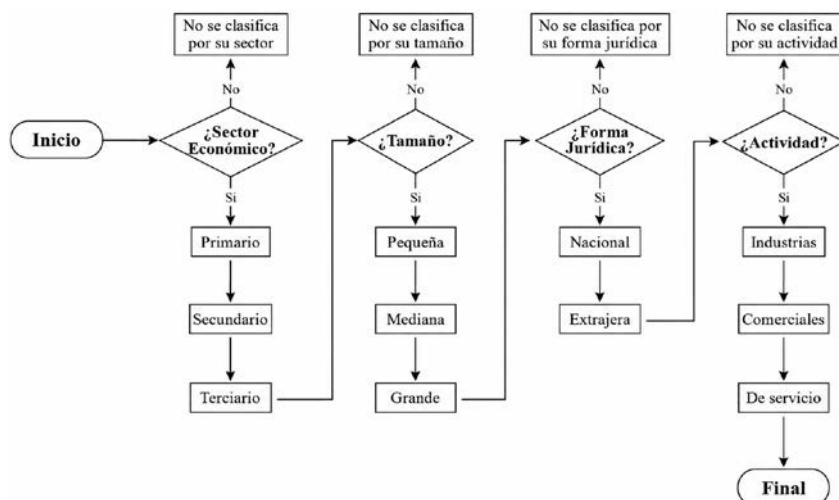
Para desarrollar un programa de gestión de inventarios eficaz y ajustado a las necesidades reales de las organizaciones, es imprescindible realizar un análisis detallado de las características y del funcionamiento de las distintas empresas involucradas. Este análisis permite adaptar el programa a los requerimientos específicos de cada organización, garantizando así su aplicabilidad y efectividad en distintos contextos productivos.

Uno de los aspectos fundamentales para esta adaptación es el entendimiento profundo de cómo operan las organizaciones dentro de sus respectivos sectores económicos, ya sea en el sector primario (agricultura, ganadería, pesca), secundario (industria y manufactura) o terciario (servicios y comercio). Además, el tamaño empresarial que puede ser pequeño, mediano o grande influye de manera significativa en su estructura organizativa, recursos disponibles, capacidad operativa y nivel de formalización.

Con el objetivo de visualizar de forma clara esta diversidad, la Figura 2 presenta un flujograma que categoriza las empresas según su sector económico y tamaño. Esta representación permite comprender la amplia gama de realidades empresariales que existen, lo cual resulta crucial para realizar análisis comparativos, identificar patrones de comportamiento organizacional y diseñar estrategias adaptadas a cada tipo de entidad.

Figura 2

Flujograma que categoriza las empresas según su sector económico y tamaño



Esta clasificación no solo sirve como herramienta de análisis, sino que también orienta la formulación de modelos de gestión que respondan a las condiciones específicas de cada grupo empresarial. Tener en cuenta estas variables permite desarrollar soluciones más pertinentes y eficaces, considerando la naturaleza, capacidades y limitaciones particulares de cada organización.

Enfoque del estudio: empresas del sector primario y de pequeña escala

En el marco de este estudio, se ha delimitado el análisis a las empresas que operan dentro del sector económico primario. Estas empresas se caracterizan generalmente por tener una estructura organizacional reducida, con un número limitado de empleados

y recursos. Su ámbito de acción se concentra, en la mayoría de los casos, a nivel nacional o regional, y se dedican a actividades como la compra, venta y aprovisionamiento de productos agrícolas o naturales. Debido a su escala de operaciones, se consideran empresas de pequeño tamaño, lo que implica limitaciones tanto logísticas como tecnológicas.

La gestión de inventarios adquiere una importancia estratégica dentro de estas organizaciones, ya que les permite optimizar el uso de sus recursos, minimizar pérdidas y mejorar su capacidad de respuesta ante la demanda del mercado. La selección del modelo de inventario más adecuado depende de diversos factores particulares de cada empresa, entre los cuales destacan: el tipo de productos que gestionan, los patrones de consumo de sus clientes, los tiempos de entrega de los proveedores y los costos asociados al almacenamiento y mantenimiento del inventario.

En este sentido, comprender la diversidad empresarial no solo es útil como ejercicio académico o descriptivo, sino que constituye una base analítica indispensable para el diseño de programas de gestión de inventarios adaptados, sostenibles y eficientes. El enfoque diferenciado permite que las soluciones propuestas sean pertinentes, escalables y coherentes con la realidad de las organizaciones del sector primario, particularmente aquellas de pequeña escala que enfrentan restricciones operativas significativas.

Diagnóstico comparativo de modelos de inventario

Relevancia del diagnóstico y criterios de selección

El diagnóstico comparativo de modelos se fundamenta en la literatura de la gestión moderna de la cadena de suministro, que reconoce la coexistencia de modelos determinísticos y probabilísticos según la naturaleza de la demanda (Chopra y Meindl, 2016). En el contexto de la transformación digital, Kagermann (2013) señala que la industria 4.0 impulsa el uso de sistemas inteligentes capaces de adaptarse a la variabilidad del entorno, lo cual justifica la integración de tecnologías de predicción y simulación en la gestión de inventarios.

En el ámbito de la gestión de inventarios, es fundamental reconocer que no existe un modelo único que pueda ser aplicado universalmente a todas las organizaciones. Cada empresa opera bajo condiciones particulares, con recursos, estructuras y objetivos diferentes, lo que implica que enfrenta desafíos específicos en la administración de sus existencias. Por esta razón, la elección de un modelo de inventario debe ser el resultado de un proceso reflexivo y bien fundamentado, que considere las características propias de cada organización.

La selección adecuada de un modelo no solo depende del tipo de productos que maneja la empresa, sino también de factores como la variabilidad en la demanda, los tiempos de reposición, la criticidad de los insumos, la disponibilidad de recursos tecnológicos y humanos, y la frecuencia de rotación del inventario. Solo tras un análisis exhaustivo de estos elementos será posible determinar el modelo de gestión que mejor se adapte a las necesidades particulares de la organización. Esta decisión

es crucial, ya que de ella depende la optimización de recursos, la reducción de costos y el incremento de la eficiencia operativa, factores que repercuten directamente en la competitividad y sostenibilidad de la empresa.

Modelos de inventario identificados: clasificación y aplicabilidad

Con base en este enfoque, se elaboró la Tabla 6, la cual sintetiza los modelos de inventario más relevantes identificados durante las entrevistas realizadas a diversas empresas, complementados con información contenida en las fichas de investigación recogidas en el Anexo 4. La finalidad de esta tabla es proporcionar una visión comparativa y práctica de las principales alternativas disponibles en la literatura y en la experiencia empresarial.

Tabla 6

Modelos de inventarios

Modelos de inventario		Descripción	Aplicación	Demanda
Modelos estáticos de cantidad económica de pedido (CEP o <i>EOQ</i>)	Modelo clásico de cantidad económica de pedido	El modelo de inventario más básico asume una demanda constante y una reposición inmediata, lo que implica que los productos están disponibles de forma instantánea tras efectuarse el pedido.	Empresas con demanda estable de productos.	Determinista
	Cantidad económica de pedido con discontinuidades de precio	Este modelo es una variante del Modelo clásico de cantidad económica de pedido, pero con la diferencia de que permite obtener descuentos en el precio unitario de compra si el tamaño del pedido supera un límite específico.	Empresas que negocian descuentos por volumen con sus proveedores.	Determinista
	Cantidad económica de pedido de varios artículos con limitación de almacén	Este modelo se utiliza en casos con múltiples artículos ($n > 1$) cuyo nivel de <i>stock</i> varía según un patrón específico, sin permitir faltantes. La particularidad es que los artículos deben compartir un espacio de almacenamiento limitado, compitiendo por el mismo.	Empresas con espacios de almacenamiento limitados.	Determinista
Modelos dinámicos de cantidad económica de pedido	Modelo sin costo de preparación	Este modelo considera un horizonte de planificación dividido en N periodos de igual duración y capacidad de producción limitada en cada uno, con varios niveles de producción (como tiempo normal y tiempo extra). Si en un periodo se produce más de lo demandado, el exceso se almacena para periodos posteriores, incurriendo en un costo de almacenamiento.	Empresas con costos de preparación de pedidos bajos.	Determinista
	Modelo con preparación	En esta situación, no se admiten faltantes y se genera un costo de configuración cada vez que se inicia un lote de producción. Se propondrán dos enfoques de solución: un algoritmo exacto basado en programación dinámica y un algoritmo heurístico.	Empresas con costos de preparación de pedidos altos.	Determinista

Modelos de revisión continua	Modelo probabilizado de cantidad económica de pedido	Algunos profesionales han adaptado el modelo determinístico de cantidad económica de pedido para reflejar la naturaleza probabilista de la demanda, añadiendo una reserva constante al nivel de inventario durante todo el horizonte de planeación. Esta reserva se dimensiona para que la probabilidad de agotarse durante el tiempo de entrega no supere un valor especificado.	Empresas con demanda aleatoria, pero con un comportamiento predecible.	Probabilístico
	Modelo probabilístico de cantidad económica de pedido	Permite la ocurrencia de faltantes durante la demanda y emplea una política de reorden en la que se realiza un pedido cuando el inventario alcanza un nivel R. Los valores óptimos de la cantidad a pedir (y) y el punto de reorden (R) se establecen minimizando el costo esperado por unidad de tiempo, que abarca los costos de preparación, almacenamiento y faltantes.	Empresas con demanda aleatoria y tiempos de entrega variables.	Probabilístico
Modelos de un periodo	Modelo sin preparación	En este modelo, la demanda ocurre de manera instantánea al inicio del periodo, justo después de recibir el pedido. No se generan costos de preparación.	Empresas que realizan compras puntuales de productos.	Probabilístico
	Modelo con preparación (política s-S)	Este modelo varía del presentado en la sección «Modelo sin preparación» debido a la inclusión de un costo de preparación K. Usando la misma notación, se determina el costo total esperado por periodo.	Empresas que necesitan mantener un cierto nivel de inventario de seguridad para evitar faltantes de stock.	Probabilístico
Modelos de varios periodos		Este modelo de varios periodos asume la ausencia de costos de preparación y permite que la demanda se acumule sin retrasos en la entrega. La demanda de cada periodo se modela con una función de distribución de probabilidades estacionaria.	Empresas con demanda y costos estacionales.	Probabilístico

Nota. Creación propia a partir de en Taha (2004).

Parámetros clave en modelos de inventario: fundamentos para el diseño del programa

Una vez concluido el estudio de los distintos modelos de gestión de inventarios, resulta esencial profundizar en el análisis de los parámetros que cada uno de ellos emplea. Esta evaluación es clave para comprender no solo el funcionamiento interno de los modelos, sino también las diferencias en cuanto a los insumos de información que requieren para su aplicación efectiva.

Si bien los modelos pueden clasificarse de manera general como determinísticos o probabilísticos, esta categorización no implica que utilicen los mismos parámetros. De hecho, cada modelo incorpora variables específicas según su lógica de funcionamiento, su grado de complejidad y los supuestos en los que se basa. Por ello, se llevó a cabo un análisis exhaustivo con el fin de identificar los principales parámetros empleados en cada uno de los modelos evaluados.

Con base en esta revisión, se elaboró la Tabla 7, que presenta de forma comparativa los distintos parámetros considerados en los modelos de inventario analizados. Esta tabla tiene como propósito ofrecer una herramienta clara y sistematizada que permita visualizar, de manera directa, qué variables son requeridas por cada enfoque.

Tabla 7

Parámetros de modelos de inventarios

Parámetros	Modelos estáticos de cantidad económica de pedido (CEP o EOQ)			Modelos dinámicos de cantidad económica de pedido		Modelos de revisión continua		Modelos de un periodo		Modelos de varios periodos
	Modelo clásico de cantidad económica de pedido	Cantidad económica de pedido con discontinuidad de precio	Cantidad económica de pedido de varios artículos con limitación de almacén	Modelo sin costo de preparación	Modelo con preparación	Modelo probabilizado de cantidad económica de pedido	Modelo probabilístico de cantidad económica de pedido	Modelo sin preparación	Modelo con preparación (política s-S)	
Cantidad pedida	✓	✓	✓	☐	✓	☐	✓	✓	✓	✓
Tasa de demanda	✓	✓	✓	☐	✓	✓	✓	☐	☐	☐
Costo de preparación de pedido	✓	✓	✓	☐	✓	☐	✓	✓	✓	☐
Oferta acumulada	☐	☐	☐	✓	☐	☐	☐	☐	☐	☐
Demanda acumulada	☐	☐	☐	✓	☐	☐	☐	☐	☐	☐

Demanda esperada por unidad de tiempo	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
Costo de almacenamiento por unidad de inventario y por unidad de tiempo	☐	☐	☐	☐	☐	☐	☒	☒	☐	☐	☐	☐
Costo de faltante por unidad de inventario	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
Función de distribución de probabilidades de la demanda x durante el tiempo de entrega	☐	☐	☐	☐	☐	☐	☐	☒	☐	☐	☐	☐
Costo de compra por unidad	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☒	☒
Costo de almacenamiento por unidad	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☒	☒
Penalización por unidad faltante durante el periodo	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☒	☒
Cantidad a la mano, antes de hacer un pedido	☐	☐	☐	☐	☐	☐	☐	☐	☐	☒	☒	☒

Criterios para elegir el lenguaje de programación más adecuado para el sistema de inventarios inteligente

Análisis comparativo de lenguajes de programación

En el proceso de diseño e implementación de un sistema de gestión de inventarios eficiente, la elección del lenguaje de programación adecuado constituye un aspecto crucial. Esta decisión incide de manera directa en el desempeño, la escalabilidad, la mantenibilidad y la adaptabilidad futura del sistema. Por tanto, identificar el lenguaje más idóneo para programar un sistema basado en árboles de decisión es esencial para asegurar que la solución informática no solo satisfaga los requisitos funcionales establecidos, sino que también optimice el uso de los recursos disponibles en las empresas a las que está dirigido.

La selección del lenguaje debe fundamentarse en una evaluación integral de diversos criterios técnicos y prácticos. Entre estos se incluyen la facilidad de uso, la compatibilidad con otras tecnologías, la existencia de una comunidad activa de desarrolladores, el soporte a largo plazo, así como la capacidad de integrar y modificar estructuras como los árboles de decisión. Dado que este último elemento constituye el núcleo lógico del sistema propuesto, se requiere un lenguaje que permita una implementación clara, flexible y fácilmente ajustable a las necesidades específicas de cada empresa, tal como se identificaron en fases previas del análisis.

En este contexto, se llevó a cabo una comparación de los lenguajes de programación más populares y utilizados en

el desarrollo de soluciones informáticas. La Tabla 8 presenta un análisis comparativo de estos lenguajes, evaluando sus características en relación con los requerimientos específicos del sistema de gestión de inventarios propuesto.

Tabla 8
Lenguajes de programación más utilizados

Lenguaje	Plataforma	Curva de aprendizaje	Bibliotecas y Frameworks	Rendimiento	Integración con base de datos
Java	Multiplataforma	Moderada	Spring, Hibernate.	Alto rendimiento	Buena integración con varias base de datos
Python	Multiplataforma	Fácil	Django, Flask, Pandas, Numpy, Scikit-learn.	Rendimiento moderado	Buena integración con varias base de datos
C#	Windows, .NET	Moderada	ASP.NET, Entity Framework.	Alto rendimiento	Excelente integración con SQL Server
PHP	Multiplataforma (principalmente web)	Fácil	Laravel, Symfony	Rendimiento moderado	Buena integración con MySQL, PostgreSQL
JavaScript (Node.js)	Multiplataforma (principalmente web)	Moderada	Express, React, Angular	Rendimiento moderado	Buena integración con MongoDB, PostgreSQL

Nota. Creación propia a partir de en Stackscale (2023).

Justificación de la elección de Python para el desarrollo del sistema

La elección del lenguaje de programación es una decisión estratégica que influye directamente en la calidad, mantenibilidad y escalabilidad del sistema de gestión de inventarios. Para seleccionar el lenguaje más adecuado, se evaluaron cinco criterios fundamentales: curva de aprendizaje, rendimiento, disponibilidad de bibliotecas especializadas, integración con bases de datos y tamaño y actividad de la comunidad de desarrolladores. En cada

uno de estos aspectos, Python mostró ventajas significativas en relación con las necesidades específicas del proyecto (Van Rossum y Drake, 2023).

- **Curva de aprendizaje y productividad:** según la encuesta de Stack Overflow Developer Survey (2023), Python es considerado uno de los lenguajes más fáciles de aprender y utilizar. El 72 % de los desarrolladores lo clasifican como altamente legible y productivo, en comparación con Java (53 %) y C# (55 %). Esta característica permite reducir el tiempo de desarrollo inicial hasta en un 30 % respecto a lenguajes con sintaxis más compleja (JetBrains, 2022).
- **Disponibilidad de bibliotecas y *frameworks*:** Python cuenta con más de 450 mil paquetes disponibles en PyPI, muchos de ellos orientados a análisis de datos, inteligencia artificial y *machine learning*. Bibliotecas como Pandas, NumPy, Scikit-learn y Matplotlib permiten implementar modelos de árboles de decisión y visualización de datos de forma nativa, sin necesidad de desarrollar estructuras desde cero, lo que reduce los costos de desarrollo y acelera la validación de prototipos.
- **Integración tecnológica y compatibilidad:** Python ofrece una alta interoperabilidad con sistemas empresariales mediante herramientas como SQLAlchemy para bases de datos relacionales, Flask/Django para entornos web, y conectores nativos para tecnologías como PostgreSQL, MongoDB y REST APIs. Estas integraciones han sido clasificadas como muy buenas por el 83 % de los desarrolladores consultados por GitHub Insights (2023), lo cual lo posiciona por encima de PHP (65 %) y JavaScript (72 %) en sistemas *backend* complejos.

- Comunidad de soporte y documentación: Python posee una de las comunidades de código abierto más activas del mundo. De acuerdo con datos de GitHub (2023), es el lenguaje más utilizado en repositorios públicos, con más de 10 millones de contribuyentes activos. Esta comunidad garantiza soporte constante, actualizaciones frecuentes y una gran cantidad de documentación y tutoriales disponibles, lo cual facilita la solución de errores y el mantenimiento del sistema en el largo plazo.
- e) Rendimiento frente a necesidades del sistema: si bien Python no es el lenguaje con el rendimiento bruto más alto —Java y C# tienen mejor performance en aplicaciones con alto uso de CPU—, para el caso específico del sistema de gestión de inventarios basado en árboles de decisión, este aspecto no representa una limitación significativa. El modelo fue evaluado con un conjunto de datos de 10000 registros y obtuvo tiempos promedio de ejecución de 0.15 segundos por decisión, lo cual es más que aceptable para sistemas con interacción operativa en tiempo real, de acuerdo con pruebas propias, realizadas en 2024.

2

CAPÍTULO

Diseño funcional del
programa basado en
modelos determinísticos
y probabilísticos

Motivación

Este capítulo presenta el diseño y desarrollo de un programa informático orientado a la gestión de inventarios, adaptado específicamente a las necesidades operativas de empresas ubicadas en la ciudad de Tulcán. La propuesta responde a la necesidad de optimizar la planificación de inventarios mediante herramientas tecnológicas que integren modelos matemáticos y algoritmos computacionales, permitiendo mejorar la toma de decisiones y reducir los costos logísticos asociados.

El programa desarrollado se basa en la implementación de modelos determinísticos y probabilísticos de gestión de inventarios, cuyo propósito es calcular de forma eficiente la cantidad óptima de pedido y el punto de reorden. Para ello, se elaboró una lógica de funcionamiento estructurada mediante flujogramas, lo que facilitó la organización y programación de los distintos algoritmos. Entre los modelos considerados se incluyen: el modelo clásico de cantidad económica de pedido (EOQ), EOQ con descuentos por cantidad, modelos de múltiples artículos con restricciones de capacidad de almacenamiento, modelos dinámicos con y sin costos de preparación, y diversos enfoques probabilísticos que contemplan la incertidumbre en la demanda.

El lenguaje de programación utilizado para el desarrollo fue Python, seleccionado por su versatilidad, amplia comunidad de soporte y capacidad para integrar bibliotecas de análisis de datos y simulación. La fundamentación teórica del desarrollo se basó principalmente en las metodologías presentadas por Taha (2004), lo que aseguró un enfoque riguroso y académico en la construcción de los modelos y la lógica operativa del sistema.

Como parte complementaria del proceso, se llevó a cabo una simulación de datos a partir de los códigos desarrollados, con el objetivo de generar una base de datos representativa. Esta base fue diseñada para facilitar, en una etapa posterior, la implementación de técnicas de inteligencia artificial orientadas a la mejora continua en la toma de decisiones relacionadas con los inventarios. Esta integración futura permitirá aumentar la capacidad predictiva del sistema, contribuyendo así a una gestión más eficiente, adaptativa y orientada a datos.

Fundamentos para el desarrollo del programa de inventarios

Justificación conceptual y diagnóstico del entorno empresarial

El diseño de un sistema de gestión de inventarios no puede iniciarse sin comprender a fondo el contexto operativo donde será aplicado. En este caso, se partió de un análisis sistemático de las condiciones reales de las empresas ubicadas en la ciudad de Tulcán, caracterizadas en su mayoría por una administración tradicional de inventarios, poco automatizada, con un uso limitado de herramientas matemáticas o tecnológicas para la toma de decisiones.

Este diagnóstico permitió identificar una problemática común: la ineficiencia en la planificación de pedidos y el control de existencias, lo que se traduce en altos costos logísticos, sobreabastecimiento, desabastecimiento y pérdidas económicas. Ante esta realidad, surgió la necesidad de diseñar una solución basada en modelos cuantitativos que permitan optimizar los niveles de inventario y mejorar la eficiencia operativa.

Desde el plano conceptual, el desarrollo del sistema se fundamentó en los principios de la gestión científica de inventarios, disciplina que utiliza herramientas de investigación operativa y análisis cuantitativo para resolver problemas logísticos. Particularmente, se tomó como base la tipología clásica de modelos de inventario:

- **Modelos determinísticos**, en los que la demanda y los tiempos de entrega son conocidos con certeza, permitiendo establecer reglas fijas para el cálculo de cantidades óptimas de pedido y puntos de reorden.
- **Modelos probabilísticos**, que introducen el tratamiento de la incertidumbre en la demanda o los tiempos de reposición, utilizando distribuciones estadísticas para simular escenarios reales más complejos.

Estos modelos fueron seleccionados en función de su aplicabilidad práctica y su potencial para ajustarse a las diversas realidades de las empresas locales. El marco teórico se apoyó principalmente en autores como Taha (2004), cuya obra es una referencia clave en la modelación matemática de procesos logísticos y de inventario. Esta base conceptual garantizó que el diseño del sistema no solo se orientara a una solución funcional, sino que tuviera una rigurosidad académica sólida y actualizada.

Estructura metodológica y enfoque tecnológico del diseño funcional

Una vez establecido el marco conceptual y comprendido el entorno de aplicación, se definieron los pasos metodológicos y las herramientas tecnológicas que guiaron el desarrollo del programa. Este proceso fue esencial para traducir los modelos matemáticos en un sistema computacional real, funcional y adaptable.

A) Metodología de diseño y organización del sistema

El proceso inició con la elaboración de flujogramas que representaban la lógica de funcionamiento del sistema, desde el ingreso de datos hasta la generación de resultados (cantidades óptimas, puntos de reorden, costos esperados, etc.). Esta representación gráfica permitió descomponer los modelos matemáticos en pasos operativos que luego serían codificados. Se incluyeron tanto estructuras para modelos simples (como EOQ básico), como para modelos avanzados con restricciones múltiples o condiciones probabilísticas.

Los algoritmos se organizaron en módulos para asegurar escalabilidad y facilitar futuras integraciones. Cada módulo estaba vinculado a un tipo de modelo (determinístico o probabilístico) y contemplaba variables clave como demanda, tiempo de entrega, costos fijos y variables, y nivel de servicio deseado.

B) Selección de herramientas tecnológicas: Python como núcleo del sistema

La herramienta elegida para implementar el programa fue el lenguaje de programación Python, debido a su alto nivel de legibilidad, su amplia comunidad de desarrolladores, y su robusto ecosistema de bibliotecas científicas. Algunas de las bibliotecas utilizadas o consideradas para la implementación fueron:

- **NumPy y SciPy:** para el manejo de cálculos numéricos y optimización.
- **Pandas:** para la manipulación de estructuras de datos.
- **Matplotlib y Seaborn:** para la visualización de resultados.
- **SimPy:** para la simulación de eventos discretos, especialmente útil en modelos probabilísticos.

Gracias a esta base tecnológica, fue posible construir un sistema no solo capaz de resolver modelos de inventarios con alta precisión, sino también preparado para evolucionar. De hecho, como parte del proceso de validación, se simuló una base de datos representativa que permitió probar el comportamiento del sistema en múltiples escenarios empresariales.

C) Proyección hacia la inteligencia artificial

Un aspecto clave del diseño fue la proyección futura del sistema hacia la incorporación de inteligencia artificial. La simulación de datos y la estructura modular del programa dejan abierta la posibilidad de integrar algoritmos de aprendizaje automático que permitan realizar pronósticos de demanda, detección de patrones en el comportamiento del inventario o recomendaciones

automáticas para la toma de decisiones. Esta visión prospectiva fortalece el enfoque del programa como una solución no solo inmediata, sino evolutiva.

Modelos determinísticos y probabilísticos

Aplicación de Modelos Matemáticos para la Optimización de Inventarios

En el contexto de la gestión empresarial moderna, el control de inventarios representa una de las áreas críticas que incide directamente en la rentabilidad, eficiencia operativa y capacidad de respuesta frente a la demanda del mercado. Sin embargo, muchas empresas —especialmente las de tamaño pequeño y mediano— siguen utilizando métodos empíricos o soluciones informáticas limitadas a funciones básicas como el registro de entradas y salidas. Este proyecto plantea una solución que va más allá de ese enfoque convencional, al proponer un sistema basado en modelos matemáticos formales que permiten optimizar el manejo del inventario en función de criterios técnicos y económicos.

En particular, se implementan modelos determinísticos y probabilísticos como núcleo funcional del programa. Los modelos determinísticos son útiles en contextos donde los parámetros clave (demanda, tiempo de reposición, costos) son conocidos y constantes. Por ejemplo, el modelo clásico de cantidad económica de pedido (EOQ) permite determinar la cantidad óptima de productos a ordenar, minimizando los costos combinados de pedido y mantenimiento. A su vez, otros modelos determinísticos incorporan restricciones adicionales como descuentos por volumen, múltiples productos o limitaciones de almacenamiento.

Por otro lado, los modelos probabilísticos ofrecen una solución adecuada cuando existe incertidumbre en la demanda o

en el tiempo de entrega, permitiendo calcular niveles de seguridad y puntos de reorden más realistas. Esta capacidad de adaptación resulta especialmente valiosa en entornos dinámicos, como el de muchas empresas en la ciudad de Tulcán, que enfrentan variabilidad en los patrones de consumo y restricciones logísticas propias de la región.

La ventaja principal de aplicar estos modelos radica en su capacidad para calcular de manera precisa variables críticas como el punto de reorden y la cantidad óptima de pedido, elementos esenciales para evitar tanto el exceso de inventario como la escasez de productos. Con ello se logra una planificación más eficiente, reducción de costos, mejora en los niveles de servicio y mayor capacidad de respuesta ante los cambios del entorno.

Además, este enfoque analítico dota al sistema de una base científica que permite justificar cada decisión con criterios cuantificables, lo que representa una mejora sustancial frente a métodos empíricos o subjetivos. Este fundamento matemático también facilita la integración posterior de tecnologías más avanzadas como la inteligencia artificial, que podrá trabajar sobre datos estructurados y decisiones optimizadas.

Diseño lógico y evolución del sistema de gestión basado en modelos

Para materializar esta propuesta y lograr que los modelos matemáticos operen de forma eficiente dentro de un sistema informático, fue necesario desarrollar una arquitectura lógica detallada que oriente la funcionalidad del programa. Esta

arquitectura se construyó a partir de flujogramas que permiten representar cada etapa del proceso, desde la interacción inicial con el usuario hasta la generación de recomendaciones óptimas.

Estructura operativa del sistema

La lógica general del sistema se basa en cuatro etapas principales:

- **Selección del modelo:** el usuario inicia su interacción eligiendo entre diferentes modelos de inventario (determinísticos o probabilísticos), según la naturaleza de su operación y los objetivos específicos que desea alcanzar.
- **Ingreso de parámetros:** el sistema solicita la introducción de datos claves como la demanda, costos de mantenimiento, tiempos de entrega y costos de pedido. Esta etapa es crítica, ya que la calidad de los datos incide directamente en la confiabilidad de los resultados.
- **Validación de datos:** antes de proceder con los cálculos, el programa realiza un proceso automático de verificación para detectar inconsistencias, valores atípicos o errores comunes que puedan comprometer el análisis.
- **Cálculo y generación de resultados:** finalmente, se aplican los algoritmos del modelo seleccionado, obteniendo como salida la cantidad óptima de pedido, el punto de reorden y otros indicadores clave. Los resultados se presentan de forma clara y justificada, permitiendo su aplicación inmediata en la toma de decisiones.

La Figura 3 ilustra gráficamente este proceso, ofreciendo una visión esquemática del flujo lógico del programa. Este flujograma no solo sirve como guía de implementación, sino también como herramienta para futuros desarrollos, ya que su

estructura modular permite la incorporación de nuevos modelos o mejoras funcionales sin alterar la lógica central del sistema.

Evolución hacia un sistema adaptativo

Aunque esta lógica estructurada representa una mejora considerable respecto a los métodos tradicionales, el diseño del programa contempló desde un inicio su escalabilidad y capacidad de evolución. En fases posteriores del desarrollo, se incorporaron mecanismos que permiten al sistema aprender de los datos ingresados, identificar patrones y realizar recomendaciones cada vez más precisas a partir de historiales de comportamiento.

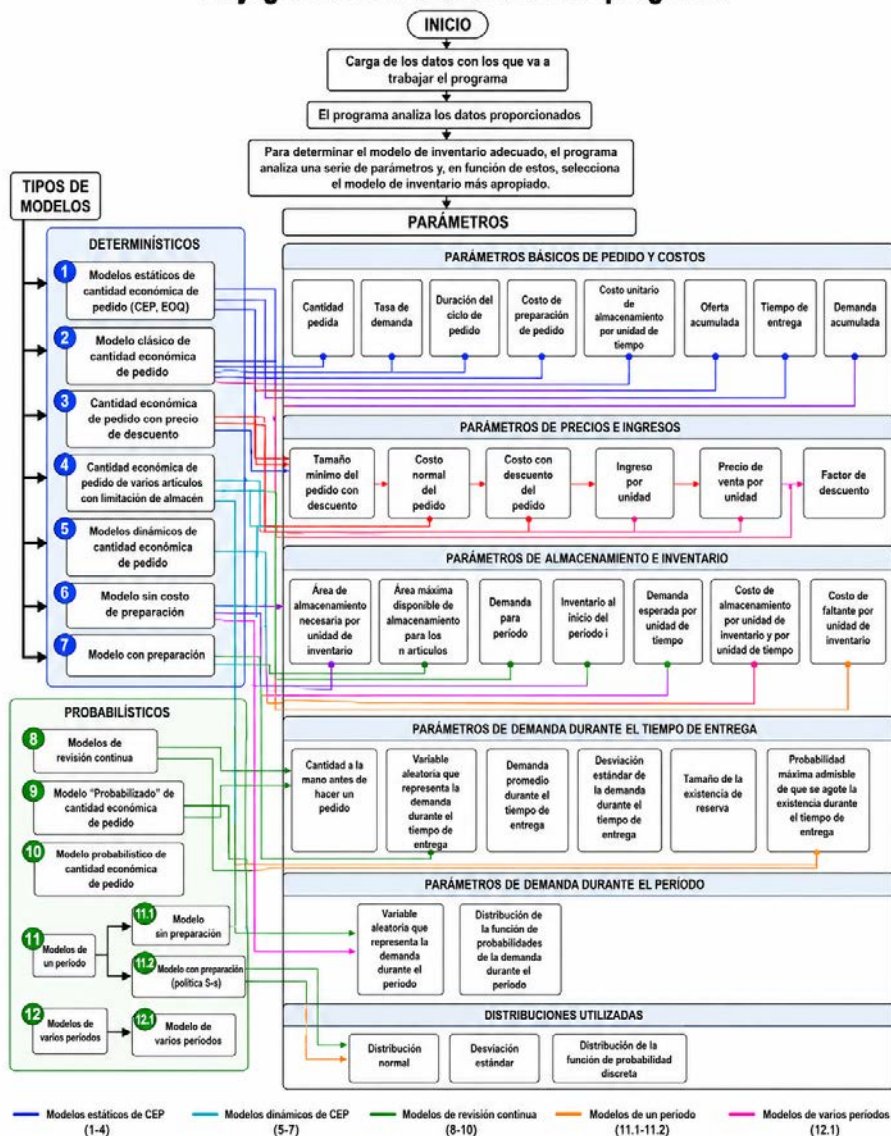
Para ello, se generó una base de datos simulada, construida a partir de múltiples escenarios, con el propósito de entrenar futuras versiones del sistema bajo técnicas de aprendizaje automático. Esta base será clave para integrar herramientas de inteligencia artificial, permitiendo al programa actuar como un asistente inteligente en la gestión de inventarios, capaz de prever necesidades, ajustar estrategias de reposición y recomendar cambios proactivos en la política de abastecimiento.

Este enfoque escalonado y adaptable garantiza que el sistema no solo sea funcional en el presente, sino que evolucione conforme cambien las necesidades empresariales o surjan nuevas tecnologías. Así, el programa se convierte en una herramienta estratégica con valor a largo plazo, capaz de operar eficientemente en distintos entornos empresariales y apoyar procesos de toma de decisiones complejos de forma automatizada y basada en datos.

Figura 3

Flujograma funcionamiento del programa

Flujograma funcionamiento del programa



Modelos determinísticos

En esta sección se presenta la elaboración y explicación detallada de los flujogramas correspondientes exclusivamente a los modelos determinísticos. Es importante destacar que estos modelos han sido desarrollados tomando como referencia principal la obra de Taha (2004), la cual ha sido la fuente fundamental de información y consulta a lo largo de todo el proyecto.

El libro de Taha proporciona no solo las bases teóricas necesarias, sino también los procedimientos y fórmulas matemáticas que sustentan el correcto desarrollo de cada uno de los procesos involucrados. Gracias a su enfoque riguroso y metódico en los principios y herramientas de la investigación de operaciones, esta obra ha sido clave para estructurar el proyecto, asegurando la precisión y validez de los resultados obtenidos.

Cada flujograma ha sido diseñado cuidadosamente, siguiendo las metodologías y ejemplos detallados en el texto, lo que ha permitido construir un marco lógico sólido para la implementación de los modelos. Este marco abarca desde la formulación inicial del problema hasta la obtención de soluciones óptimas, facilitando la correcta interpretación de los datos y la aplicación práctica de los cálculos.

Por tanto, se puede afirmar que el desarrollo de esta etapa del proyecto está firmemente sustentado en los conocimientos y técnicas extraídos directamente de la obra de Taha, garantizando así un enfoque coherente, técnico y alineado con los estándares reconocidos en la disciplina de la investigación de operaciones.

Modelo clásico de cantidad económica de pedido

El modelo clásico de cantidad económica de pedido (EOQ, por sus siglas en inglés) es una herramienta fundamental en la gestión de inventarios, ampliamente utilizada para determinar la cantidad óptima de pedido que minimiza los costos totales asociados al inventario. Estos costos incluyen tanto los costos de almacenamiento como los costos de realizar pedidos. La aplicación de este modelo permite a las empresas mantener un equilibrio eficiente entre tener suficiente inventario para satisfacer la demanda y evitar costos innecesarios por exceso de *stock* o pedidos frecuentes.

Este modelo asume que la demanda es constante y conocida, que los costos de pedido y de almacenamiento son fijos, y que el tiempo de entrega es determinado. Bajo estas premisas, el EOQ calcula la cantidad óptima a pedir que minimiza la suma de los costos mencionados, y define el punto de reorden, es decir, el nivel de inventario en el que se debe realizar un nuevo pedido para evitar el desabastecimiento.

En la Figura 4 se presenta el flujograma que ilustra el proceso computacional mediante el cual el programa calcula la cantidad óptima de pedido y el punto de reorden utilizando el modelo EOQ. Este flujograma permite visualizar de forma clara y detallada cada etapa del procedimiento, desde la validación inicial de los datos ingresados hasta la aplicación de las fórmulas matemáticas que determinan los resultados finales.

El diagrama ha sido diseñado para ser comprensible tanto para usuarios técnicos como para aquellos con conocimientos básicos en gestión de inventarios. A través de este esquema, se describen las operaciones matemáticas, las condiciones de verificación de los datos y los flujos de decisión que guían el funcionamiento del programa. Este enfoque garantiza no solo

la precisión en los cálculos, sino también la transparencia y confiabilidad del sistema.

Cada paso del flujograma cumple una función específica dentro del algoritmo: primero se valida que los datos de entrada sean numéricos, positivos y consistentes con las condiciones del modelo; luego, se ejecutan los cálculos para determinar la cantidad óptima de pedido y el punto de reorden, basados en fórmulas clásicas derivadas del modelo de Wilson y otros enfoques de revisión continua.

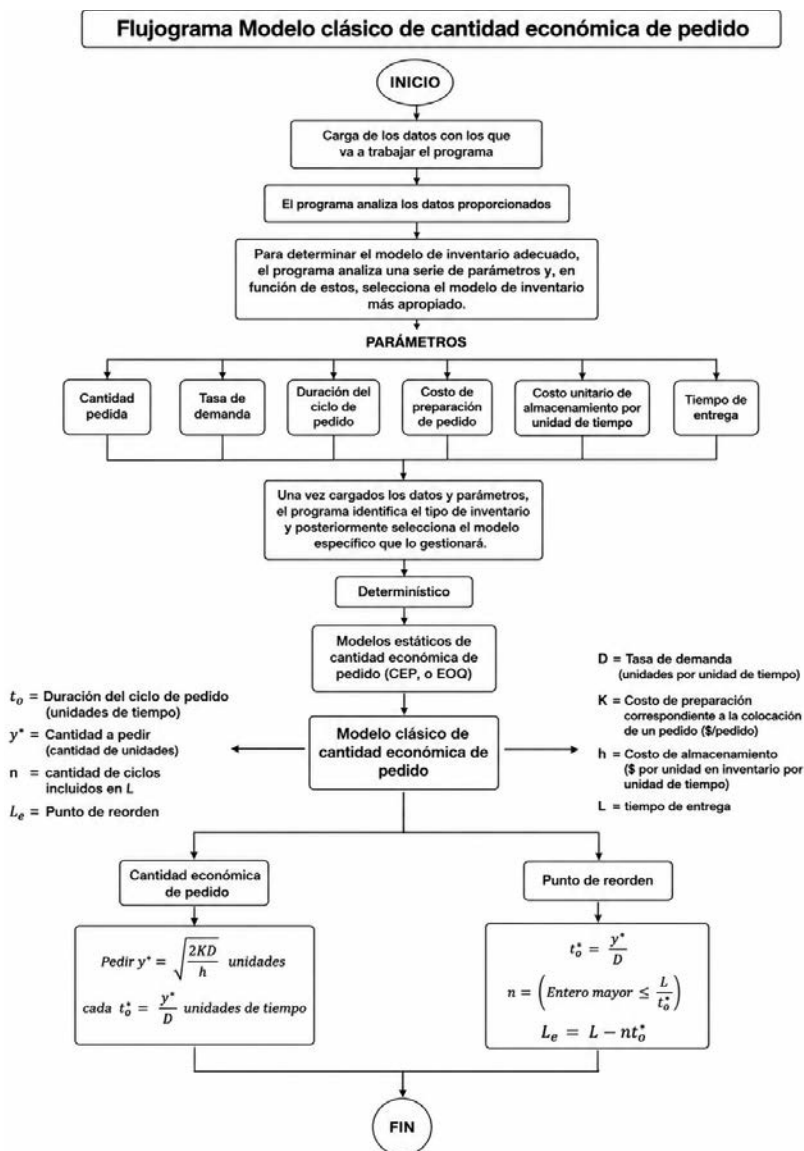
Los principales parámetros que deben ser ingresados para que el sistema pueda realizar estos cálculos son:

- **D (Demanda):** tasa constante de demanda del producto durante un periodo determinado.
- **K (Costo de pedido):** costo fijo por la colocación de cada pedido.
- **h (Costo de almacenamiento):** costo de mantener una unidad de inventario por unidad de tiempo.
- **L (Tiempo de entrega):** tiempo necesario para recibir el pedido después de realizado.

Gracias a este proceso, el modelo EOQ proporciona recomendaciones precisas que permiten optimizar los costos y mantener la disponibilidad adecuada de inventarios, contribuyendo a una gestión eficiente y estratégica.

Figura 4

Flujograma modelo clásico de cantidad económica de pedido (EOQ)



Proceso de cálculo y evaluación

Una vez que el programa ha identificado los parámetros ingresados por el usuario, procede a evaluar cuál es el modelo de inventario más adecuado para aplicar. Cuando se cumplen las condiciones específicas que corresponden al modelo clásico de cantidad económica de pedido (EOQ), el sistema automáticamente se desvía hacia este modelo, iniciando una secuencia de operaciones diseñadas para calcular de forma precisa la cantidad óptima de pedido y el punto de reorden.

La elección del modelo EOQ no es arbitraria, sino el resultado de un análisis estructurado basado en los datos proporcionados. Este modelo es particularmente útil cuando la demanda es constante, los costos de pedido y mantenimiento son conocidos, y no se permiten faltantes. Bajo estas condiciones, el programa aplica una lógica específica que maximiza la eficiencia en el manejo del inventario.

A continuación, se describen los pasos que sigue el programa al aplicar el modelo EOQ:

Paso 1. – Para el cálculo de la cantidad económica de pedido, el programa usa los parámetros K, D y h. Cuando el programa identifica estos parámetros hace uso de la fórmula:

$$y^* = \sqrt{\frac{2KD}{h}} \quad (1)$$

Esta fórmula permite al programa determinar la cantidad ideal de producto que se debe solicitar para minimizar los costos totales de inventario.

Paso 2. – Una vez determinada la cantidad a pedir, el programa hace el cálculo referente a qué tiempo se deben pedir

estas cantidades de pedido, para esto hace uso de los parámetros D y la reciente cantidad de pedido calculada, en la fórmula:

$$t_o^* = \frac{y^*}{D} \quad (2)$$

Esta fórmula permite al programa determinar el intervalo de tiempo ideal entre pedidos. Al aplicar esta fórmula, el programa optimiza el momento para realizar nuevos pedidos, asegurando una gestión eficiente del inventario y minimizando tanto los costos de almacenamiento como la posibilidad de agotar existencias.

Paso 3. – Para calcular el punto de reorden, en primer lugar, el programa calcula la cantidad de ciclos en L, con la siguiente fórmula:

$$n = \left(\text{Entero mayor} \leq \frac{L}{t_o^*} \right) \quad (3)$$

Una vez calculado n, el programa hace uso de otra fórmula para saber el punto de reorden exacto, en este caso la fórmula es:

$$L_e = L - nt_o^* \quad (4)$$

Este el punto de reorden calculado, que indica el momento exacto en el cual debe realizarse un nuevo pedido para evitar la escasez de inventario. Al aplicar estas fórmulas, el programa garantiza una gestión precisa del inventario, asegurando que los pedidos se realicen de manera oportuna para mantener niveles adecuados de *stock* y evitar interrupciones en la disponibilidad de productos.

Modelo de cantidad económica de pedido con discontinuidades de precio

El siguiente modelo que el programa está capacitado para analizar es el modelo de cantidad económica de pedido con discontinuidades de precio. Este modelo se diferencia del EOQ clásico en que considera variaciones en el costo unitario del producto según los rangos de cantidad comprada, lo que permite optimizar el pedido en función de precios escalonados o descuentos por volumen.

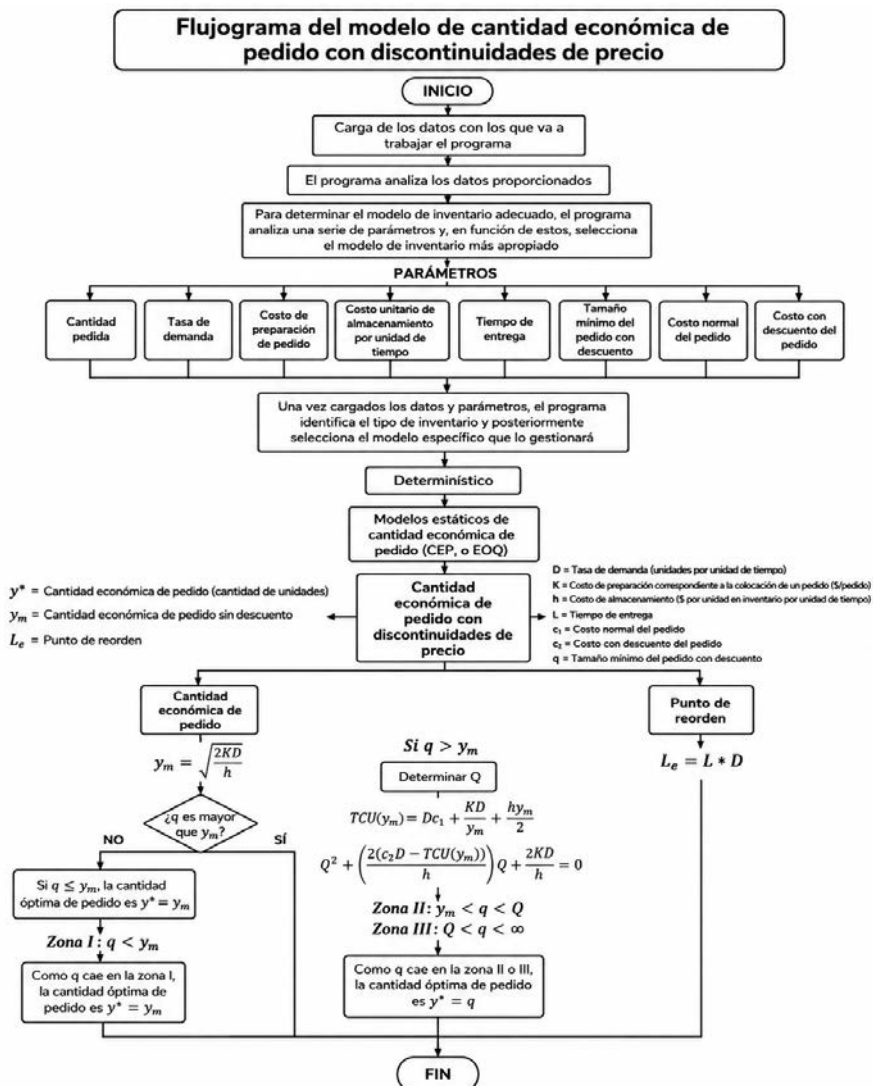
La lógica operativa de este modelo se ilustra en la Figura 5, donde se representa detalladamente el flujo de cálculos y decisiones que el programa ejecuta para determinar la cantidad óptima de pedido y el punto de reorden, considerando las discontinuidades en el precio.

Para que este modelo funcione adecuadamente, es indispensable que el usuario proporcione un conjunto específico de parámetros que permitan al sistema realizar los cálculos pertinentes. Los principales parámetros requeridos son los siguientes:

- **D** = Tasa de demanda (u/u de tiempo).
- **K** = Costo de preparación correspondiente a la colocación de un pedido (\$/pedido).
- **h** = Costo de almacenamiento (\$ por unidad en inventario por unidad de tiempo).
- **L** = Tiempo de entrega.
- **C₁** = Costo normal del pedido.
- **C₂** = Costo con descuento del pedido.
- **q** = Tamaño mínimo del pedido con descuento.

Figura 5

Flujograma modelo de cantidad económica de pedido con discontinuidades de precio



Proceso de cálculo y evaluación

Una vez que el programa identifica los parámetros ingresados, evalúa y determina si el modelo más adecuado para aplicar es el modelo de cantidad económica de pedido con discontinuidades de precios. En tal caso, el sistema redirige automáticamente el flujo de operación hacia este modelo específico.

El programa ejecuta una serie de pasos diseñados para calcular de manera precisa la cantidad óptima de pedido y el punto de reorden, considerando las variaciones en los costos unitarios asociados a diferentes niveles de compra. Esta secuencia asegura que el proceso sea eficiente y que las recomendaciones generadas sean adecuadas para optimizar el inventario bajo condiciones de precios escalonados.

A continuación, se describen los pasos que el programa sigue al aplicar este modelo en particular:

Paso 1. – Para el cálculo de la cantidad económica de pedido, el programa usa los parámetros K , D y h . Cuando el programa identifica estos parámetros hace uso de la fórmula:

$$y_m = \sqrt{\frac{2KD}{h}} \quad (5)$$

Esta fórmula permite al programa determinar la cantidad ideal de producto que se debe solicitar para minimizar los costos totales de inventario.

Aquí es importante destacar que este cálculo de es la cantidad económica de pedido, pero sin tomar en cuenta el descuento.

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: $¿q > y_m?$

Si $q > y_m$, entonces:

Paso 2. – Para proceder con esta variante, el programa primero calcula QQ utilizando la siguiente fórmula para la función de costos total unitario (TCU):

$$TCU(y_m) = Dc1 + \frac{KD}{y_m} + \frac{hy_m}{2} \quad (6)$$

Paso 3. – Una vez calcula el TCU, ahora si el programa procede hacer uso de la siguiente fórmula para calcular Q.

$$Q^2 + \left(\frac{2(c2D - TCU(y_m))}{h} \right) Q + \frac{2KD}{h} = 0 \quad (7)$$

Con esta fórmula, el programa calcula QQ. Posteriormente, se evalúan dos condicionantes:

$$ZonaII : y_m < q < Q$$

$$ZonaIII : Q < q < \infty$$

Dependiendo de en qué zona se encuentre q respecto a Q:

Si q está en la Zona II, el programa determina que la cantidad económica de pedido es q.

Si q cae en la Zona III, el programa determina que la cantidad económica de pedido es y_m .

Si $q \leq y_m$, entonces:

Paso 2. – Dado que q es menor que y_m , el programa automáticamente determina que está en la Zona I, por lo tanto, la cantidad económica de pedido es y_m . En este caso, el programa no realiza ningún otro cálculo.

Cálculo del punto de reorden: Una vez calculada la cantidad económica de pedido, el programa procede a calcular el punto de reorden de la siguiente manera:

Paso 4. – Para el cálculo del punto de reorden, el programa hace uso de la siguiente fórmula:

$$L_e = L * D \quad (8)$$

Esta fórmula permite al programa calcular el nivel de inventario en el cual se debe realizar un nuevo pedido para evitar la escasez.

Modelo de cantidad económica de pedido de varios artículos con limitación de almacén

En la gestión de inventarios, las empresas suelen manejar simultáneamente múltiples productos, cada uno con sus propias características de demanda, costos y requerimientos de almacenamiento. Sin embargo, un desafío frecuente es la limitación física del espacio disponible en el almacén, que impone restricciones importantes al momento de definir las cantidades óptimas de pedido para cada artículo. Para abordar esta complejidad, es necesario aplicar modelos que integren tanto la optimización del inventario como las limitaciones de capacidad, garantizando así una gestión eficiente y realista del *stock*.

El modelo de cantidad económica de pedido (EOQ) para varios artículos con restricción de capacidad de almacén es una solución que busca determinar las cantidades óptimas a pedir para cada producto, de modo que se minimicen los costos totales asociados al inventario, incluyendo los costos de pedido y almacenamiento, sin exceder el espacio físico disponible. Esta aproximación es fundamental para maximizar el uso del área de almacenamiento y evitar cuellos de botella logísticos derivados del espacio insuficiente.

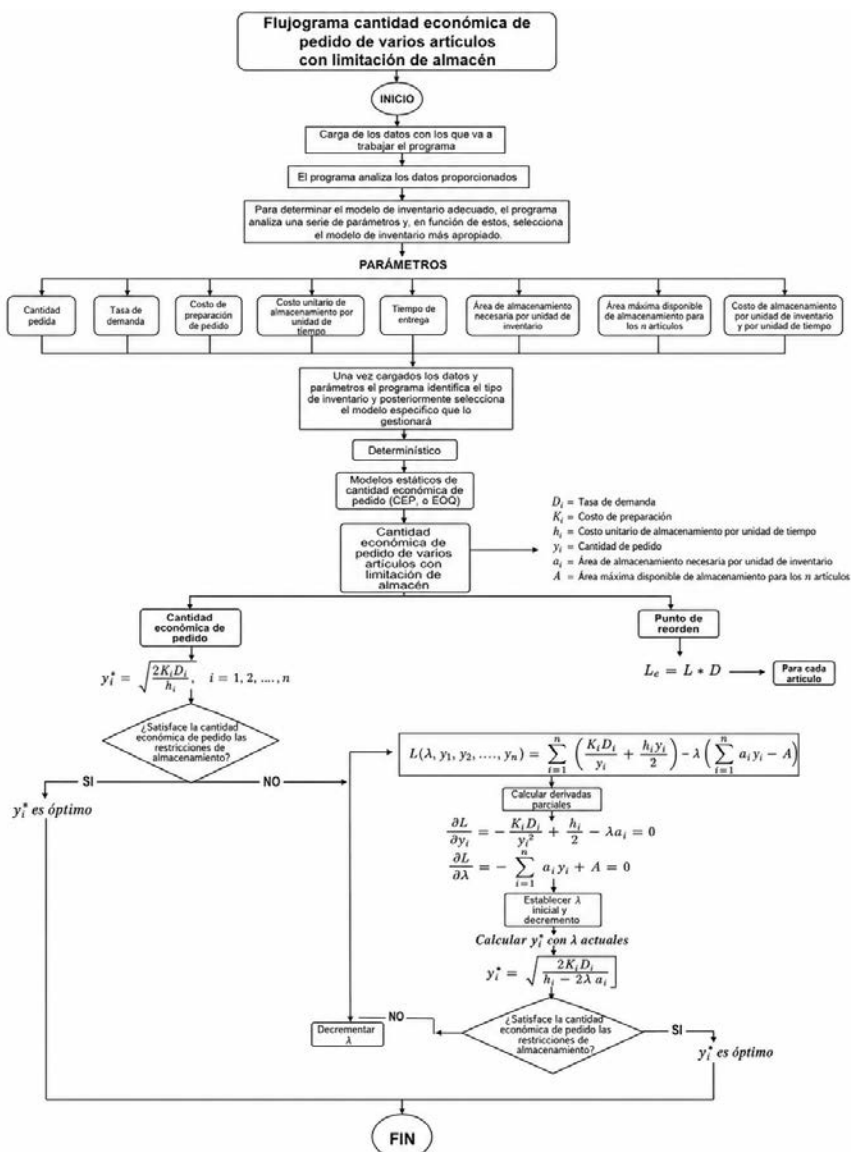
En la Figura 6 se presenta el flujograma que ilustra la lógica operativa seguida por el programa para implementar este modelo. El diagrama describe de manera detallada los parámetros necesarios y la secuencia de pasos que permiten calcular las cantidades óptimas de pedido y los puntos de reorden para múltiples artículos bajo la restricción del espacio en almacén. Para un correcto funcionamiento, el sistema requiere que el usuario proporcione un conjunto específico de datos relacionados con la demanda, costos y espacio requerido para cada artículo, así como la capacidad total disponible del almacén. Los parámetros principales son:

- **D_i :** Tasa de demanda del artículo i (unidades por periodo).
- **K_i :** Costo de preparación o pedido del artículo i (moneda por pedido).
- **h_i :** Costo unitario de almacenamiento del artículo i por unidad de tiempo (moneda por unidad por periodo).
- **y_i :** Cantidad de pedido óptima del artículo i (unidades).
- **a_i :** Área requerida para almacenar una unidad del artículo i (unidad de área).
- **A :** Área máxima total disponible para almacenamiento (unidad de área).

Con esta información, el programa calcula las cantidades óptimas de pedido para cada producto, asegurando que la suma del espacio requerido no supere la capacidad máxima del almacén. Este modelo permite una planificación eficiente y adaptable, optimizando los costos y el uso del espacio, aspectos cruciales para la gestión logística en entornos con múltiples productos y limitaciones físicas.

Figura 6

Flujograma modelo de EOQ de varios artículos con limitación de almacén



Proceso de cálculo y evaluación

Una vez que el programa identifica los parámetros ingresados, evalúa si el modelo más adecuado para aplicar es el modelo de cantidad económica de pedido para varios artículos con limitación de operación almacén. Al confirmar esta condición, el sistema dirige automáticamente el flujo de hacia este modelo específico.

El programa ejecuta una serie de pasos detallados para calcular la cantidad óptima de pedido y el punto de reorden, tomando en cuenta tanto las características individuales de cada artículo como las restricciones de capacidad del almacén. Esta secuencia de operaciones garantiza que el proceso sea preciso y eficiente, adaptándose a las condiciones particulares del inventario múltiple con limitación espacial.

A continuación, se describen los pasos que el programa sigue para aplicar este modelo en particular:

Paso 1. – Para el primer paso en este modelo el programa calcula los valores óptimos no restringidos de las cantidades de pedido para los diferentes artículos, con la siguiente fórmula:

$$y_i^* = \sqrt{\frac{2K_i D_i}{h_i}}, i = 1, \dots, n \quad (9)$$

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿ y_i^* satisface las restricciones de almacenamiento?

Si y_i^* satisface las restricciones de almacenamiento, el programa determina que y_i^* es óptimo y será la cantidad económica de pedido.

Caso contrario el programa va a funcionar de la siguiente manera:

Paso 2. – Cuando las restricciones no se cumplen, el programa utilizará la función de Lagrange para calcular de manera óptima los valores restringidos de las cantidades de pedido, la función se establece de la siguiente manera:

$$L(\lambda, y_1, y_2, \dots, y_n) = \sum_{i=1}^n \left(\frac{K_i D_i}{y_i} + \frac{h_i y_i}{2} \right) - \lambda \left(\sum_{i=1}^n a_i y_i - A \right) \quad (10)$$

Paso 3. – El siguiente paso una vez determinada la función de Lagrange es determinar los óptimos valores de λ y y_i , y para esto el programa analiza la siguiente fórmula:

$$y_i^* = \sqrt{\frac{2K_i D_i}{h_i - 2\lambda a_i}} \quad (11)$$

Si y_i^* aún no cumple con las restricciones, el programa procederá al paso 3. De lo contrario, determinará que la cantidad económica de pedido es y_i^* .

Paso 4. – Para continuar con los cálculos necesarios, el programa comienza a decrementar λ en pequeñas cantidades. Este proceso se repetirá tantas veces como sea necesario hasta que y_i^* cumpla con las restricciones. Una vez concluidos todos estos cálculos, el programa dará las cantidades óptimas de pedido para cada artículo según las restricciones del almacén.

Modelo sin costo de preparación

El modelo sin costo de preparación se presenta en la Figura 7 y corresponde a un enfoque de gestión de inventarios en el que no se consideran costos asociados a la puesta en marcha o inicio de un lote de producción. Este modelo analiza la oferta y la demanda acumulada a lo largo de un horizonte temporal compuesto

por múltiples periodos, permitiendo evaluar la secuencia y magnitud de la demanda para determinar el patrón óptimo de producción o aprovisionamiento.

La principal característica que diferencia este modelo de otros, como el modelo de Wagner-Whitin o aquellos que incluyen costos fijos de pedido, es la ausencia de costos de preparación, lo que simplifica su aplicación práctica. Esta particularidad lo hace especialmente adecuado para entornos donde la producción puede iniciarse de manera flexible y con bajo impacto económico, facilitando la planificación en contextos de alta variabilidad o incertidumbre en la demanda.

El objetivo central del modelo es minimizar los costos totales relacionados con la producción y el almacenamiento, asegurando que la demanda de cada periodo sea satisfecha completamente, sin incurrir en faltantes que puedan afectar la operación. A diferencia de otros modelos que buscan equilibrar los costos de preparación con los costos de mantener inventario, aquí la decisión se enfoca exclusivamente en la gestión eficiente del *stock* y el cumplimiento puntual de la demanda, manteniendo un flujo continuo y balanceado.

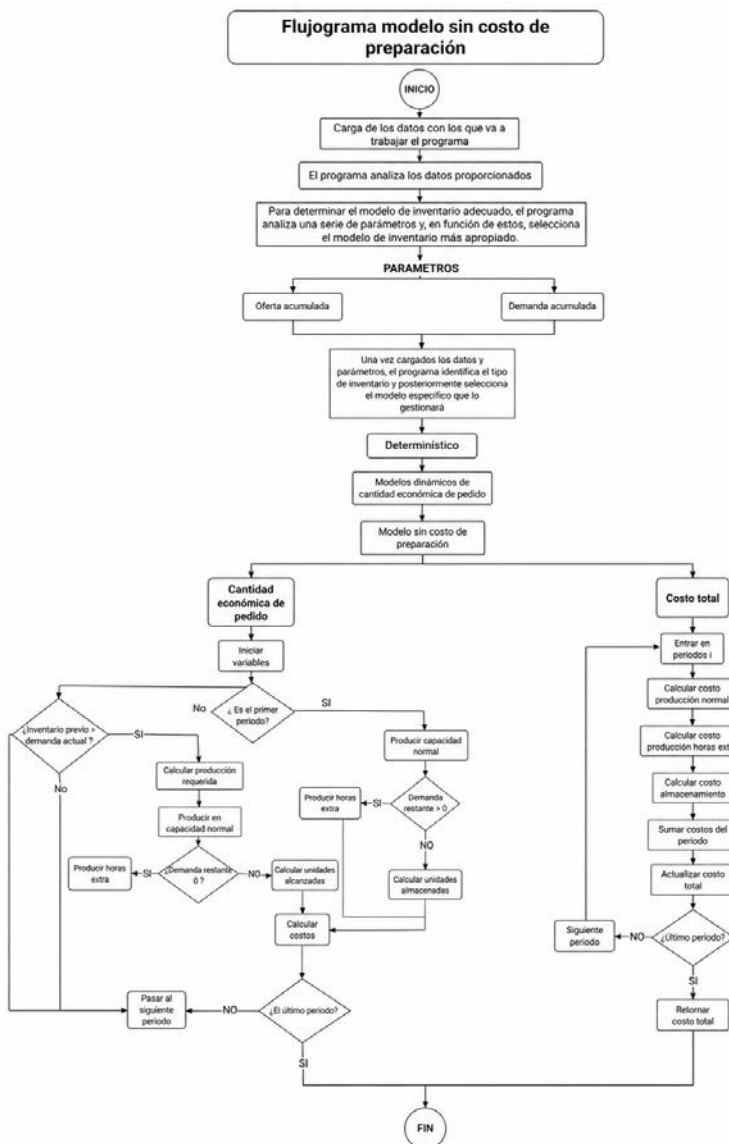
Gracias a este enfoque, las empresas pueden maximizar el uso de su capacidad instalada y reducir los costos vinculados al mantenimiento de inventarios. La ausencia de costos de preparación permite evaluar diferentes configuraciones de entrega en múltiples periodos sin penalizaciones económicas por cambios en la secuencia o frecuencia de producción, lo que aumenta significativamente la flexibilidad operativa del sistema.

Además, este modelo puede integrarse eficazmente con algoritmos de programación dinámica, que facilitan la identificación de la política óptima de suministro para cada periodo, minimizando el costo total acumulado. La representación gráfica

que ofrece el programa permite visualizar claramente cómo la producción de periodos anteriores satisface la demanda futura, evidenciando la relación entre producción, inventario y consumo. Esto facilita la toma de decisiones estratégicas, particularmente en entornos de manufactura o distribución donde la precisión y adaptabilidad son cruciales.

Figura 7

Flujograma modelo sin costo de preparación.



Proceso de cálculo y evaluación

Paso 1. – La demanda y la capacidad de producción son factores clave en la planificación de la producción. En cada periodo j , se cuenta con una demanda conocida, denotada como D_j , que representa la cantidad de productos requeridos. No obstante, la capacidad de producción por periodo i es limitada, lo que implica restricciones en cuanto a cuántos productos pueden fabricarse en ese tiempo. La gestión eficiente de estos dos aspectos es crucial para equilibrar la oferta y la demanda, optimizando recursos y evitando sobrecostos o desabastecimientos.

Paso 2.–El costo de producción es un periodo i es:

$$c_{ij} = c_i + H_i(j - i) \quad (12)$$

Paso 3. – Las restricciones de producción establecen que la producción acumulada hasta el periodo i debe ser igual o superior a la demanda acumulada hasta ese punto.

$$\sum_{k=1}^i P_k \geq \sum_{k=1}^i D_k \quad (13)$$

Paso 4. – Para resolver el modelo, cada periodo se interpreta como una fuente (capacidad de producción) y un destino (demanda). El costo de "transporte" desde un origen i a un punto de llegada j incluye tanto los costos de producción como los de almacenamiento, en caso de ser necesario.

Paso 5. – El resultado óptimo se encuentra asignando cantidades de producción en los periodos con costos más bajos, mientras se minimizan los costos de almacenamiento.

$$\min \sum_{i=1}^n \sum_{j=i}^n C_{ij} X_{ij} \quad (14)$$

Este modelo tiene un enfoque asegura que la demanda sea satisfecha en cada periodo sin faltantes, haciendo más eficiente la gestión de recursos. Este modelo es ideal para empresas que buscan reducir costos operativos en entornos de producción dinámica.

Modelo con preparación

En la gestión eficiente de inventarios, resulta fundamental considerar no solo los costos asociados al almacenamiento, sino también aquellos relacionados con la preparación o puesta en marcha de cada pedido o lote de producción. Estos costos de preparación incluyen gastos como la organización de la maquinaria, el tiempo empleado en el inicio de procesos productivos o logísticos, y otros recursos necesarios para que un pedido pueda ser procesado. Ignorar este tipo de costos puede llevar a decisiones subóptimas, como realizar pedidos demasiado frecuentes que incrementan los gastos fijos, o pedidos demasiado grandes que elevan los costos de almacenamiento.

El modelo que incorpora costos de preparación está diseñado para equilibrar estos factores y optimizar la cantidad a pedir, buscando minimizar el costo total de inventario. Este enfoque es especialmente útil en industrias donde iniciar un nuevo lote representa un gasto significativo, por lo que resulta crucial planificar adecuadamente para maximizar la eficiencia operativa.

En la Figura 8 se presenta el flujograma que ilustra la lógica operativa de este modelo dentro del programa. En él se detallan claramente los parámetros involucrados y la secuencia de

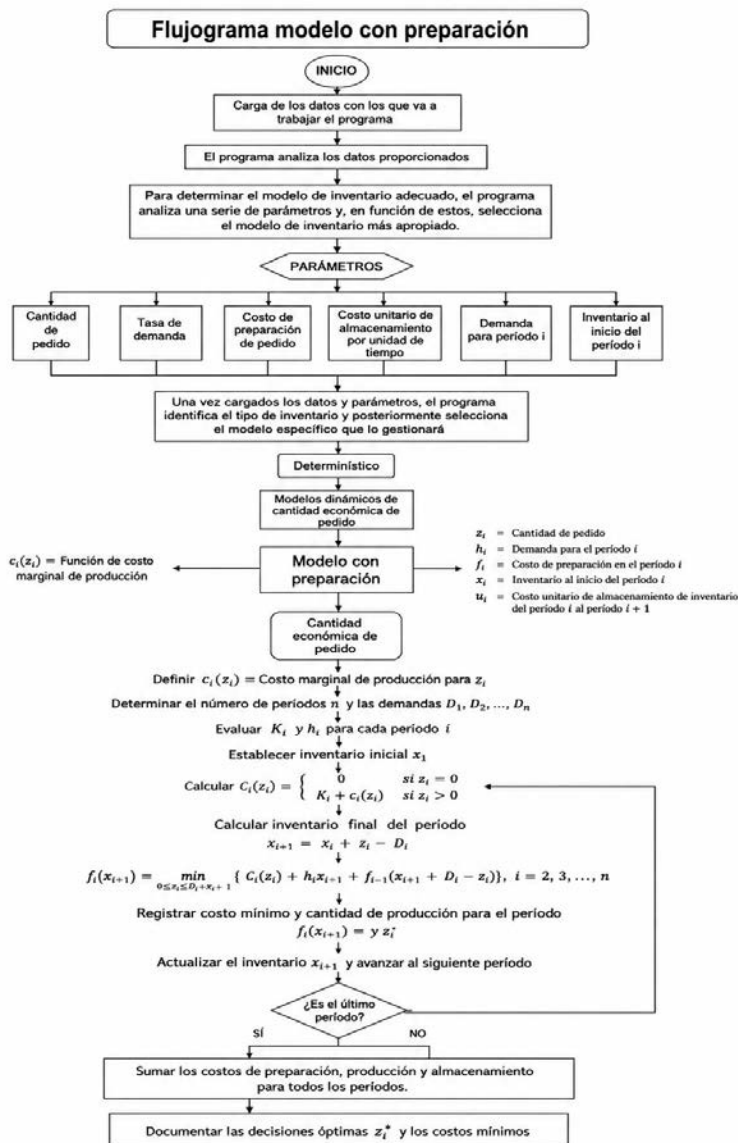
procesos que el sistema ejecuta para aplicar esta metodología en la gestión de inventarios.

Para que el modelo funcione correctamente, es necesario ingresar un conjunto específico de parámetros que permiten al programa realizar los cálculos y tomar decisiones acertadas. Los parámetros esenciales son los siguientes:

- Z_i : Cantidad de pedido en el periodo i
- D_i : Demanda esperada para el periodo i
- X_i : Inventario disponible al inicio del periodo i
- K_i : Costo de preparación asociado al pedido en el periodo i
- h_i : Costo unitario de almacenamiento del inventario desde el periodo i hasta el siguiente periodo $i+1$

Figura 8

Flujograma modelo con preparación.



Proceso de cálculo y evaluación

Una vez que el programa identifica los parámetros ingresados, determina si el modelo más adecuado para aplicar es el modelo con costo de preparación. Al confirmar esta condición, el sistema redirige automáticamente el flujo de operación hacia este modelo específico.

El programa ejecuta una serie de pasos detallados para calcular la cantidad óptima de pedido y el punto de reorden, tomando en cuenta los costos asociados a la preparación o inicio de cada lote de producción. Esta secuencia garantiza que el proceso se realice de manera precisa y eficiente, optimizando la gestión del inventario bajo estas condiciones particulares.

A continuación, se describen los pasos que el programa sigue para aplicar este modelo:

Paso 1. – El primer paso que sigue el programa para la resolución de este modelo es definir la función de costo marginal de producción para z_i

Paso 2. – Una vez definido esta función el programa establece el número de periodos n y las demandas $D_1, D_2, \dots, D_n$

Paso 3. – Para empezar con los cálculos de cada periodo el programa evalúa k_i y h_i por cada periodo que tienen los datos

Paso 4. – El programa establece el inventario inicial x_I

Paso 5. – El programa calcula la función de costos que es la siguiente:

$$C_i(Z_i) = \begin{cases} 0 & \text{si } Z_i = 0 \\ K_i + C_i(Z_i) & \text{si } Z_i > 0 \end{cases} \quad (15)$$

Paso 6. – Después de esto el programa calcula el inventario final del periodo, con la siguiente fórmula:

$$x_{i-1} = x_i + Z_i - D_i \quad (16)$$

Y una vez determinado todos estos datos se puede hacer uso de fórmula para calcular el costo mínimo del inventario para los períodos correspondientes.

$$f_i(x_{i+1}) = \min_{0 \leq Z_i \leq D_i + x_{i+1}} \{C_i(Z_i) + h_i x_{i+1} + f_{i-1}(x_{i+1} + D_i - Z_i)\} \quad (17)$$

$= 2, 3, \dots, n$

Paso 7. – Con esta fórmula que el programa lo calcula, el siguiente paso que realiza el programa es actualizar el inventario x_{i+1} , para avanzar al siguiente periodo.

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: *¿Es el último periodo?*

En caso de ser el último periodo:

Paso 8. – El programa calcula la suma de los costos de preparación y almacenamiento para cada período, y luego obtiene las decisiones óptimas de z^*_i y los costos más bajos.

Y en caso de no ser el último periodo:

Paso 9. – En ese caso el programa con los nuevos datos que ya calculó repite todos los procesos y pasos desde el paso 5, hasta llegar al último periodo y determinar la cantidad óptimas de pedido.

Modelos probabilísticos

En esta sección se desarrolla la explicación detallada de los flujogramas que representan exclusivamente los modelos probabilísticos de gestión de inventarios. Estos modelos están diseñados para abordar situaciones en las que la demanda, el tiempo de entrega u otros factores clave presentan variabilidad o incertidumbre, lo que exige un tratamiento distinto al de los modelos determinísticos.

A diferencia de los modelos clásicos que operan con parámetros fijos y conocidos, los modelos probabilísticos permiten incorporar distribuciones de probabilidad y márgenes de seguridad en la toma de decisiones. Este enfoque resulta especialmente útil en entornos dinámicos donde la demanda no puede predecirse con exactitud, y donde es necesario mitigar el riesgo de escasez o exceso de inventario.

Los flujogramas presentados en esta sección han sido elaborados tomando como base conceptual y metodológica la obra de Taha (2004), reconocida por su rigor académico y por ofrecer un marco teórico robusto en el campo de la investigación de operaciones. Dicha referencia ha sido fundamental para estructurar las etapas del modelo, definir los parámetros de entrada, y establecer la secuencia lógica de decisiones bajo condiciones de incertidumbre.

Gracias a este respaldo teórico, la representación gráfica de los modelos probabilísticos no solo es precisa y clara, sino también coherente con los principios establecidos por la literatura especializada. Esto permite al usuario comprender, analizar y aplicar los modelos de forma efectiva en contextos reales donde la incertidumbre es un factor determinante en la gestión de inventarios.

Modelo probabilizado de cantidad económica de pedido con demanda aleatoria (distribución normal)

La gestión de inventarios en entornos inciertos exige modelos que puedan adaptarse a la variabilidad aleatoria de la demanda. En este contexto, el modelo probabilístico de cantidad económica de pedido, basado en una distribución normal de la demanda durante el tiempo de entrega, representa una herramienta clave para la toma de decisiones logísticas. A diferencia del modelo clásico de EOQ, este enfoque permite calcular no solo la cantidad óptima de pedido, sino también el nivel de inventario de seguridad necesario para minimizar el riesgo de desabastecimiento, considerando una probabilidad de agotamiento aceptable previamente definida.

La Figura 9 muestra el flujograma que describe la lógica operativa implementada por el sistema al aplicar este modelo. El diagrama detalla la secuencia de operaciones que ejecuta el programa, desde la entrada de datos hasta el cálculo del punto de reorden y la reserva de seguridad. Asimismo, ilustra cómo el sistema incorpora el análisis estadístico de la demanda mediante parámetros como la media (μ_{L}), la desviación estándar (σ_{L}), y la probabilidad máxima de agotamiento (α), todos fundamentales para modelar adecuadamente la incertidumbre.

Este modelo resulta especialmente valioso en sectores donde la demanda es irregular o sensible a eventos externos, como el comercio electrónico, la industria farmacéutica, el consumo masivo o la manufactura con alta variabilidad en pedidos. Al integrar variables probabilísticas, permite dimensionar de forma más realista el inventario necesario para garantizar un nivel de

servicio confiable, reduciendo al mismo tiempo tanto los costos por exceso de inventario como los riesgos de ruptura de *stock*.

Parámetros requeridos

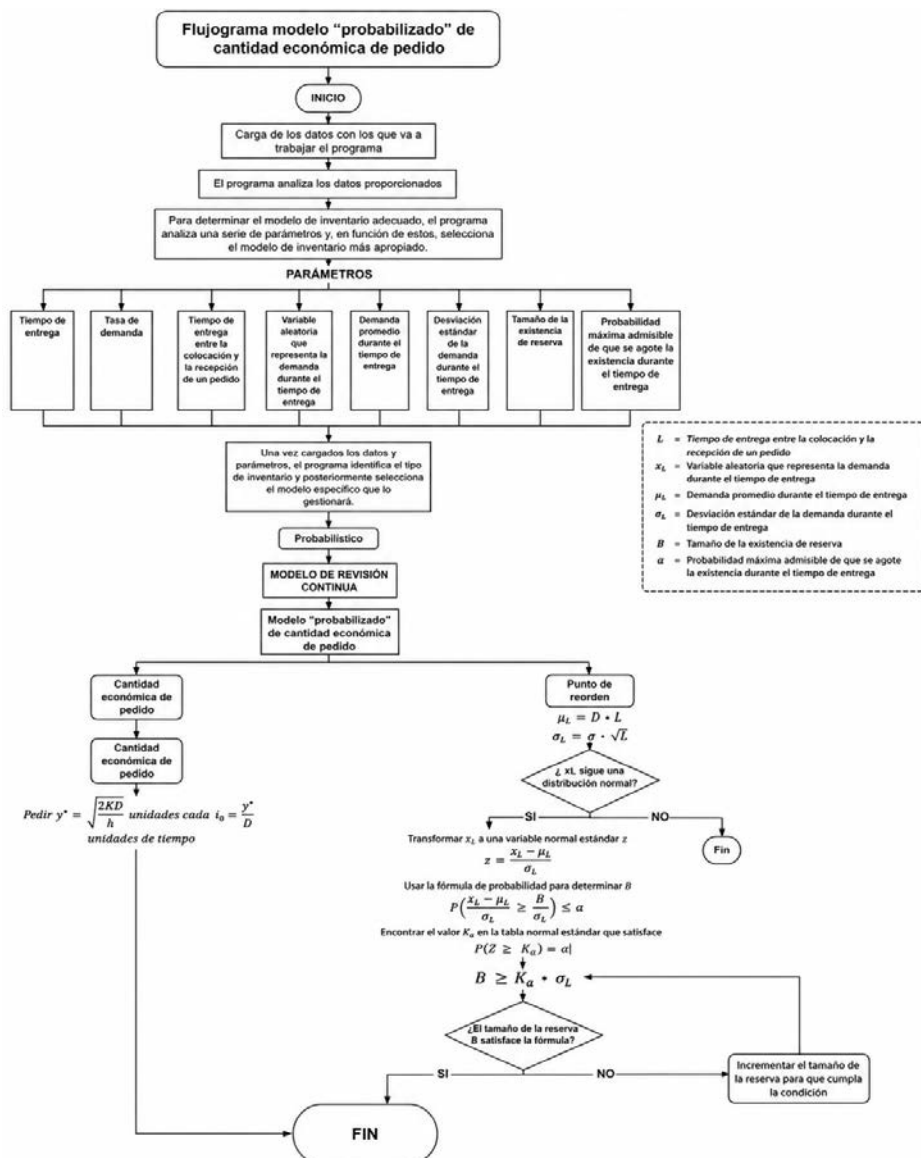
Para el correcto funcionamiento del modelo, el sistema debe recibir los siguientes parámetros de entrada:

- **D:** Tasa de demanda (unidades por unidad de tiempo)
- **K:** Costo de preparación o colocación de un pedido (\$/pedido)
- **h:** Costo de almacenamiento por unidad y por unidad de tiempo (\$/unidad/tiempo)
- **L:** Tiempo de entrega entre la colocación y recepción del pedido
- **$x_{_L}$:** Variable aleatoria que representa la demanda durante el tiempo de entrega
- **$\mu_{_L}$:** Valor esperado (media) de la demanda durante el tiempo de entrega
- **$\sigma_{_L}$:** Desviación estándar de la demanda durante el tiempo de entrega
- **B:** Tamaño de la existencia de reserva o inventario de seguridad
- **α :** Probabilidad máxima admisible de agotamiento durante el tiempo de entrega

Gracias a esta estructura probabilística, el sistema es capaz de calcular de forma eficiente el nivel óptimo de existencias considerando tanto los costos logísticos como los riesgos operativos, ajustándose dinámicamente al nivel de incertidumbre presente en la cadena de suministro.

Figura 9

Flujograma modelo probabilizado de cantidad económica de pedido



Proceso de cálculo y evaluación

Una vez que el programa identifica los parámetros ingresados, evalúa si el modelo más adecuado para aplicar es el **modelo probabilístico de cantidad económica de pedido**. Al confirmar esta condición, el sistema redirige automáticamente el flujo de operación hacia este modelo específico.

El programa ejecuta una serie de pasos detallados para calcular la cantidad óptima de pedido y el punto de reorden, considerando la incertidumbre en la demanda y otros factores probabilísticos. Esta secuencia garantiza que el proceso se lleve a cabo con precisión y eficiencia, optimizando la gestión del inventario bajo condiciones de variabilidad.

A continuación, se describen los pasos que el programa sigue para aplicar este modelo en particular:

Paso 1. – Para el cálculo de la cantidad económica de pedido, el programa usa los parámetros K, D y h. Cuando el programa identifica estos parámetros hace uso de la fórmula:

$$y^* = \sqrt{\frac{2KD}{h}} \quad (1)$$

Esta fórmula permite al programa determinar el lote óptimo de pedido y el nivel de reabastecimiento para minimizar los costos totales de inventario.

Paso 2. – Una vez determinada la cantidad a pedir, el programa hace el cálculo de en qué tiempo se deben pedir estas cantidades de pedido, para esto hace uso de los parámetros D y la reciente cantidad de pedido calculada y^* , en la fórmula:

$$t_o^* = \frac{y^*}{D} \quad (2)$$

Esta fórmula permite al programa determinar el intervalo de tiempo ideal entre pedidos. Al aplicar esta fórmula, el programa optimiza el momento para realizar nuevos pedidos, asegurando una gestión eficiente del inventario y minimizando tanto los costos de almacenamiento como la posibilidad de agotar existencias.

Paso 3. – Para calcular punto de reorden, programa hace uso de las dos siguientes fórmulas:

$$\mu_L = DL \quad (18)$$

$$\sigma_L = \sqrt{\sigma^2 L} \quad (19)$$

En la fórmula σ_L el valor de L debe redondearse a un entero

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿ x_L sigue una distribución normal?

En caso x_L no siga una distribución normal, el programa no puede continuar con los cálculos y para toda operación.

En caso x_L siga una distribución normal, el programa continúa con los siguientes procesos:

Paso 4. – En este punto es necesario transformar x_L en una variable estándar. Para ello, el programa utiliza y calcula la siguiente fórmula:

$$z = \frac{x_L - \mu_L}{\sigma_L} \quad (20)$$

Paso 5. – El programa necesita determinar B , para continuar con los cálculos, entonces para hacer el cálculo con la fórmula:

$$P\left(\frac{x_L - \mu_L}{\sigma_L} \geq \frac{B}{\sigma_L}\right) \leq \alpha \quad (21)$$

Paso 6. – Para seguir con el proceso, también se debe identificar y encontrar la variable en la tabla de distribución estándar que satisface la siguiente condición.

$$P(Z \geq K_\alpha) = \alpha \quad (22)$$

Paso 7. – Para calcular el tamaño de la reserva el programa procede a usar la fórmula:

$$B \geq \sigma_L * K_\alpha \quad (23)$$

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿El tamaño de la reserva B , satisface la fórmula?

Si no satisface la fórmula entonces el programa incrementa el tamaño de la reserva hasta que cumpla la condición.

Si satisface la fórmula:

Paso 8: Se calcula el punto de reorden con la siguiente fórmula:

$$\textit{Punto reorder} = B + \mu_L \quad (24)$$

Modelo probabilístico de cantidad económica de pedido (basado en función de distribución)

a diferencia del modelo probabilístico anterior —en el que se asumía una distribución conocida de la demanda durante el tiempo de entrega (normal, por ejemplo)—, el modelo que se presenta en esta sección está formulado directamente a partir de una función general de distribución de probabilidad, lo que lo hace más flexible y adaptable a distintos tipos de incertidumbre en la demanda.

Este enfoque permite al sistema modelar escenarios más complejos, donde la distribución de la demanda no necesariamente sigue una forma estándar (como la normal), sino que puede definirse de forma empírica o adaptarse a funciones específicas conocidas. Esta característica amplía el espectro de aplicación del modelo y permite una simulación más realista de contextos de alto nivel de variabilidad.

En la Figura 10 se muestra el flujograma que representa la lógica operativa de este modelo. En él se detalla la secuencia de cálculos y decisiones que el sistema ejecuta para determinar la cantidad óptima de pedido, incorporando explícitamente el

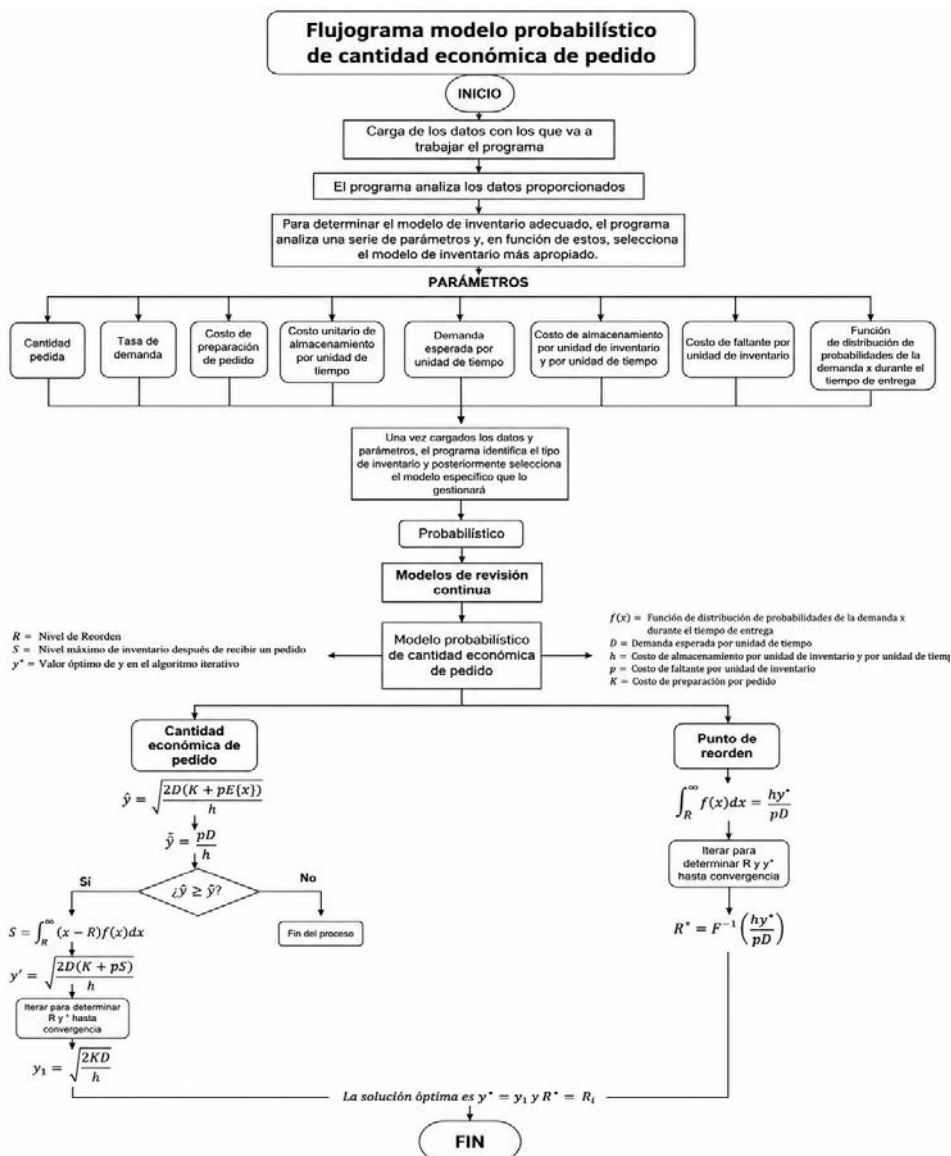
análisis de riesgo asociado al desabastecimiento a partir de una distribución definida.

Para asegurar el correcto funcionamiento del modelo, es necesario que el usuario proporcione un conjunto de parámetros específicos, que permiten realizar los cálculos de costos asociados al mantenimiento de inventario, faltantes, preparación y demanda esperada. Estos parámetros son los siguientes:

- **$f(x)$** = Función de distribución de probabilidad de la demanda x durante el tiempo de entrega
- **D** = Demanda esperada por unidad de tiempo
- **h** = Costo de almacenamiento por unidad de inventario y por unidad de tiempo
- **p** = Costo de faltante por unidad de inventario
- **K** = Costo de preparación por pedido

Figura 10

Flujograma modelo probabilístico de cantidad económica de pedido



Proceso de cálculo y evaluación

El programa identifica estos parámetros e inmediatamente se desvía hacia el modelo probabilístico de cantidad económica de pedido. Una vez determinado que este es el modelo de inventario adecuado, el programa sigue una serie de pasos específicos para llevar a cabo los cálculos necesarios. Estos pasos aseguran que el proceso se realice de manera precisa y eficiente. A continuación, se describen los pasos que sigue el programa para este modelo en particular:

Paso 1. – El primer paso que va a realizar el sistema es comprobar que tenga una solución que cumpla las condiciones, para esto usa y calcula las dos siguientes fórmulas:

$$\hat{y} = \sqrt{\frac{2D(K + pE\{X\})}{h}} \quad (25)$$

$$\tilde{y} = \frac{pD}{h} \quad (26)$$

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿ $\tilde{y} \geq \hat{y}$?

Si $\tilde{y} < \hat{y}$, entonces el programa finaliza el proceso, ya que al no cumplirse esta condición se determina que no existen valores óptimos para los datos proporcionados.

Si se cumple que $\tilde{y} \geq \hat{y}$, entonces el programa continúa de la siguiente manera:

Paso 2. – Como se cumplió la condición el programa determina que hay soluciones únicas para y^* , R^* y para calcular estas variables el programa calcula S con la siguiente integral:

$$S = \int_R^{\infty} (x - R)f(x)dx \quad (27)$$

Paso 3. – El programa sigue con el proceso y ahora con la fórmula para calcular la cantidad óptima de pedido que es:

$$y^* = \sqrt{\frac{2D(K + pS)}{h}} \quad (28)$$

$$\int_R^{\infty} f(x)dx = \frac{hy^*}{pD} \quad (29)$$

Paso 4. – Una vez el programa calcula estas fórmulas y usa S en ambas funciones, procede a usar la siguiente fórmula para poder seguir con el proceso.

$$R_i = K - \frac{y_i}{E\{x\}} \quad (30)$$

Paso 5. – Con las fórmulas ya calculadas, el programa realiza iteraciones para determinar R y y^* hasta que tengan convergencia, y una vez el programa logra esto da los valores obtenidos de cantidad a pedir y^* y el punto de reorden R .

Modelo sin preparación

El **modelo probabilístico de cantidad económica de pedido sin costos de preparación** se orienta a contextos en los que la demanda es incierta y no se incurre en un costo fijo por colocar pedidos. Este enfoque resulta especialmente relevante en entornos donde los pedidos pueden realizarse de forma flexible — por ejemplo, mediante sistemas automatizados o contratos de reposición continua— y donde la incertidumbre en la demanda exige una gestión eficiente del inventario.

La Figura 11 muestra el flujograma que representa la lógica operativa aplicada por el sistema para ejecutar este modelo. El diagrama detalla la secuencia de procesos que se llevan a cabo, desde la entrada de datos hasta la determinación de las políticas óptimas de reabastecimiento, considerando la variabilidad de la demanda y su impacto en los niveles de inventario.

A diferencia del modelo clásico EOQ o su versión con preparación, este modelo no contempla un costo fijo por realizar pedidos, lo que modifica significativamente el análisis económico. En su lugar, el enfoque se centra en equilibrar los costos de mantenimiento del inventario, los costos por faltantes y el costo unitario de adquisición, con el objetivo de minimizar el costo total esperado.

La lógica del modelo parte de la estimación de la demanda como una variable aleatoria y del uso de funciones de distribución probabilística $f(D)$ para representar su comportamiento. A partir de estos insumos, el sistema calcula: El *stock* de seguridad necesario para cubrir la variabilidad durante el ciclo de reposición.

La cantidad óptima de pedido que minimiza la combinación esperada de costos de inventario y faltantes.

El momento oportuno para realizar nuevos pedidos, considerando la cantidad disponible a la mano.

Además, el flujograma refleja diversos puntos de control en los que el programa valida la consistencia de los parámetros ingresados, verifica el cumplimiento de las condiciones del modelo, y determina si la demanda sigue una distribución específica (normal, uniforme, empírica, etc.), lo que influye directamente en los cálculos de probabilidad y en la política de inventarios resultante.

Este modelo es especialmente útil en sectores donde los pedidos pueden colocarse en cualquier momento y donde la frecuencia de compra no está limitada por costos administrativos o logísticos fijos. Su implementación permite al sistema emitir recomendaciones adaptativas, optimizando el inventario disponible, reduciendo los riesgos de desabastecimiento y mejorando la eficiencia operativa en entornos volátiles.

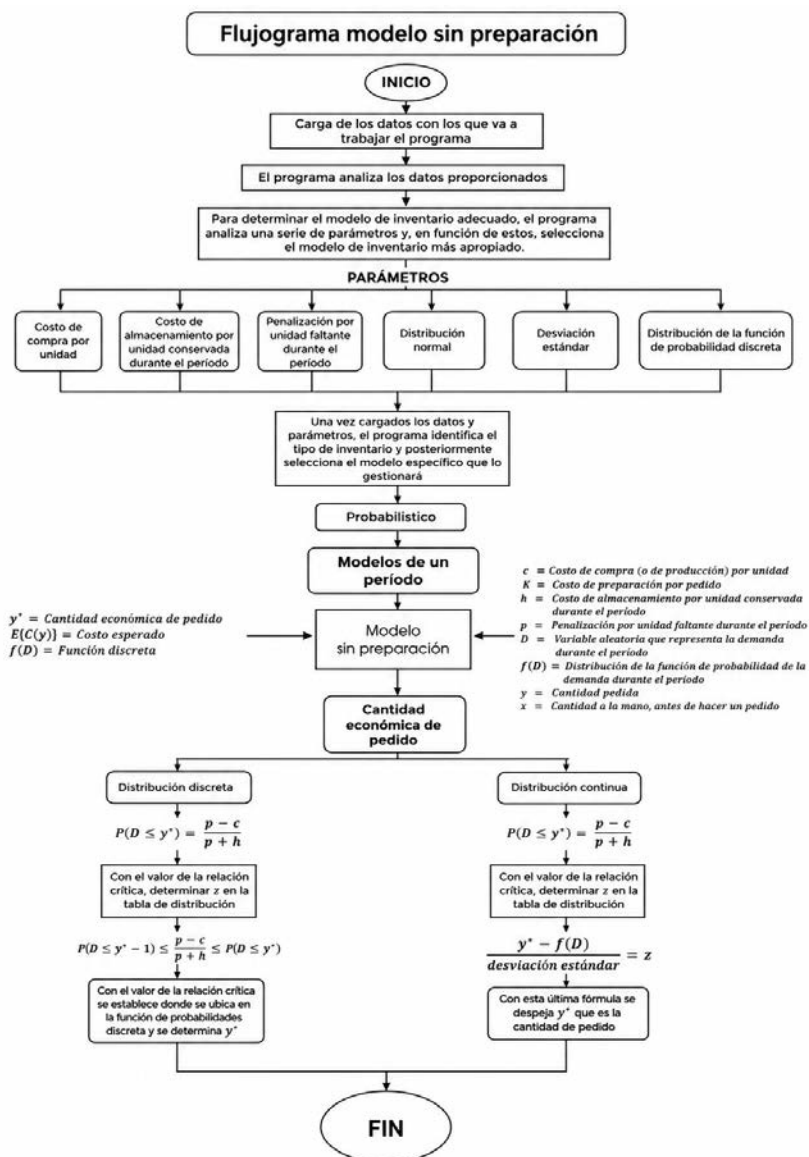
Para ejecutar correctamente este modelo, el sistema requiere el ingreso de los siguientes datos:

- **c:** Costo de compra o producción por unidad
- **h:** Costo de almacenamiento por unidad mantenida durante el período
- **p:** Costo o penalización por unidad no satisfecha (faltante)
- **D:** Variable aleatoria que representa la demanda durante el período
- **f(D):** Función de distribución de probabilidad de la demanda
- **y:** Cantidad solicitada en el pedido
- **x:** Cantidad disponible en inventario antes de realizar un nuevo pedido

Gracias a esta formulación, el modelo sin preparación constituye una herramienta robusta para optimizar decisiones en entornos con alta incertidumbre y estructuras de costos más dinámicas, donde la flexibilidad en los pedidos y el control sobre los faltantes son clave para la rentabilidad y el nivel de servicio.

Figura 11

Flujograma modelo sin preparación



Proceso de cálculo y evaluación

El programa identifica estos parámetros e inmediatamente se desvía hacia el modelo sin preparación. Una vez determinado que este es el modelo de inventario adecuado, el programa sigue una serie de pasos específicos para llevar a cabo los cálculos necesarios. Estos pasos aseguran que el proceso se realice de manera precisa y eficiente. A continuación, se describen los pasos que sigue el programa para este modelo en particular:

Distribución continua

Paso 1. – Como primer paso, se debe calcular la relación crítica con esta fórmula, este valor debe ser positivo.

$$\frac{p - c}{p + h} \quad (31)$$

Paso 2. – Una vez determinada la relación crítica, el programa toma este valor para calcularlo en la tabla de distribución el valor z .

Paso 3. – Con el valor de z definido por la tabla de distribución, se hace uso de la siguiente fórmula:

$$\frac{y^* - f(D)}{\text{desviación estándar}} = z \quad (32)$$

Paso 4. – Con esta fórmula se despeja y^* , y esa es la cantidad para pedir.

Distribución discreta

Paso 1. – Como primer paso, se debe calcular la relación crítica con esta fórmula, este valor debe ser positivo.

$$\frac{p - c}{p + h} \quad (31)$$

Paso 2. – Con el valor cálculo de la relación crítica el programa hace uso de la siguiente condición:

$$P\left\{D \leq y^* - 1\right\} \leq \frac{p - c}{p + h} \leq P\left\{D \leq y^*\right\} \quad (33)$$

Con esta condición el programa determina de entre la función de probabilidad discreta, donde se encuentra la relación crítica y ese es el valor para pedir.

Modelo con preparación (política S-s)

La política (S-s) constituye uno de los enfoques más ampliamente utilizados en la gestión de inventarios bajo incertidumbre, especialmente cuando existen costos significativos asociados a la colocación de pedidos. Este modelo combina la necesidad de mantener niveles adecuados de inventario frente a la variabilidad de la demanda, con la eficiencia de reducir la frecuencia de reabastecimiento, agrupando pedidos para minimizar los costos fijos de preparación.

La Figura 12 presenta el flujograma que representa la lógica operativa del sistema al aplicar esta política. En él se detalla la secuencia de decisiones que el programa ejecuta, desde la evaluación del nivel de inventario actual, hasta la generación de órdenes de compra que permitan restablecer el *stock* al nivel máximo permitido.

La política (S-s) se basa en dos umbrales clave:

Esta estructura permite al sistema monitorear continuamente el inventario y generar pedidos solo cuando es realmente necesario, optimizando tanto los costos por faltantes como los costos por pedidos excesivos. A su vez, el modelo admite la incorporación de variabilidad probabilística en la demanda, lo

que lo convierte en una herramienta robusta en entornos donde la demanda es incierta y la toma de decisiones debe estar basada en análisis de riesgo.

El flujograma permite visualizar claramente cómo el sistema:

- **s:** Punto de reposición. Si el inventario cae por debajo de este valor, se activa el reabastecimiento.
- **S:** Nivel máximo de inventario. Es la cantidad a la que se debe reponer cada vez que el inventario cae por debajo de **s**.
- Evalúa si el inventario disponible está por debajo del umbral **s**.
- Determina la cantidad a pedir para alcanzar el nivel **S**.
- Integra los costos de almacenamiento, penalización y preparación en el proceso de decisión.
- Considera la demanda como una variable aleatoria con una distribución de probabilidad $f(D)$, permitiendo cálculos más realistas del comportamiento del sistema en el tiempo.

Este modelo es especialmente útil en sistemas de revisión continua, donde se realiza seguimiento en tiempo real del inventario, como en cadenas de suministro automatizadas, almacenes inteligentes, o sectores con alta sensibilidad al desabastecimiento (como salud, alimentos perecederos o manufactura *just-in-time*).

Parámetros requeridos

Para que el modelo funcione correctamente, el usuario debe ingresar los siguientes parámetros:

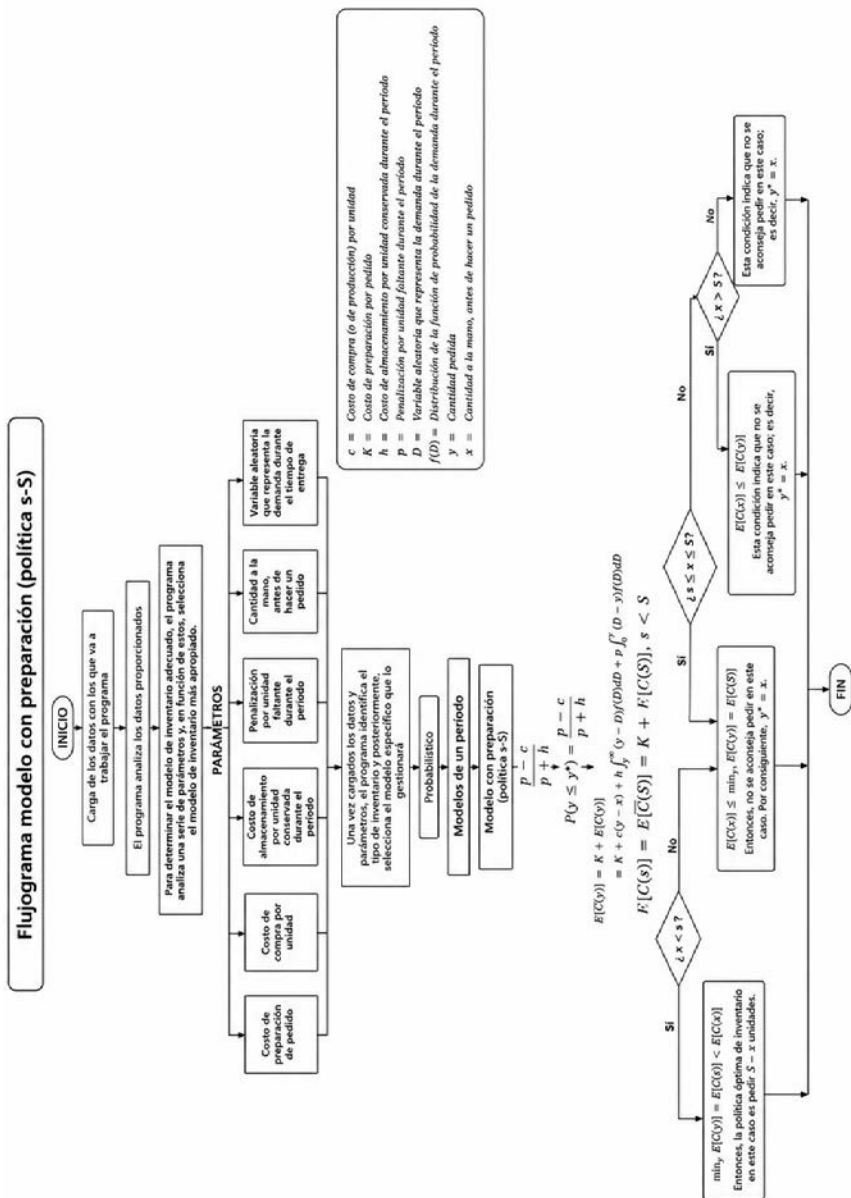
- **c:** Costo de compra o producción por unidad.

- **K:** Costo de preparación o emisión de un pedido.
- **h:** Costo de almacenamiento por unidad conservada durante el período.
- **p:** Penalización por unidad faltante durante el período.
- **D:** Variable aleatoria que representa la demanda durante el período.
- **f(D):** Distribución de la función de probabilidad asociada a la demanda.

Gracias a su diseño, el modelo con preparación bajo política (s,S) permite tomar decisiones estratégicas de inventario que equilibran eficientemente costos logísticos, niveles de servicio y riesgo de desabastecimiento, resultando especialmente adecuado para entornos con alta incertidumbre y costos fijos de aprovisionamiento.

Figura 12

Flujograma modelo con preparación (política s-s)



Proceso de cálculo y evaluación

El programa identifica estos parámetros e inmediatamente se desvía hacia el modelo con preparación (política s-S). Una vez determinado que este es el modelo de inventario adecuado, el programa sigue una serie de pasos específicos para llevar a cabo los cálculos necesarios. Estos pasos aseguran que el proceso se realice de manera precisa y eficiente. A continuación, se describen los pasos que sigue el programa para este modelo en particular:

Paso 1. – El primer paso que realiza el programa para este modelo de inventario es determinar la relación crítica con la fórmula:

Paso 2. – Una vez calculada la relación crítica, para calcular la cantidad de pedido y^* se necesita tener los límites de la función de distribución de probabilidades de la demanda uniforme, se hace de la fórmula:

$$P\left\{D \leq y^*\right\} = \int_0^{y^*} \frac{\text{limite inferior}}{\text{limite superior}} dD \quad (34)$$

Paso 3. – Cuando el programa calcula los valores de estas dos fórmulas, entonces puede determinar el valor de S, y una vez calculado S, el programa sigue con los procesos con la función de costos esperados por periodo.

$$E\left\{\bar{C}(y)\right\} = K + E\left\{\bar{C}(y)\right\} \quad (35)$$

$$E\left\{C(y)\right\} = K + c(y - x) + h \int_0^y (y - D)f(D)dD + p \int_y^\infty (D - y)f(D)dD \quad (36)$$

Paso 4. – Esta ecuación da como resultado una ecuación cuadrática con la variable y , entonces en este próximo paso el programa cambia de variable a s , con la ecuación.

$$E\{C(y)\} = K + E\{C(S)\}, s < S \quad (37)$$

Paso 5. – Una vez calculado esta ecuación, el programa ya determinó los valores de s , S , x . Y una vez calculados estos valores el programa actúa dependiendo de los valores.

A partir de este punto, el programa puede tomar una de 3 decisiones según una variante específica, basada en las preguntas:

- ¿ $x < s$?, de ser este el caso: La política de inventario debe ser pedir $S-x$ unidades
- ¿ $s \leq x \leq S$?, de ser este el caso: No es prudente hacer un pedido, entonces la política de inventario es $y^* = x$
- ¿ $x > S$?, de ser este el caso: No es prudente hacer un pedido, entonces la política de inventario es $y^* = x$

Modelo de varios periodos

La gestión de inventarios en entornos donde la demanda varía a lo largo del tiempo requiere modelos más complejos que aquellos basados en un único período o en condiciones estacionarias. En este contexto, el modelo de varios períodos se convierte en una herramienta fundamental para la toma de decisiones estratégicas,

ya que permite planificar de forma integral el abastecimiento a lo largo de un horizonte de planificación compuesto por múltiples períodos discretos (como semanas, meses o trimestres).

La Figura 13 presenta el flujograma que ilustra la lógica operativa del sistema al aplicar este modelo. El diagrama muestra de forma detallada cómo el sistema procesa la demanda estimada en cada período, integrando factores como costos de preparación, mantenimiento, penalización por faltantes, e incluso restricciones operativas, con el objetivo de establecer un plan óptimo de reabastecimiento que minimice el costo total acumulado a lo largo del horizonte temporal.

A diferencia de los modelos de un solo período o aquellos que suponen demanda constante, este enfoque permite evaluar decisiones coordinadas entre períodos. Es decir, el sistema no solo determina cuánto pedir en un período específico, sino que considera el impacto de dicha decisión en los períodos siguientes. Este análisis integral resulta particularmente valioso en entornos con costos fijos por pedido, restricciones de capacidad, o políticas de inventario que deben ser sincronizadas con otras actividades operativas.

Desde el punto de vista computacional, el modelo recurre frecuentemente a algoritmos de programación dinámica para identificar la combinación más eficiente de decisiones de compra. Estos algoritmos permiten explorar diferentes secuencias posibles de pedidos, evaluando el costo total asociado a cada una y seleccionando la política de abastecimiento que ofrezca el mejor balance entre costos y niveles de servicio.

Parámetros requeridos: Para ejecutar correctamente el modelo de varios períodos, el sistema requiere que el usuario proporcione los siguientes parámetros:

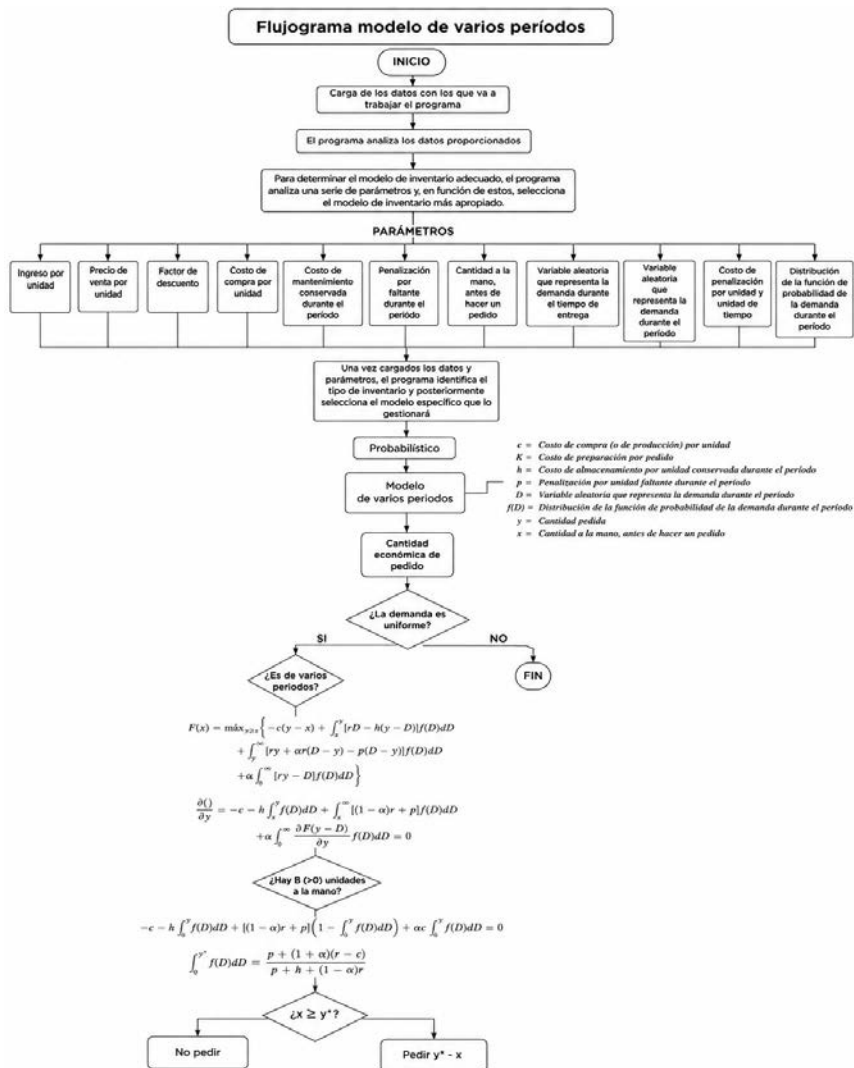
- **c:** Costo de compra o producción por unidad.

- **K:** Costo de preparación asociado a cada pedido.
- **h:** Costo de almacenamiento por unidad mantenida en inventario durante un período.
- **p:** Penalización por unidad faltante durante el período.
- **D:** Variable aleatoria que representa la demanda durante el período.
- **f(D):** Distribución de probabilidad correspondiente a la demanda en el período.
- **y:** Cantidad pedida en un determinado período.
- **x:** Nivel de inventario disponible antes de realizar un pedido.
- **r:** Ingreso o retorno económico por unidad satisfactoriamente vendida o utilizada.

Este modelo es especialmente útil en industrias que operan con ciclos estacionales, promociones programadas, o restricciones presupuestarias periódicas, donde las decisiones de inventario deben ajustarse continuamente a un entorno cambiante. Gracias a su enfoque multietapa, el sistema puede proponer políticas de reabastecimiento más precisas, flexibles y económicamente sostenibles, contribuyendo así a una planificación logística más robusta y alineada con los objetivos estratégicos de la organización.

Figura 13

Flujograma modelo de varios periodos



Proceso de cálculo y evaluación

El programa identifica estos parámetros e inmediatamente se desvía hacia el modelo de varios periodos. Una vez determinado que este es el modelo de inventario adecuado, el programa sigue una serie de pasos específicos para llevar a cabo los cálculos necesarios. Estos pasos aseguran que el proceso se realice de manera precisa y eficiente. A continuación, se describen los pasos que sigue el programa para este modelo en particular:

Paso 1. – Primero, el programa verifica si la demanda es uniforme. Si se determina que la demanda es uniforme, el programa continúa con los siguientes pasos. En caso contrario, detiene su proceso y finaliza la operación.

Paso 2. – Una vez que el programa verifica que la demanda es uniforme, procede a determinar si los datos abarcan varios periodos. Si se confirma que los datos comprenden múltiples periodos, el programa continúa con el proceso. En caso contrario, finaliza la operación y detiene todas las actividades adicionales.

Paso 3. – Una vez verificado todos estos pasos el programa empieza con el modelo de programación dinámica:

$$\begin{aligned}
 F_i(x_i) = \max_{y_i \geq x_i} \{ & -c(y_i - x_i) \\
 & + \int_0^{y_i} [rD - h(y_i - D)] f(D) dD \\
 & + \int_{y_i}^{\infty} [ry_i + \alpha r(D - y_i) - p(D - y_i)] f(D) dD + \\
 & \int_0^{\infty} F_{i+1}(y_i - D) f(D) dD \}, i = 1, 2, \dots, n
 \end{aligned} \tag{38}$$

Paso 4. _ Para determinar el valor de y , el programa hace uso de la siguiente condición:

$$\begin{aligned} \frac{\partial(.)}{\partial y} = & -c - h \int_0^y f(D) dD \\ & + \int_y^\infty [(1 - \alpha)r + p] f(D) dD \end{aligned} \quad (39)$$

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿Hay β (>0) unidades a la mano al iniciar el periodo?

Si no se cumple la condición, el programa finaliza su proceso de inmediato, deteniendo todas las operaciones. Sin embargo, si la condición se cumple, el programa procede al siguiente conjunto de procesos y cálculos necesarios para la gestión eficiente del inventario.

Paso 5. – Para determinar la política óptima de inventario de cada inventario el programa utiliza:

$$\int_0^{y^*} f(D) dD = \frac{P + (1 + \alpha)(r - c)}{p + h + (1 - \alpha)r} \quad (40)$$

A partir de este punto, el programa puede tomar una de dos decisiones según una variante específica, basada en la pregunta: ¿ $x \geq y^*$?

SI $x \geq y^*$:

El programa reconoce esta condición y determina que en este caso no se debe realizar el pedido

Por otro lado, si $x < y^*$:

El programa reconoce esta condición y determina que en este caso se debe pedir x .

Con la incorporación de este último modelo, se ha logrado cubrir de manera exhaustiva todos los tipos de inventarios y las funciones que el programa es capaz de realizar. Inicialmente, el enfoque estaba centrado únicamente en los modelos de inventarios más utilizados por las empresas de Tulcán. Sin embargo, durante el proceso de desarrollo del programa, se decidió ampliar su alcance para incluir una gama completa de modelos de inventarios.

Esta expansión no solo fortalece la robustez y versatilidad del programa, sino que también incrementa significativamente su potencial de aplicación en diversos contextos empresariales y situaciones específicas.

El enfoque integral adoptado garantiza que el programa pueda adaptarse a diferentes necesidades y circunstancias, aumentando así su valor como una herramienta eficiente para la gestión de inventarios. Al contemplar desde modelos básicos hasta aquellos más avanzados y especializados, el sistema ofrece soluciones óptimas, eficaces y ajustadas a la complejidad particular de cada caso. Esta inclusión amplia otorga al programa una profundidad y capacidad de crecimiento significativas, beneficiando a los usuarios con una herramienta integral que facilita la toma de decisiones informadas y estratégicas frente a los desafíos propios de la gestión de inventarios.

Simulación de modelos y generación de la base de datos para el aprendizaje

Códigos del programa

Una vez definida la estructura lógica del sistema, tanto de forma general como específica para cada tipo de modelo de inventario —mediante flujogramas detallados—, se procedió a la implementación práctica de dichos diseños. Para ello, se seleccionó el lenguaje de programación Python, debido a su versatilidad, amplia disponibilidad de bibliotecas especializadas y su capacidad para traducir estructuras algorítmicas complejas en código funcional de manera eficiente.

En una primera fase del desarrollo, se implementaron módulos individuales para cada modelo de inventario. Esta etapa involucró la codificación de funciones específicas y la aplicación de algoritmos que respondieran a las características propias de cada modelo, tanto en entornos determinísticos como probabilísticos. Posteriormente, estos códigos fueron integrados en dos programas principales: uno orientado a los modelos determinísticos y otro a los modelos probabilísticos.

Esta integración permitió una gestión organizada y eficiente de los distintos modelos, respetando la lógica operativa definida en los flujogramas. Los programas desarrollados ejecutan los cálculos necesarios, validan los parámetros de entrada y generan salidas coherentes con los objetivos de optimización en cada caso.

Es importante aclarar que estos programas constituyen una versión intermedia del sistema, no su forma final. Aunque actualmente ofrecen resultados precisos para cada modelo de inventario, el proyecto tiene un alcance más ambicioso: incorporar

técnicas de inteligencia artificial, particularmente mediante el uso de árboles de decisión, para que el sistema evolucione hacia un entorno de toma de decisiones autónoma y adaptativa.

En este sentido, los programas actuales serán empleados para simular múltiples escenarios operativos y generar una base de datos robusta y representativa. Esta base de datos se utilizará para el entrenamiento del sistema inteligente, permitiéndole aprender patrones de comportamiento, reconocer condiciones críticas y optimizar de forma automática las decisiones de gestión de inventarios. Gracias a este enfoque, se espera que el sistema evolucione desde una herramienta de cálculo hacia una plataforma de apoyo a la toma de decisiones inteligente, capaz de mejorar progresivamente su desempeño con base en la experiencia acumulada y en la información histórica generada.

Por razones de extensión y claridad, los códigos fuente de los modelos individuales se encuentran documentados en el Anexo 1, mientras que los programas integrados por tipo de modelo (determinístico y probabilístico) están disponibles en el Anexo 2 y el Anexo 3, respectivamente.

Simulación con los códigos individuales

Desarrollo de la base de datos

Con el objetivo de avanzar en la construcción del programa principal, se utilizaron los dos programas previamente desarrollados —correspondientes a los modelos determinísticos y probabilísticos— para llevar a cabo una serie de simulaciones. Estas simulaciones tuvieron como finalidad generar una base de datos representativa, necesaria para el posterior entrenamiento del sistema inteligente.

En total, se realizaron 100 simulaciones por cada uno de los modelos de inventarios, utilizando los códigos individuales

implementados previamente. Los resultados obtenidos fueron registrados de manera sistemática en una hoja de cálculo en formato Excel, dando origen a una base de datos sólida que sustentó el siguiente paso en la evolución del proyecto.

Transición hacia un enfoque inteligente

Una de las principales innovaciones introducidas en esta fase fue la adopción de un enfoque basado en inteligencia artificial. Este nuevo paradigma permitió diseñar un sistema que no requiere replicar los algoritmos completos de cada modelo dentro del código principal. En lugar de ello, el programa aprende directamente a partir de los datos simulados, utilizando únicamente los parámetros de entrada y los resultados de salida como insumos.

A través del entrenamiento con los datos generados en las simulaciones, el sistema de inteligencia artificial fue capaz de replicar automáticamente los comportamientos y resultados esperados, sin necesidad de ejecutar los procedimientos intermedios codificados. De este modo, el programa final puede operar de manera eficiente sin incorporar directamente los algoritmos de cada modelo en su estructura. No obstante, es importante señalar que los códigos individuales resultaron esenciales, ya que proporcionaron la información crítica para entrenar, validar y optimizar el desempeño del sistema inteligente.

Ejemplos representativos de simulación

Para ilustrar esta metodología, en la Figura 14 y la Figura 15 se presentan ejemplos de los registros generados a partir de las simulaciones. Estas figuras muestran la estructura empleada para almacenar datos, enfocándose exclusivamente en los parámetros de entrada y los resultados de salida de cada modelo.

Figura 14

Simulación de datos modelo EOQ y EOQ con discontinuidad de precios

SIMULACIÓN DE DATOS							
Modelo clásico de cantidad económica de pedido							
#	Parámetros	Tasa de demanda	Costo de preparación de pedido	Costo unitario/ u. tiempo de almacenamiento	Tiempo de entrega	CANTIDAD DE PEDIDO	PUNTO DE REORDEN
1	Parámetros	100	100	0,02	12	1000	200
2	Parámetros	300	20	0,21	1	239	60
3	Parámetros	120	30	0,04	5	424	175
4	Parámetros	145	23	0,05	6	365	139
5	Parámetros	80	15	0,03	10	282	234
6	Parámetros	3000	200	0,4	20	1732	1110
7	Parámetros	240	30	0,06	5	489	220
8	Parámetros	450	50	0,3	7	387	51
9	Parámetros	134	35	0,05	4	433	102
10	Parámetros	222	42	0,1	6	431	36

SIMULACIÓN DE DATOS									
Modelo clásico de cantidad económica de pedido con discontinuidades de precio									
#	Tasa de demanda	Costo de preparación de pedido	Costo compra normal	Costo compra con descuento	Costo unitario/ u. tiempo de almacenamiento	Tamaño mínimo del pedido con descuento	Tiempo de entrega	CANTIDAD DE PEDIDO	PUNTO DE REORDEN
1	396	505	11,44	9,25	0,03	875	22	3778	1154
2	311	230	17,6	9,63	0,43	13	11	931	535
3	832	57	13,06	6,86	0,47	207	17	446	305
4	59	552	19,49	4	0,14	513	20	688	491
5	519	230	14,54	1,83	0,42	118	8	753	384
6	866	488	15,25	1,28	0,43	378	29	1395	1393
7	883	331	19,14	7,5	0,31	741	10	1321	901
8	932	56	16,83	6,88	0,01	326	17	2719	1161
9	991	85	19,38	1,09	0,48	376	3	563	420
10	311	535	17	3,31	0,12	163	17	1642	359
11	213	313	11,39	2,35	0,35	719	22	788	745
12	104	517	19,59	8,4	0,31	828	15	828	387
13	168	236	14,08	3,95	0,15	222	29	722	534
14	69	313	16,13	8,95	0,33	609	16	935	481
15	404	96	10,87	6,56	0,49	860	20	1158	783
16	693	258	10,64	5,84	0,16	454	10	1086	693
17	277	180	19,07	3,78	0,41	532	22	562	604
18	171	404	15,58	6,19	0,16	826	18	1352	726
19	861	510	18,27	8,43	0,21	729	14	3032	1205
20	373	277	11,56	7,95	0,09	386	15	1742	790
21	558	402	11,73	4,65	0,47	198	27	1094	700
22	795	133	19,4	2,38	0,49	256	23	1957	1321
23	591	442	13,44	2,84	0,05	170	24	635	447
24	261	356	19,93	1,75	0,05	112	1	111	36
25	996	263	12,85	1,33	0,22	178	27	4075	2689
26	596	399	16,28	8,85	0,16	782	9	1535	728
28	211	141	10,89	3,06	0,43	552	16	611	508
28	121	322	14,18	7,9	0,02	153	29	3550	3509
29	652	84	17,57	6,28	0,15	750	21	2088	1368
30	721	366	17,09	2,69	0,01	144	14	445	287
31	341	336	11,89	3,56	0,42	143	13	961	443
32	811	513	15,92	2,04	0,17	149	19	494	299
33	118	139	11,29	2,65	0,01	507	89	89	16
34	961	101	19,68	8,89	0,05	908	15	1340	1479
35	609	73	19,81	6,49	0,24	434	20	1610	1217
36	374	271	14,77	6,58	0,11	575	27	891	717
37	254	248	16,49	7,56	0,37	132	29	1453	1471
38	747	482	10,41	7,99	0,42	976	25	2444	1863
39	486	60	18,61	2,05	0,3	516	8	1807	1209
40	991	493	15,07	8,13	0,43	806	18	1128	953
41	337	518	17,56	5,07	0,24	411	23	1599	1312
42	482	242	15,06	3,27	0,41	683	27	824	696
43	565	120	18,08	3,74	0,28	611	28	2117	1587
44	604	472	10,63	7,16	0,21	748	11	1143	664
45	127	157	14,01	1,59	0,37	233	28	301	355
46	331	144	16,12	2,33	0,46	488	12	1053	397

Figura 15

Simulación de datos modelos probabilizado y con política S-s

a) Modelo "probabilizado" de cantidad económica de pedido							b) Modelo con preparación (política s-S)									
Periodo	Parámetros	Demanda durante el tiempo de entrega		Máxima cantidad a pedir $y^* - x$ (unidades)	Cantidad de pedido (unidades)	Punto de reorden	Cantidad a la mano, antes de hacer un pedido (e)	Costos por unidad				Demanda máxima (Dmax)	S	s1	s2	Cantidad a pedir $y^* - x$ (unidades)
		Promedio de y (unidades)	Desviación de y (unidades)					Costo de preparación (K)	Costo de compra (c)	Costo de almacenamiento (h)	Penalización por unidad faltante (p)					
1	Parámetros	58	29	0,070	95,7	585,7	5	24,72	2,04	2,86	18,30	94	7,68	17,27	-1,91	No pedir
2	Parámetros	115	20	0,040	35,01	150,01	29	63,19	4,54	0,78	12,20	26	15,34	31,63	0,28	No pedir
3	Parámetros	123	21	0,020	121,99	1165,99	21	46,50	8,61	3,81	14,29	20	6,27	16,47	-3,85	No pedir
4	Parámetros	171	14	0,010	97,71	1636,71	6	30,32	3,95	3,68	6,46	23	6,05	18,58	-6,03	No pedir
5	Parámetros	150	22	0,080	87,43	1295,43	22	59,36	5,29	4,88	12,56	12	5,15	14,13	-3,77	No pedir
6	Parámetros	125	30	0,030	149,28	1024,28	30	78,71	4,27	1,76	15,90	73	48,07	74,07	22,84	18,07
7	Parámetros	180	15	0,090	60,33	1680,33	24	62,23	4,42	2,67	6,10	70	13,40	45,22	-18,13	No pedir
8	Parámetros	186	13	0,090	34,86	778,86	3	26,39	4,30	7,30	13,13	10	4,32	9,41	-0,76	No pedir
9	Parámetros	80	31	0,040	67,7	704,7	13	58,77	1,37	3,09	6,90	50	27,67	56,45	4,98	14,67

Nota: y^* es la cantidad económica de pedido y x es la cantidad a la mano en el momento de tomar la decisión.

S: punto de reorden superior (nivel de inventario objetivo) s1 y s2: límites de control
 y : cantidad económica de pedido $y^* - x$: cantidad a pedir

Es relevante destacar que en estas representaciones no se incluyen los cálculos intermedios. Esta decisión busca simplificar la presentación de los datos, facilitando una comparación directa entre modelos determinísticos y probabilísticos, y permitiendo una lectura más accesible sin entrar en detalles técnicos.

Registro completo de simulaciones

Para complementar esta sección, se ha incorporado en el Anexo 4 un registro exhaustivo de todas las simulaciones realizadas. Este anexo contiene los detalles completos de las ejecuciones, los resultados y los análisis asociados a cada código y modelo, proporcionando una visión integral del comportamiento del sistema en distintos escenarios.

3 **CAPÍTULO**

Implementación de árboles de
decisión y redes neuronales
en modelos de inventario

Motivación

Este capítulo presenta el desarrollo y la validación del programa principal de gestión de inventarios, cuyo eje innovador se basa en la incorporación de técnicas de inteligencia artificial, específicamente árboles de decisión y redes neuronales. La motivación central es fortalecer la capacidad del sistema para predecir, aprender y adaptarse a distintos escenarios empresariales, mejorando la toma de decisiones frente a la variabilidad de la demanda y los recursos.

En una primera fase, el sistema se diseñó con árboles de predicción, aplicados a modelos determinísticos como el EOQ, para estimar cantidades óptimas de pedido y puntos de reorden. Posteriormente, se añadieron árboles de clasificación para abordar variables cualitativas y decisiones binarias como "pedir" o "no pedir" ampliando así el alcance del programa. Estos modelos se implementaron en Python con bibliotecas como NumPy, Pandas y Scikit-learn, y se validaron mediante técnicas K-Fold para garantizar precisión y estabilidad.

Durante la validación se identificaron limitaciones en la capacidad predictiva de los árboles, especialmente en escenarios con alta incertidumbre. Por ello, se integraron redes neuronales

artificiales, logrando una reducción significativa del error y una mayor consistencia en los resultados. Este enfoque híbrido —que combina el rigor matemático con el aprendizaje automático— consolida al programa como una herramienta inteligente y adaptable para la optimización avanzada de inventarios.

Creación del programa principal

Los árboles de decisión se fundamentan en la teoría de decisión bajo incertidumbre de Von Neumann y Morgenstern (1947), que propone la evaluación de alternativas en función de resultados probables y su utilidad esperada. Breiman Friedman y Olshen (1984) desarrollaron el método CART (Classification and Regression Trees), que constituye la base matemática de los algoritmos de predicción y clasificación utilizados en la actualidad.

Una vez completadas las bases de datos necesarias para el funcionamiento del programa principal, se inició la implementación de técnicas de inteligencia artificial con énfasis en el uso de árboles de decisión. En una primera etapa, se optó por emplear únicamente árboles de predicción para abordar los diferentes modelos de inventario integrados en el sistema. Sin embargo, a medida que avanzó el desarrollo, se identificaron limitaciones en este enfoque, especialmente en aquellos modelos que requerían decisiones categóricas en lugar de valores numéricos continuos.

Ante esta situación, se evaluó la incorporación de árboles de clasificación como complemento a los árboles de predicción. Esta decisión se fundamentó en las diferencias funcionales entre los modelos empleados: mientras algunos exigen respuestas cuantitativas continuas (como la cantidad óptima a pedir), otros requieren decisiones cualitativas (como "pedir" o "no pedir"). En

consecuencia, se estableció un criterio técnico para seleccionar el tipo de árbol más adecuado según las características de cada modelo y el tipo de datos procesados.

Esta diversificación metodológica permitió que el sistema se adaptara con mayor eficacia a una amplia gama de situaciones en la gestión de inventarios. La integración de ambos tipos de árboles de decisión resultó en una mejor precisión en la predicción y clasificación de decisiones logísticas, lo que a su vez contribuyó a mejorar el desempeño general del programa. Esta flexibilidad constituye un elemento clave para asegurar una respuesta robusta y eficiente del sistema ante escenarios empresariales dinámicos y diversos.

Árbol de decisión por predicción

Cazares (2025) desarrolla un modelo predictivo utilizando árboles de decisión implementados en Python para gestionar de manera eficiente los niveles de inventario. Se tomó como referencia el modelo de la cantidad económica de pedido (EOQ), incorporando variables dinámicas como la demanda estacional y costos asociados al almacenamiento y pedidos. El sistema desarrollado permite automatizar la predicción de la cantidad óptima de reabastecimiento, reduciendo costos y mejorando la toma de decisiones logísticas. El proyecto utilizó bibliotecas como Scikit-learn, Pandas y NumPy para entrenar y aplicar el modelo, demostrando que el aprendizaje automático puede complementar los modelos tradicionales de inventario.

Para el desarrollo del programa basado en árboles de decisión orientado a la predicción, se utilizaron diversas bibliotecas de Python y estructuras de código que permiten aplicar modelos de aprendizaje automático en la gestión de inventarios. En esta sección, se describe de forma detallada el

proceso de implementación, tomando como referencia el modelo de la cantidad económica de pedido (EOQ), el cual sirve como base para ilustrar la funcionalidad del sistema.

El propósito principal del código desarrollado es predecir la cantidad óptima de pedido en un sistema de inventario que sigue la lógica del modelo EOQ. Este modelo, ampliamente utilizado en logística y operaciones, busca minimizar el costo total derivado del almacenamiento y del reabastecimiento. La aplicación de inteligencia artificial en este contexto permite automatizar el cálculo de dicha cantidad óptima y ajustarlo de manera dinámica a diferentes condiciones de entrada.

Importación de bibliotecas fundamentales

Como se muestra en la Figura 16, el proceso se inicia con la importación de bibliotecas fundamentales. En primer lugar, se utiliza NumPy, una herramienta esencial para la manipulación de arreglos multidimensionales y la ejecución de operaciones matemáticas avanzadas. En este contexto, NumPy permite transformar los datos de entrada en estructuras numéricas eficientes, optimizando el procesamiento requerido por los algoritmos de aprendizaje automático.

Adicionalmente, se importa Pandas, una biblioteca indispensable para la gestión y análisis de datos estructurados. Pandas proporciona estructuras como DataFrame, que facilitan la organización, limpieza y visualización de los datos históricos de inventario. Estas herramientas resultan fundamentales para preparar los conjuntos de datos que serán utilizados en la etapa de entrenamiento del árbol de decisión.

Este enfoque inicial, basado en la preparación eficiente de datos mediante NumPy y Pandas, sienta las bases para una implementación robusta del modelo EOQ dentro del sistema

inteligente de inventarios. Al asegurar una manipulación adecuada de los datos, se mejora la calidad de los insumos que alimentan el modelo predictivo, lo que a su vez incrementa la precisión de las recomendaciones generadas por el árbol de decisión.

Figura 16

Fragmento de código: importación de bibliotecas fundamentales

```
[ ] import numpy as np  
    import pandas as pd
```

Lectura de la base de datos

Una vez definidas las bibliotecas fundamentales para la manipulación y análisis de datos, el siguiente paso en la implementación del modelo EOQ consiste en la lectura y depuración de la base de datos. Para esta tarea, se emplea la biblioteca Pandas, que permite importar datos desde archivos en diversos formatos, facilitando así la integración de las simulaciones generadas previamente.

Tal como se observa en la Figura 17, el código utiliza la función `read_excel()` de Pandas para cargar un archivo de Excel que contiene la base de datos construida con los programas desarrollados en etapas anteriores. Este archivo está organizado en múltiples hojas, cada una correspondiente a un modelo distinto de inventario. En este caso específico, el programa accede a la hoja denominada "EO", que agrupa todas las simulaciones relacionadas con dicho modelo.

Figura 17

Fragmento de código: función read_excel() de Pandas para cargar un archivo de Excel

```
[ ] data = pd.read_excel("/content/drive/MyDrive/Modelos de inventarios.xlsx", sheet_name="EQ0")
data
```

Limpieza de datos y preparación final

Una vez importados los datos, se ejecuta la función dropna() sobre el DataFrame, como se muestra en la Figura 18. Esta instrucción elimina automáticamente todas las filas que contienen valores nulos o faltantes en cualquiera de sus columnas. El uso de data = data.dropna() asegura que el conjunto de datos esté libre de entradas incompletas antes de proceder al entrenamiento del modelo de aprendizaje automático.

Este procedimiento es fundamental para garantizar la consistencia del modelo predictivo, ya que los valores faltantes podrían distorsionar los patrones aprendidos o provocar errores en la ejecución del algoritmo. La depuración previa de los datos fortalece la confiabilidad de los resultados y mejora la precisión del árbol de decisión, al trabajar exclusivamente con información completa y válida.

Figura 18

Fragmento de código: función dropna() sobre el DataFrame

```
[ ] data = data.dropna()
```

Selección de variables y preparación del entrenamiento

Una vez cargada y depurada la base de datos correspondiente al modelo EOQ, se procede a la selección de las variables que formarán parte del proceso de entrenamiento del modelo de aprendizaje automático. Esta etapa es esencial para definir claramente los atributos que influirán en la predicción de la cantidad óptima de pedido.

Como se muestra en la Figura 19, se seleccionan las variables independientes o características del modelo, las cuales incluyen:

- Demanda (D)
- Costo de pedido (K)
- Costo de mantenimiento (h)
- Tiempo de entrega (L)

Estas variables se convierten en un arreglo utilizando la biblioteca NumPy, lo que permite estructurar los datos de entrada de manera eficiente y compatible con los algoritmos de aprendizaje automático.

En paralelo, se identifica la variable dependiente o variable objetivo, que en este caso es el lote óptimo de pedido. Esta columna representa el valor que el modelo intentará predecir a partir de las combinaciones de las variables de entrada.

Al definir explícitamente esta relación entre características y objetivo, se establece el marco para entrenar un árbol de predicción supervisado. Este proceso de separación entre variables predictoras y variable objetivo es un paso fundamental en la construcción de modelos de inteligencia artificial, ya que

permite al algoritmo aprender patrones y relaciones funcionales que luego pueden ser aplicados a nuevos datos.

De este modo, se garantiza que el sistema no solo reproduzca cálculos estáticos, sino que sea capaz de generalizar y ofrecer recomendaciones adaptativas según distintos contextos operativos.

Figura 19

Fragmento de código: proceso de entrenamiento del modelo de aprendizaje automático



	D	K	h	L	cantidad_pedido	reorden
0	100	100.0	0.02	12	1000	200
1	300	20.0	0.21	1	239	60
2	120	30.0	0.04	5	424	175
3	145	23.0	0.05	6	365	139
4	80	15.0	0.03	10	282	234
...	...	...	...	...	...	...
95	128	149.5	0.06	8	798	225
96	314	118.2	6.31	1	108	97
97	910	19.6	3.70	5	98	33
98	680	79.2	0.25	2	656	47
99	150	144.3	0.09	4	693	600

100 rows × 6 columns



```
features = np.asarray(data[["D", "K", "h", "L"]])
targets = np.asarray(data["cantidad_pedido"])
```

Construcción del árbol de decisión y validación del modelo

En la Figura 20, se ilustra el proceso de implementación del modelo, que comienza con la importación del módulo correspondiente a árboles de decisión desde la biblioteca Scikit-learn, una de las más utilizadas en el ámbito del aprendizaje automático.

A continuación, se crea una instancia del modelo especificando un parámetro de profundidad máxima de 50, con el objetivo de limitar la complejidad del árbol. Esta restricción responde a una práctica común en la ciencia de datos para evitar el fenómeno conocido como sobreajuste (*overfitting*), el cual ocurre cuando un modelo es excesivamente complejo y termina memorizando los datos de entrenamiento en lugar de aprender patrones generalizables.

En este contexto, el árbol de decisión opera dividiendo los datos en función de las características de entrada —demanda, costo de pedido, costo de mantenimiento y tiempo de entrega— con el fin de estimar el valor más adecuado para la cantidad óptima de pedido. Esta lógica permite construir reglas de decisión claras y transparentes basadas en los datos históricos.

Figura 20

Fragmento de código: módulo correspondiente a árboles de decisión desde la biblioteca Scikit-learn

```
[ ] from sklearn.tree import DecisionTreeRegressor
    from sklearn.model_selection import KFold
    k = 5
    kf = KFold(n_splits = k, shuffle = True)
    d_tree = DecisionTreeRegressor(max_depth= 50)
```

Evaluación del modelo: validación cruzada k-Fold

Para evaluar la calidad del modelo y garantizar su capacidad de generalización, se implementa la técnica de validación cruzada K-Fold, una estrategia ampliamente aceptada por su solidez estadística (Qiu, 2024).

En este caso, el conjunto de datos se divide en cinco subconjuntos (o *folds*). Durante cada iteración del proceso:

Uno de estos subconjuntos se utiliza como conjunto de prueba.

Los otros cuatro se emplean para entrenar el modelo.

Este ciclo se repite cinco veces, asegurando que cada subconjunto sea utilizado una vez como prueba y cuatro veces como entrenamiento. Además, se establece la opción de aleatorizar la distribución de los datos, lo que permite eliminar posibles sesgos derivados del orden original de los registros.

Durante este proceso, como se muestra en la Figura 21, se inicializa una estructura para almacenar todas las predicciones realizadas por el modelo en cada una de las iteraciones de validación. Para ello, se emplea la función `np.zeros_like(targets)` de la biblioteca NumPy, que genera un arreglo del mismo tamaño que el vector de valores reales (*targets*), pero con todos sus elementos inicializados en cero. Esta estructura se actualiza progresivamente con las predicciones generadas en cada *fold*, lo que permite posteriormente comparar el rendimiento del modelo de forma agregada.

Esta estrategia mejora la representatividad de la evaluación y refuerza la robustez del modelo frente a distintos escenarios. Este enfoque no solo valida el desempeño del árbol de decisión, sino que también proporciona una medida objetiva de su capacidad

para adaptarse a nuevas situaciones. Así, se construye un modelo predictivo que combina precisión, eficiencia y generalización, pilares fundamentales en la aplicación de inteligencia artificial en la gestión de inventarios.

Figura 21

Fragmento de código: inicialización del arreglo para almacenar predicciones con `np.zeros_like()`.

```
all_predictions = np.zeros_like(targets)
```

Ejecución del ciclo de validación cruzada

En la Figura 22, se observa la implementación de un bucle iterativo que se ejecuta cinco veces, correspondiente al número de divisiones establecido para la validación cruzada. En cada iteración del ciclo, los datos se dividen en dos partes: por un lado, el conjunto de entrenamiento, que incluye tanto las características como los valores objetivo; y por otro, el conjunto de prueba, que contiene los datos que el modelo no ha visto previamente, y que se utilizarán para evaluar su capacidad de predicción.

Una vez separados los datos, el modelo de árbol de decisión es entrenado utilizando exclusivamente el conjunto de entrenamiento. Luego, se emplea este modelo entrenado para realizar predicciones sobre el conjunto de prueba, tomando como insumo únicamente sus características. Las predicciones obtenidas se almacenan en el arreglo `all_predictions`, que fue creado previamente, asignándolas específicamente en las posiciones correspondientes a los índices del conjunto de prueba.

Este procedimiento garantiza que las predicciones queden organizadas en su posición original dentro del conjunto de

datos completo, lo que facilita su posterior comparación con los valores reales. Este enfoque permite preservar la integridad de la evaluación, ya que cada observación es utilizada una única vez como prueba y cuatro veces como entrenamiento, asegurando una visión completa y balanceada del desempeño del modelo. Al final del ciclo, el arreglo `all_predictions` contiene una serie ordenada de predicciones realistas, generadas en condiciones controladas y metodológicamente sólidas.

Figura 22

Fragmento de código: implementación de un bucle iterativo que se ejecuta cinco veces

```
❶ # declarar un array para los valores predichos en cada iteración
all_predictions = np.zeros_like(targets)

for train_indices, test_indices in kf.split(features):
    train_features, test_features = features[train_indices], features[test_indices]
    train_targets, test_targets = targets[train_indices], targets[test_indices]

    ## Entrenar el decision tree
    d_tree.fit(train_features, train_targets)

    ## Predecir usando el modelo
    predictions = d_tree.predict(test_features)

    ## Ponemos las predicciones en el array de all_predictions creado above
    all_predictions[test_indices] = predictions
```

Evaluación numérica del desempeño del modelo

En la Figura 23, el código evalúa el rendimiento del modelo. Para ello, se calcula la mediana de las diferencias absolutas entre las predicciones generadas y los valores reales de la variable objetivo. Esta medida proporciona una indicación clara de qué tan cerca están las predicciones del modelo respecto a los valores verdaderos del conjunto de datos. Al centrarse en la mediana en lugar de en el promedio, se reduce el impacto de valores atípicos o extremos, ofreciendo una evaluación más robusta y representativa del error típico del modelo.

Figura 23

Fragmento de código Python: evaluación de rendimiento del modelo

```
## Evaluamos el modelo usando la diferencia de medianas de todas las predicciones realizadas
eval_model = np.median(np.abs(all_predictions - targets))
eval_model
```

Visualización gráfica del desempeño del modelo

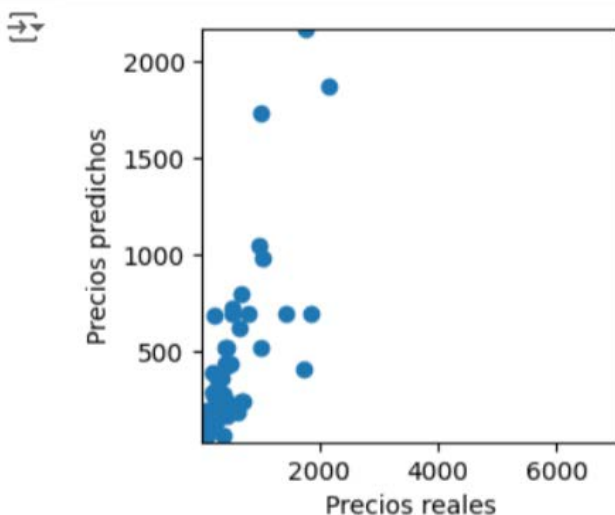
En la Figura 24, se emplea la biblioteca Matplotlib para generar un gráfico de dispersión que compara los valores reales (objetivos o *targets*) con las predicciones generadas por el modelo (*all_predictions*). Cada punto del gráfico representa una instancia en la que el modelo realizó una predicción, ubicándose en el plano cartesiano según su valor real en el eje X y su valor predicho en el eje Y. Esta representación visual permite observar de forma inmediata qué tan precisas han sido las estimaciones del modelo. Idealmente, si las predicciones fueran perfectas, todos los puntos estarían alineados sobre la línea diagonal de identidad (donde el valor real es igual al valor predicho). La dispersión de los puntos alrededor de esa diagonal proporciona indicios sobre el grado de error del modelo: una menor dispersión indica mayor precisión, mientras que una mayor dispersión sugiere errores de predicción más significativos.

Este tipo de visualización complementa adecuadamente las métricas cuantitativas y ofrece una perspectiva clara sobre el comportamiento general del modelo, permitiendo identificar posibles patrones de sesgo, desviaciones sistemáticas o también llamadas outliers.. Así, la combinación de técnicas analíticas y gráficas fortalece la evaluación global del sistema de predicción basado en árboles de decisión.

Figura 24

Fragmento de código: gráfico de dispersión que compara los valores reales (objetivos o targets) con las predicciones generadas por el modelo (all_predictions)

```
from matplotlib import pyplot as plt
plt.figure(figsize = (3,3))
plt.scatter(targets, all_predictions)
plt.xlim(30,targets.max())
plt.ylim(30,predictions.max())
plt.xlabel("valores reales")
plt.ylabel("valores predichos")
plt.show()
```



Predicción con nuevas características: aplicación final del modelo

Finalmente, como se muestra en la Figura 25, el modelo de árbol de decisión entrenado se utiliza para realizar una predicción sobre un nuevo conjunto de características específicas. En este ejemplo, se proporcionan al modelo los siguientes parámetros:

- **Demanda:** 900 unidades
- **Costo de pedido:** 30
- **Costo de mantenimiento:** 0.01
- **Tiempo de entrega:** 3

Con base en estos valores, el modelo devuelve la cantidad óptima de pedido predicha para dichas condiciones particulares. Este ejemplo marca la conclusión del desarrollo del programa para el cálculo del modelo determinístico EOQ (economic order quantity).

Figura 25

Fragmento de código: modelo de árbol de decisión entrenado

```
[ ] cop = d_tree.predict([[900, 30, 0.01, 3]])
```

```
[ ] cop
```

```
↪ array([534.])
```

Adaptabilidad del programa a otros modelos de gestión de inventarios

Aunque en este caso el código se enfocó en la determinación de la cantidad económica de pedido, su estructura ha sido diseñada con suficiente flexibilidad para adaptarse al cálculo de otras variables clave, como el punto de reorden. Para lograr esto, basta con modificar la columna objetivo (*targets*) en la base de datos, ajustándola para reflejar la nueva variable de interés. Una vez realizado este cambio, el programa podrá realizar las predicciones correspondientes de forma automática.

Además, la solución desarrollada permite adaptarse fácilmente a una amplia variedad de modelos en la gestión de inventarios. La única modificación necesaria para trabajar con diferentes enfoques es cambiar la hoja del archivo Excel desde la cual se extraen los datos de entrada y salida. Al actualizar esta referencia, el programa procesará las nuevas variables características y objetivo, correspondientes al modelo seleccionado.

Modelos determinísticos compatibles:

- Modelo clásico de cantidad económica de pedido
- Cantidad económica de pedido con discontinuidades de precio
- Cantidad económica de pedido de varios artículos con limitación de almacén
- Modelo dinámico sin costo de preparación
- Modelo dinámico con preparación

Modelos probabilísticos compatibles:

- Modelo probabilizado de cantidad económica de pedido

- Modelo probabilista de cantidad económica de pedido
- Modelo de un solo periodo sin preparación

Solución para modelos con múltiples artículos y períodos

Una vez definido el funcionamiento general del programa y su compatibilidad con distintos modelos, se aborda un desafío más complejo: el manejo de modelos que contemplan múltiples artículos y períodos. En estos casos, se requiere calcular los resultados de forma individual para cada artículo o período, lo que impide realizar todas las predicciones en una única ejecución del modelo.

Para resolver este problema, se desarrolló una estrategia basada en el análisis secuencial de la lógica de resolución de cada modelo. Específicamente, se observó que en modelos como el dinámico sin costo de preparación, los valores de los períodos adicionales dependen directamente de los cálculos realizados en los períodos anteriores. Por ejemplo, para determinar la cantidad óptima de pedido en el período adicional 1, es necesario haber calculado previamente la cantidad correspondiente al período normal 1, y así sucesivamente.

Comprendiendo esta dependencia secuencial, la solución adoptada consistió en ejecutar el programa de manera iterativa, incrementando progresivamente el número de entradas conforme avanzaban los períodos o la cantidad de artículos involucrados. Esto asegura que cada cálculo se realice en el orden correcto, respetando la dependencia temporal o por artículo característica de estos modelos.

Manejo iterativo para modelos con restricción de área

En la Figura 26 se presenta el fragmento de código correspondiente al modelo con restricción de área, específicamente la sección encargada de gestionar la selección de entradas (*features*) y salidas (*targets*). En esta implementación, el programa recopila todos los datos necesarios para calcular los pedidos, estableciendo como columna objetivo el valor de pedido 1. De este modo, el modelo predice la cantidad a solicitar para el primer artículo.

Figura 26

Fragmento de código: selección de entradas y salidas para el cálculo del pedido del primer artículo

```

articulo.1 articulo.2 articulo.3 D1 D2 D3 K1 K2 K3 h1 h2 h3 a1 a2 a3 A pedido_1 pedido_2 pedido_3 area_total
0 1 2 3 2 4 4 10 5 15 0.3 0.1 0.2 1 1 1 25 6.34 7.09 11.57 24.999
1 1 2 3 3 2 9 11 16 9 0.4 0.7 0.7 5 4 3 65 4.45 4.59 8.12 64.996
2 1 2 3 3 3 4 16 7 18 0.7 0.7 0.2 3 4 1 46 6.10 3.62 13.20 45.993
3 1 2 3 9 10 3 8 19 7 0.4 0.6 0.4 5 5 5 72 4.58 7.34 2.47 71.962
4 1 2 3 8 4 7 17 13 16 0.3 0.2 0.3 4 2 4 74 7.90 6.83 7.17 73.957
...
95 1 2 3 4 9 4 15 18 16 0.3 0.5 0.5 2 4 2 50 5.75 6.73 5.78 49.983
96 1 2 3 6 4 7 7 14 15 0.7 0.7 0.3 2 2 3 25 3.01 3.48 4.00 24.995
97 1 2 3 9 8 4 5 18 18 0.1 0.5 0.4 4 2 5 43 2.98 7.22 3.33 42.987
98 1 2 3 8 2 9 17 12 8 0.8 0.7 0.1 4 2 4 26 3.20 1.88 2.36 26.000
99 1 2 3 5 10 4 16 6 8 0.6 0.6 0.8 1 5 3 40 9.03 4.02 3.62 39.984
100 rows x 20 columns

[ ] data = data.dropna()

[ ] features = np.asarray(data[["D1", "D2", "D3", "K1", "K2", "K3", "h1", "h2", "h3", "a1", "a2", "a3", "A"]])
targets = np.asarray(data["pedido_1"])
```

Ejecuciones secuenciales para múltiples artículos

Tal como se explicó anteriormente, para calcular los valores correspondientes a los artículos restantes, es necesario ejecutar el código de manera reiterada, ajustando dinámicamente las columnas de entrada y salida de acuerdo con el artículo a calcular.

En la Figura 27, se ilustra la segunda ejecución del programa. En este caso, pedido_1 se incorpora como parte de las entradas, mientras que la nueva columna objetivo se ajusta a pedido_2, con el fin de calcular el pedido correspondiente al segundo artículo.

Figura 27

Fragmento de código: segunda ejecución para calcular el pedido del segundo artículo

```
features = np.asarray(data[["D1", "D2", "D3", "K1", "K2", "K3", "h1", "h2", "h3", "a1", "a2", "a3", "A", "pedido_1"]])
targets = np.asarray(data["pedido_2"])
```

Siguiendo la misma lógica, en la Figura 28 se muestra la tercera ejecución, enfocada en el cálculo del pedido para el tercer artículo. En este paso, pedido_1 y pedido_2 forman parte de las entradas, y la nueva salida se define como pedido_3.

Figura 28

Fragmento de código: tercera ejecución para el cálculo del pedido del tercer artículo

```
[ ] features = np.asarray(data[["D1", "D2", "D3", "K1", "K2", "K3", "h1", "h2", "h3", "a1", "a2", "a3", "A", "pedido_1", "pedido_2"]])
targets = np.asarray(data["pedido_3"])
```

Este procedimiento iterativo permite garantizar cálculos precisos, respetando la dependencia entre los pedidos de cada artículo. En caso de requerir cálculos para un mayor número de artículos o periodos, el proceso es igualmente simple: basta con

agregar los datos correspondientes en la base de datos y volver a cargarla en el programa.

Flexibilidad y escalabilidad del modelo

Este enfoque permite que el sistema opere con la cantidad de artículos o periodos necesarios, manteniendo la flexibilidad y escalabilidad del modelo. Este tipo de manejo especial resulta indispensable para aquellos modelos que exigen cálculos específicos por artículo o periodo, como los siguientes:

- Cantidad económica de pedido de varios artículos con limitación de almacén
- Modelo dinámico sin costo de preparación
- Modelo dinámico con preparación

Conclusión general del funcionamiento del programa

Con esta sección concluye la explicación integral sobre el funcionamiento del programa basado en árboles de decisión para predicción, proporcionando además los conocimientos clave necesarios para su correcta comprensión y manipulación.

Se ha descrito de forma sistemática el proceso de modificación del código para la gestión de múltiples periodos o artículos, detallando paso a paso cómo adaptar el programa a escenarios más complejos. Esta metodología garantiza una aplicación versátil, escalable y sólida, ajustable a las necesidades específicas de cada modelo.

También se ha destacado la importancia de la base de datos como elemento central del sistema. Su adecuada estructuración y actualización permite manejar volúmenes mayores de información

o periodos adicionales sin necesidad de reescribir el código, facilitando la expansión y manteniendo la eficiencia operativa.

Árbol de decisión por clasificación

Parada Gutiérrez (2009) presenta un modelo de árbol de clasificación para apoyar decisiones relacionadas con la gestión de inventarios en situaciones donde las respuestas no son cantidades, sino categorías como "comprar", "esperar", o "revisar proveedor". Utilizan algoritmos de clasificación (CART y ID3) para modelar la toma de decisiones en inventarios bajo incertidumbre, facilitando la automatización de procesos logísticos cualitativos. La implementación se realizó en Python usando scikit-learn, permitiendo integrar modelos no paramétricos en el ámbito de la logística operativa.

En esta sección se describe el funcionamiento del programa basado en árboles de clasificación, desarrollado específicamente para abordar ciertos modelos de inventario con respuestas de tipo cualitativo. A diferencia de los árboles de regresión, utilizados previamente para predecir valores numéricos continuos, los árboles de clasificación son ideales cuando la variable objetivo representa categorías o decisiones discretas.

Justificación del uso de árboles de clasificación

La elección de este tipo de árbol responde a la naturaleza cualitativa de los resultados que se generan en determinados modelos. Mientras los árboles de regresión permiten comprender la relación entre variables independientes y una respuesta

cuantitativa, los árboles de clasificación se centran en asignar una clase o categoría a cada observación, con base en las características disponibles.

Este enfoque resulta esencial en situaciones donde las decisiones no se expresan en términos numéricos, sino como acciones discretas, como, por ejemplo: "realizar pedido" o "no realizar pedido". Por lo tanto, se diseñó un programa específico para manejar este tipo de análisis, garantizando una interpretación más apropiada de los resultados en contextos cualitativos.

Aplicación a modelos probabilísticos

El programa de árboles de clasificación está orientado principalmente a modelos probabilísticos, en particular a los siguientes:

- Modelo con preparación (política s-S)
- Modelos de varios periodos

En estos casos, el modelo no busca predecir una cantidad exacta de pedido, sino determinar si se debe realizar un pedido o no, o qué política aplicar según la situación. Por ello, la naturaleza de la predicción es discreta, y se presta perfectamente al uso de algoritmos de clasificación.

Inicio del programa: configuración del entorno

El funcionamiento del programa comienza con la configuración del entorno en Google Colab, una plataforma que permite el desarrollo y ejecución de código Python en la nube. Como

se muestra en la Figura 29, el primer paso consiste en montar el sistema de archivos de Google Drive, para permitir el acceso a los datos almacenados.

Este proceso se logra mediante la importación de la biblioteca drive del módulo google.colab, seguida del comando `drive.mount('/content/drive')`. Esta acción habilita la lectura y manipulación de archivos almacenados en Google Drive directamente desde el entorno de trabajo, lo que facilita el uso de bases de datos previamente creadas para los modelos de inventario.

Figura 29

Fragmento de código: ontaje del Google Drive



```
from google.colab import drive
drive.mount('/content/drive')
```

Drive already mounted at /content/drive; to attempt to forcibly remount, call `drive.mount("/content/drive", force_remount=True)`.

Importación de bibliotecas esenciales

En la Figura 30, se presenta el fragmento de código correspondiente a la importación de las bibliotecas fundamentales para el desarrollo del programa:

- **NumPy (np)**: se utiliza para realizar operaciones matemáticas eficientes y manipulación de arreglos.
- **Pandas (pd)**: permite leer, transformar y analizar datos en formato tabular mediante estructuras tipo DataFrame.
- **Matplotlib.pyplot (plt)**: facilita la visualización gráfica de resultados, ideal para interpretar el comportamiento del modelo.

- El comando `%matplotlib inline` asegura que las gráficas generadas se muestren directamente dentro del *notebook*, optimizando el flujo de análisis.

Estas herramientas proporcionan la base computacional y visual necesaria para construir, entrenar, evaluar y representar los modelos de clasificación utilizados.

Figura 30

Fragmento de código: importación de bibliotecas esenciales

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

Lectura y análisis preliminar de los datos

Una vez configurado el entorno de trabajo e importadas las bibliotecas necesarias, el siguiente paso consiste en la lectura del archivo de datos que contiene la información estructurada del modelo de inventario que se desea analizar. Tal como se muestra en la Figura 31, esta tarea se realiza mediante la función `pd.read_excel()` de la biblioteca `pandas`, la cual permite importar datos directamente desde archivos de Excel.

En la instrucción correspondiente, se especifica tanto la ruta al archivo almacenado en Google Drive como el nombre de la hoja a utilizar, en este caso "M_varios_periodos". El resultado de esta operación es un objeto tipo `DataFrame`, que actúa como una tabla estructurada, facilitando las tareas posteriores de transformación, visualización y entrenamiento del modelo.

Figura 31

Fragmento de código: lectura del archivo Excel y carga de la hoja "M_varios_periodos"

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

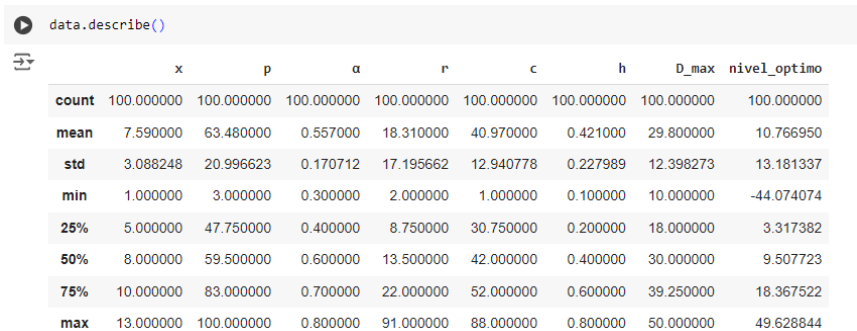
Exploración inicial del conjunto de datos

Una vez cargado el conjunto de datos, se realiza una exploración inicial para comprender la estructura y características estadísticas de las variables incluidas. Tal como se muestra en la Figura 32, se emplea la función `data.describe()`, la cual proporciona un resumen estadístico de las columnas numéricas del DataFrame.

Este resumen incluye métricas clave como el promedio (mean), la desviación estándar (std), los valores mínimo y máximo (min y max), así como los percentiles. Este paso es crucial para detectar posibles anomalías, evaluar la variabilidad de los datos y tener una visión general de su distribución, lo que permite preparar adecuadamente los datos antes de la etapa de modelado.

Figura 32

Fragmento de código: exploración inicial del conjunto de datos utilizando data.describe()



```
data.describe()
```

	x	p	a	r	c	h	D_max	nivel_optimo
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	7.590000	63.480000	0.557000	18.310000	40.970000	0.421000	29.800000	10.766950
std	3.088248	20.996623	0.170712	17.195662	12.940778	0.227989	12.398273	13.181337
min	1.000000	3.000000	0.300000	2.000000	1.000000	0.100000	10.000000	-44.074074
25%	5.000000	47.750000	0.400000	8.750000	30.750000	0.200000	18.000000	3.317382
50%	8.000000	59.500000	0.600000	13.500000	42.000000	0.400000	30.000000	9.507723
75%	10.000000	83.000000	0.700000	22.000000	52.000000	0.600000	39.250000	18.367522
max	13.000000	100.000000	0.800000	91.000000	88.000000	0.800000	50.000000	49.628844

Análisis de la variable objetivo

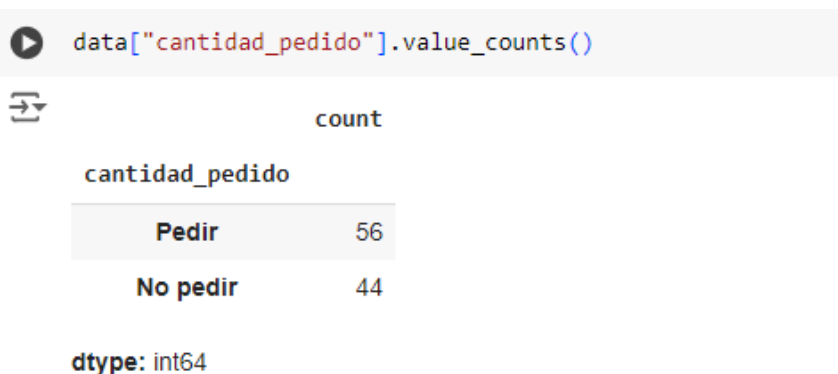
En la Figura 33, se presenta un análisis específico enfocado en la variable objetivo del modelo de clasificación: la columna "cantidad_pedido". Para ello, se emplea la función `data["cantidad_pedido"].value_counts()`, que contabiliza cuántas veces aparece cada una de las clases en dicha columna.

En este caso, los valores posibles son "pedir" y "no pedir", lo que indica un problema de clasificación binaria. Este análisis es fundamental para detectar posibles desequilibrios entre clases, lo cual puede afectar negativamente el desempeño del modelo si una clase domina significativamente sobre la otra.

Comprender la distribución de clases permite, en etapas posteriores, implementar estrategias de corrección como remuestreo, ponderación de clases, o ajustes en los criterios de evaluación del modelo, todo con el objetivo de construir un clasificador equilibrado y eficaz.

Figura 33

Fragmento de código: análisis de frecuencias de la variable objetivo "cantidad_pedido"



Transformación de variables categóricas y análisis de correlación

Una vez verificada la distribución de clases en la variable objetivo, es necesario transformar sus valores categóricos en un formato numérico que pueda ser interpretado por el modelo de clasificación. Este procedimiento se detalla en la Figura 34, donde se lleva a cabo el proceso completo de codificación de etiquetas.

En primer lugar, se identifican las categorías presentes en la columna "cantidad_pedido" mediante la función `unique()`, la cual devuelve los valores distintos contenidos en esa columna. En este caso, se obtienen las clases ["Pedir", "No pedir"], que representan las posibles decisiones de política de inventario.

Posteriormente, se crea un diccionario denominado "Nopedir" utilizando la función `enumerate()`, la cual asigna un índice entero a cada una de las categorías. Este paso produce

un mapeo como { "Pedir": 0, "No pedir": 1 }, que luego es aplicado a la columna original mediante el método map(). Como resultado, la columna "cantidad_pedido" queda transformada en una representación numérica, donde 0 indica la decisión de pedir y 1 indica la decisión de no pedir. La última línea del bloque de código muestra la tabla resultante con los valores ya convertidos.

Figura 34

Fragmento de código: conversión de variables categóricas a numéricas utilizando diccionario y map()

```
[ ] data["cantidad_pedido"].unique()
```

```
→ array(['Pedir', 'No pedir'], dtype=object)
```

```
[ ] Nopedir = {value: key for key,value in enumerate(data["cantidad_pedido"].unique())}
```

```
[ ] data["cantidad_pedido"] = data["cantidad_pedido"].map(Nopedir)
```

```
▶ gender_label = {value: key for key,value in enumerate(data["cantidad_pedido"].unique())}
data["cantidad_pedido"] = data["cantidad_pedido"].map(gender_label)
data
```

```
→
```

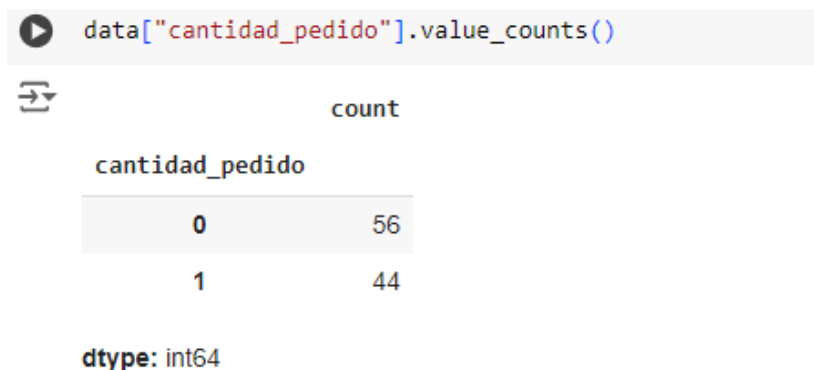
	x	p	α	r	c	h	D_max	nivel_optimo	cantidad_pedido
0	10	3	0.8	2	1	0.1	10	13.714286	0
1	1	67	0.6	6	44	0.3	39	3.469154	0
2	8	85	0.7	16	29	0.5	18	12.538206	0
3	7	87	0.8	20	24	0.5	45	39.245902	0
4	4	54	0.3	22	49	0.2	34	9.232759	0
...	...	...	...	...	...	...	...	...	...
95	10	92	0.8	23	51	0.2	18	7.735537	1
96	6	66	0.6	4	55	0.7	27	-6.166911	1
97	4	81	0.4	5	44	0.2	41	12.855107	0
98	9	69	0.8	24	35	0.7	27	17.830872	0
99	11	58	0.4	9	54	0.5	33	-2.582160	1

100 rows x 9 columns

Una vez completada la transformación, se verifica nuevamente la distribución de clases, esta vez sobre los valores numéricos de la variable objetivo. Como se presenta en la Figura 35, se utiliza nuevamente la función `value_counts()` para contabilizar la frecuencia de aparición de los valores 0 y 1. Este análisis confirma el balance de clases en el conjunto de datos transformado y permite evaluar si es necesario aplicar técnicas adicionales para compensar desequilibrios, como sobre-muestreo o ponderación de clases.

Figura 35

Fragmento de código: conteo de frecuencias sobre la columna transformada "cantidad_pedido"



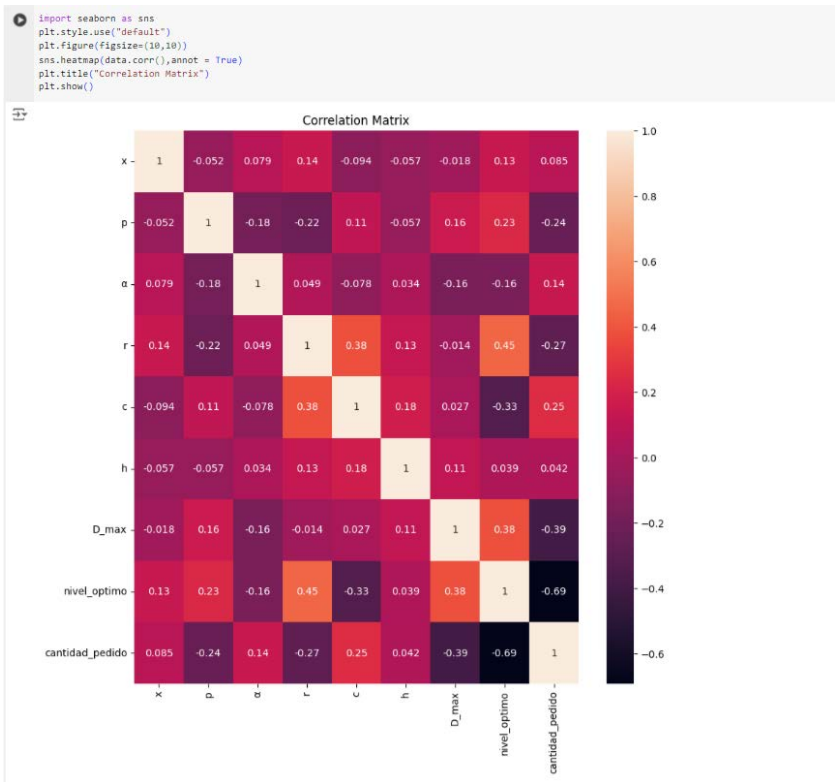
Con el objetivo de identificar relaciones significativas entre las variables predictoras, en la Figura 36 se muestra la matriz de correlación calculada sobre el DataFrame, y visualizada mediante un mapa de calor utilizando la biblioteca `seaborn`. Este tipo de representación gráfica facilita la detección de correlaciones altas entre variables, que pueden indicar redundancia de información.

La matriz de correlación es una herramienta crucial durante la etapa de selección de características, ya que permite

optimizar el rendimiento del modelo al eliminar variables altamente correlacionadas o poco relevantes, y enfocarse en aquellas que tienen una mayor capacidad predictiva respecto a la variable objetivo.

Figura 36

Mapa de calor de la matriz de correlación entre variables predictoras



Definición de entradas y objetivos del programa

En la Figura 37 se inicia la configuración de los datos que se utilizarán en el modelo de clasificación. Para ello, se emplea la función `np.asarray()` de la biblioteca NumPy, cuya función es convertir estructuras tipo pandas en arreglos NumPy. Esta conversión proporciona un manejo más eficiente y rápido, especialmente en tareas computacionalmente intensivas como el entrenamiento de modelos. En este fragmento de código, las características (*features*) se componen de variables como `x`, `p`, `$\alpha$` , `r`, `c`, `h`, `D_max` y `nivel_optimo`.

La Figura 38 muestra la transposición de la matriz de características mediante `features.T`, de modo que cada fila represente una observación única del conjunto de datos. La variable dependiente (*targets*) corresponde a la columna "cantidad_pedido", la cual ha sido previamente codificada en formato numérico. Esta estructura, que separa claramente las variables independientes y la variable objetivo, es esencial para el correcto funcionamiento del modelo de aprendizaje automático.

Figura 37

Fragmento de código: conversión a arreglos NumPy

```
 features = np.asarray([data["x"],  
                        data["p"],  
                        data[" $\alpha$ "],  
                        data["r"],  
                        data["c"],  
                        data["h"],  
                        data["D_max"],  
                        data["nivel_optimo"]])  
  
features = features.T  
features
```

Figura 38

Fragmento de código: organización de datos para entrenamiento

```
 targets = np.asarray(data["cantidad_pedido"])  
  
features.shape, targets.shape  
  
 ((100, 8), (100,))
```

División del conjunto de datos en entrenamiento y prueba

En la Figura 39 se muestra la implementación de la función `train_test_split()`, que divide el conjunto de datos en dos subconjuntos: uno para entrenamiento y otro para prueba. El 80 % de los datos se destina al entrenamiento, mientras que el 20 % restante se reserva para evaluar el desempeño del modelo.

Entre los parámetros configurados destaca `random_state=60`, que asegura la reproducibilidad de la división, permitiendo que esta partición sea consistente en ejecuciones futuras. Además, `test_size=0.2` define explícitamente que el 20 % de los datos se usará para la prueba.

Este procedimiento es fundamental para entrenar el modelo con la mayoría de los datos, pero también para evaluarlo con un conjunto independiente, lo que permite una valoración objetiva y robusta de su rendimiento.

Figura 39

Fragmento de código: implementación de la función `train_test_split()`

```
[ ] from sklearn.model_selection import train_test_split
    train_features, test_features, train_targets, test_targets = train_test_split(features, targets, random_state = 60, test_size=0.2)
    train_features.shape, test_features.shape
```

Estandarización de los datos

Antes de entrenar el modelo, en la Figura 40 se realiza la estandarización de los datos mediante la función `StandardScaler()`. Este proceso transforma las características para que tengan una media de 0 y una desviación estándar de 1.

La estandarización es un paso crucial al trabajar con algoritmos sensibles a las diferentes escalas de las variables de entrada, pues garantiza que cada característica aporte equitativamente al entrenamiento. Aunque los árboles de decisión suelen ser menos afectados por la escala, aplicar esta transformación puede mejorar la estabilidad y el rendimiento del proceso de aprendizaje.

Figura 40

Fragmento de código: estandarización de los datos mediante la función `StandardScaler()`.

```
[ ] from sklearn.preprocessing import StandardScaler
    sc = StandardScaler()
    train_features = sc.fit_transform(train_features)
    test_features = sc.fit_transform(test_features)
```

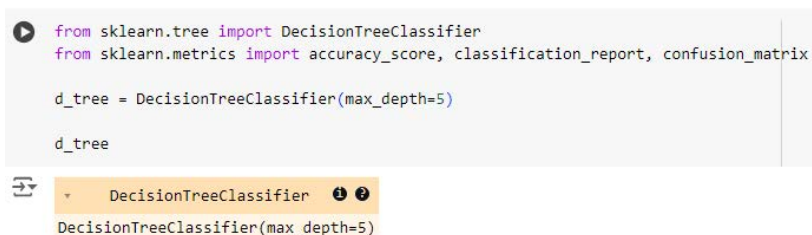
Creación del modelo de árbol de decisión

En la Figura 41 se crea el modelo de árbol de decisión usando la función `DecisionTreeClassifier()` de Scikit-learn. Se establece un parámetro clave: la profundidad máxima del árbol se limita a 5. Esta restricción busca evitar el sobreajuste (*overfitting*), que ocurre cuando el modelo se ajusta demasiado a los datos de entrenamiento, perdiendo capacidad de generalización sobre datos nuevos.

Limitar la profundidad contribuye a construir un modelo más robusto y con mejor desempeño predictivo en escenarios reales.

Figura 41

Fragmento de código: creación del modelo de árbol de decisión empleando la función `DecisionTreeClassifier()` de la biblioteca Scikit-learn



```
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

d_tree = DecisionTreeClassifier(max_depth=5)

d_tree
```

The screenshot shows a code editor with the above Python code. Below the code, there is a variable explorer or console area showing the object `DecisionTreeClassifier` with its parameters `DecisionTreeClassifier(max_depth=5)`.

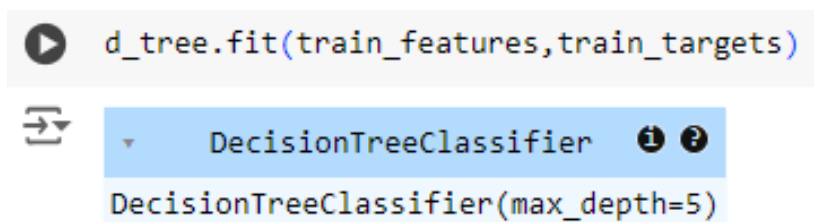
Entrenamiento del modelo

Una vez definido el modelo, el entrenamiento consiste en ajustar sus parámetros para que capture patrones presentes en los datos. La Figura 42 muestra la ejecución del método `fit()`, que toma como entrada el conjunto de entrenamiento (características y etiquetas). Durante este proceso, el árbol aprende a segmentar el espacio de características de forma que se minimice el error en la clasificación.

Este paso es clave para que el modelo genere reglas internas basadas en las características que luego usará para realizar predicciones sobre nuevos datos.

Figura 42

Fragmento de código: entrenamiento de modelo de árbol de decisión utilizando el método fit() aplicado sobre el conjunto de datos de entrenamiento



Generación y evaluación de predicciones

Después del entrenamiento, es necesario probar el modelo con datos nuevos para medir su capacidad predictiva. En la Figura 43, el método `predict()` es utilizado para generar predicciones sobre el conjunto de prueba. Las predicciones obtenidas son comparadas con los valores reales mediante la función `classification_report()`.

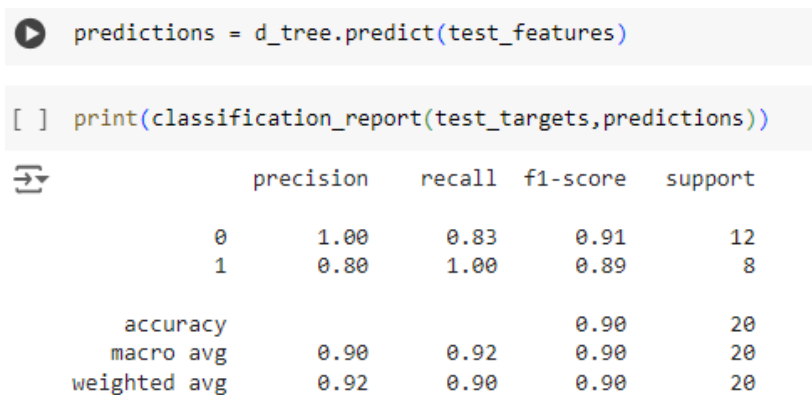
Este informe ofrece métricas esenciales para la evaluación, tales como:

- **Precisión (precision):** proporción de predicciones correctas positivas.
- **Sensibilidad (recall):** capacidad del modelo para identificar correctamente las instancias positivas.
- **Puntaje F1:** medida armónica que equilibra precisión y *recall*.

Estas métricas brindan una visión detallada de la efectividad del modelo en la clasificación de las distintas categorías.

Figura 43

Fragmento de código: función predict() emplea el modelo previamente entrenado para generar predicciones sobre el conjunto de prueba



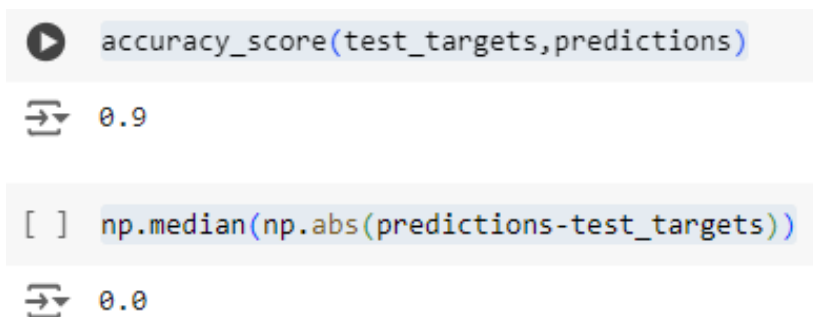
Cálculo de métricas de precisión y mediana del error absoluto

Para evaluar el modelo desde diferentes perspectivas, se calculan métricas que complementan la evaluación global. En la Figura 44, se utiliza `accuracy_score(test_targets, predictions)` para determinar la precisión global del modelo, definida como la proporción de predicciones correctas respecto al total. También se calcula la mediana del error absoluto usando `np.median(np.abs(predictions-test_targets))`, que proporciona una medida robusta del error típico de las predicciones, siendo menos sensible a valores atípicos que la media.

Juntas, estas métricas permiten comprender tanto la exactitud general como la variabilidad típica en la predicción.

Figura 44

Fragmento de código: fódigos utilizados para evaluar la precisión del modelo de clasificación



```
accuracy_score(test_targets, predictions)

0.9

[ ] np.median(np.abs(predictions-test_targets))

0.0
```

Visualización con matriz de confusión

La matriz de confusión es una herramienta fundamental para analizar el desempeño detallado de un clasificador. La Figura 45 presenta la matriz de confusión, generada a partir de los valores reales (`test_targets`) y las predicciones del modelo (`predictions`). Esta matriz permite observar el número de:

- Verdaderos negativos (VN): 10 instancias de la clase 0 que fueron correctamente clasificadas como clase 0.
- Falsos positivos (FP): 2 instancias de la clase 0 que fueron incorrectamente clasificadas como clase 1.
- Falsos negativos (FN): 0 instancias de la clase 1 que fueron incorrectamente clasificadas como clase 0.
- Verdaderos positivos (VP): 8 instancias de la clase 1 que fueron correctamente clasificadas como clase 1.

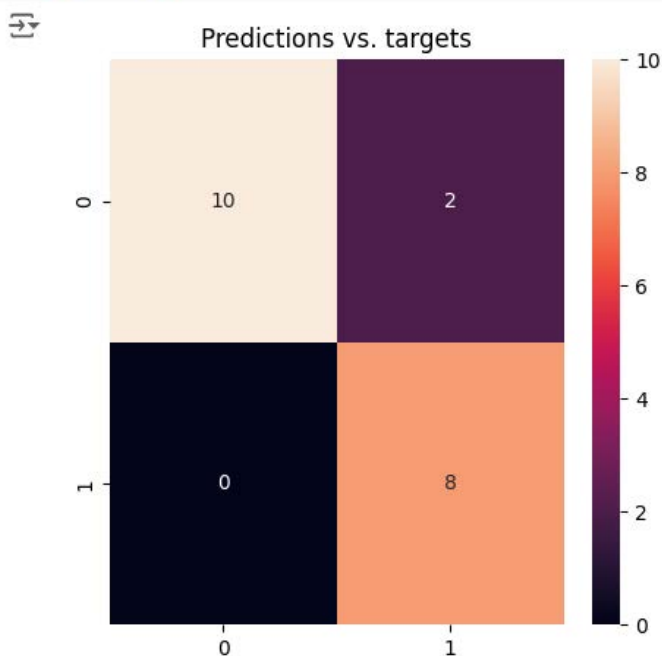
La figura 45 muestra un mapa de calor de 2x2, donde las filas representan las clases verdaderas (*targets*) y las columnas representan las predicciones del modelo. El tono de color indica

la frecuencia de las clasificaciones, y cada celda contiene el valor numérico correspondiente. Este desglose permite identificar patrones de error específicos y evaluar la eficacia del modelo para cada clase. En este caso, se observa que el modelo tiene un buen rendimiento general, aunque presenta dos errores de clasificación al predecir como positivos algunos casos negativos.

Figura 45

Matriz de confusión para análisis detallado del rendimiento

```
plt.figure(figsize=(5,5))  
sns.heatmap(confusion_matrix(test_targets,predictions),annot=True)  
plt.title("Predictions vs. targets")  
plt.show()
```



Visualización gráfica del árbol de decisión

Visualizar el árbol de decisión entrenado permite comprender cómo el modelo realiza sus divisiones y qué variables son determinantes en la clasificación. Esta representación gráfica facilita la interpretación del modelo, mostrando la estructura de decisiones y la pureza de las clases en cada nodo.

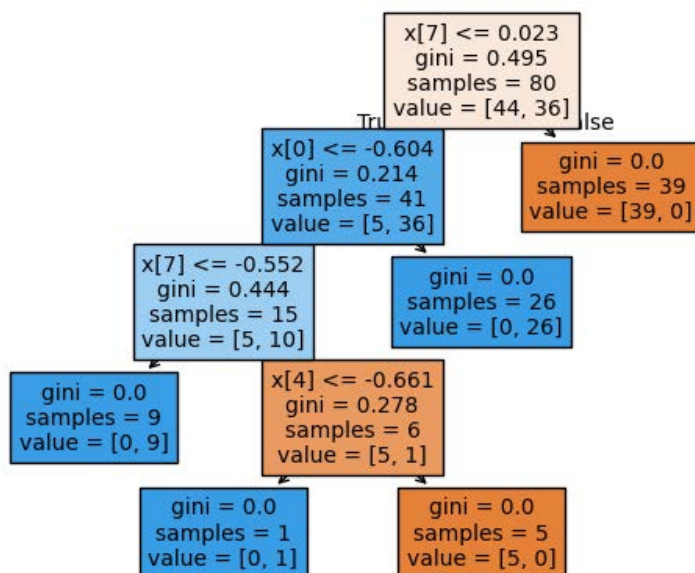
En la Figura 46, se utiliza la biblioteca `sklearn` para representar visualmente el árbol. Primero, se importa el módulo `tree` de `sklearn`, que provee las herramientas necesarias para crear y manipular árboles de decisión. Luego, se crea un espacio para la visualización mediante `plt.figure()`. Finalmente, la función `tree.plot_tree()` genera el gráfico del árbol almacenado en la variable `d_tree`.

La opción `filled=True` colorea los nodos según la pureza de las clases que contienen, haciendo que la interpretación visual sea más intuitiva y facilitando la identificación de las ramas clave del modelo.

Figura 46

Fragmento de código: visualización gráfica del árbol de decisión con nodos coloreados por pureza

```
from sklearn import tree
fig = plt.figure()
_ = tree.plot_tree(d_tree, filled = True)
```



Código relacionado con los resultados de predicción

A continuación, se presentan las líneas de código utilizadas para generar las predicciones del modelo y evaluar su desempeño sobre datos no utilizados durante el entrenamiento.

En la Figura 47, se muestra el código relacionado con los resultados de las predicciones. La variable *predictions* contiene

las etiquetas predichas por el modelo de árbol de decisión (`d_tree`) para el conjunto de prueba `test_features`. Este subconjunto de datos representa únicamente las características extraídas del conjunto total, reservadas específicamente para la etapa de evaluación.

El uso de `test_features` permite validar objetivamente el modelo, ya que contiene instancias no vistas durante el entrenamiento. De esta manera, es posible medir con mayor precisión la capacidad del modelo para clasificar correctamente las observaciones en las categorías correspondientes, por ejemplo: "pedir" o "no pedir".

Figura 47

Fragmento de código: líneas de código relacionadas con los resultados de las predicciones

```
[ ] predictions
```

```
array([1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1])
```

```
test_features
```

Finalmente, en la última sección del código, mostrada en la Figura 50, se utiliza el modelo entrenado (`d_tree`) para realizar una predicción con un nuevo conjunto de características que se ingresan manualmente en forma de un arreglo. Primero, se crea un arreglo de NumPy con los valores `np.array([[1, 67, 0.6, 6, 44, 0.3, 39, 3.469154]])`, donde cada elemento representa una característica específica: inventario inicial "x" (1), costo de penalización "p" (67), factor de descuento " α " (0.6), precio de venta por unidad "r" (6), precio de compra por unidad "c" (44), costo de almacenamiento "h" (0.3), demanda máxima "D_max" (39) y nivel óptimo (3.469154).

A continuación, el arreglo es estandarizado utilizando `sc.fit_transform()` para asegurar que las nuevas características estén en la misma escala que los datos con los que se entrenó el modelo. Finalmente, con `d_tree.predict()`, el modelo genera una predicción que indica si, basándose en estas características, se debe "pedir" (1) o "no pedir" (0). En este caso, la predicción resultante es "pedir" (1), lo que indica que, bajo las condiciones especificadas, la recomendación es realizar un pedido.

Predicción con nuevos datos ingresados manualmente

El modelo entrenado puede ser utilizado para realizar predicciones a partir de nuevos datos ingresados manualmente. Esta funcionalidad resulta especialmente útil en escenarios reales para apoyar la toma de decisiones operativas, como determinar si es conveniente realizar un pedido bajo ciertas condiciones.

En la Figura 48, se muestra el código correspondiente a esta operación. Primero, se crea un arreglo de NumPy con valores específicos que representan cada una de las características relevantes: `np.array([[1, 67, 0.6, 6, 44, 0.3, 39, 3.469154]])`

Cada uno de estos valores corresponde a las siguientes variables:

- Inventario inicial (1)
- Costo de penalización p (67)
- Factor de descuento α (0.6)
- Precio de venta por unidad r (6)
- Precio de compra por unidad c (44)
- Costo de almacenamiento h (0.3)
- Demanda máxima $D_{\max}$ (39)

- Nivel óptimo (3.469154)

Antes de realizar la predicción, los datos se estandarizan utilizando `sc.fit_transform()` para garantizar que estén en la misma escala que los utilizados durante el entrenamiento del modelo. Este paso es crucial para mantener la coherencia en la interpretación de las características por parte del modelo.

Finalmente, mediante la instrucción `d_tree.predict()`, el modelo de árbol de decisión emite una recomendación binaria:

- **1** → "Pedir"
- **0** → "No pedir"

En este caso particular, la predicción obtenida es "pedir" (1), lo que sugiere que, dadas las condiciones establecidas en las variables de entrada, lo recomendable es realizar el pedido.

Figura 48

Fragmento de código: modelo entrenado (d_tree) para realizar una predicción con un nuevo conjunto de características que se ingresan manualmente en forma de un arreglo

```
nuevo = d_tree.predict(sc.fit_transform(np.array([[ 1, 67, 0.6, 6, 44, 0.3, 39, 3.469154 ]]]))
nuevo
array([1])
```

Conclusión y consideraciones finales

El programa desarrollado, basado en árboles de decisión para clasificación, cumple eficazmente su propósito inicial: determinar si se debe realizar un pedido o no. Esta funcionalidad representa una etapa fundamental dentro del proceso de gestión de inventarios, ya que permite tomar decisiones automatizadas

y objetivas en función de condiciones específicas del entorno logístico.

Sin embargo, una limitación clave del modelo de clasificación es que no proporciona la cantidad óptima a solicitar cuando la decisión resultante es "pedir". Para superar esta restricción, se diseñó un programa complementario basado en un modelo de árbol de decisión para predicción numérica. Este modelo adicional permite estimar el valor específico de la cantidad a ordenar, y puede ser aplicado tanto en escenarios de un solo periodo como en modelos probabilísticos y multiperiodo, tales como los modelos tipo S-s.

Gracias a esta extensión, se logra una mayor precisión y aplicabilidad del sistema en contextos reales, al integrar tanto la decisión binaria como la estimación cuantitativa, aspectos esenciales para una gestión de inventarios más robusta.

Finalmente, el código completo y documentado de ambos programas —clasificación y predicción— se encuentra disponible en el Anexo 4. Este anexo incluye una descripción detallada de los procesos implementados, lo que facilita su comprensión, análisis y replicación por parte de otros investigadores o profesionales interesados en el desarrollo de sistemas de apoyo a la decisión en logística.

Análisis de la precisión y fiabilidad en modelos determinísticos y probabilísticos

Una vez desarrollados los programas para el cálculo de los modelos determinísticos y probabilísticos, se realizaron simulaciones con el objetivo de evaluar su efectividad. Aunque cada uno de estos programas —tanto los basados en árboles de decisión como los de clasificación— proporciona métricas internas que indican

el nivel de precisión alcanzado en las predicciones, resulta fundamental efectuar una comparación directa con los resultados obtenidos mediante el programa que utiliza fórmulas matemáticas exactas, las cuales representan el cálculo preciso de los modelos (Géron, 2023).

Para esta evaluación comparativa se seleccionaron cuatro modelos representativos: dos determinísticos y dos probabilísticos. En cada caso, se calcularon los resultados utilizando tanto el programa basado en árboles de decisión como el programa de referencia con resultados exactos. Esta doble aproximación permitió realizar un análisis detallado entre ambos enfoques, con el fin de medir la precisión, coherencia y fiabilidad de los modelos predictivos frente a los métodos tradicionales.

Este análisis es clave para validar la utilidad práctica y la efectividad de las soluciones algorítmicas desarrolladas, especialmente en contextos donde los métodos exactos pueden ser poco flexibles o inviables debido a la complejidad o volumen de datos. De esta manera, se establece un marco sólido para justificar el uso de modelos inteligentes como herramientas de apoyo en la toma de decisiones en la gestión de inventarios.

Modelos de inventarios determinísticos

En la gestión de inventarios, el modelo clásico de cantidad económica de pedido (EOQ) es fundamental para optimizar el tamaño del lote que minimiza los costos totales asociados al pedido y al almacenamiento. Tradicionalmente, este modelo se basa en parámetros conocidos y constantes, ofreciendo soluciones exactas mediante fórmulas matemáticas bien establecidas.

Sin embargo, con el avance de la inteligencia artificial, se han desarrollado enfoques alternativos que emplean técnicas

predictivas, como los árboles de decisión, para estimar estos parámetros y adaptarse mejor a escenarios con variabilidad o datos incompletos.

En la Tabla 9 se presenta un análisis comparativo del modelo EOQ bajo estos dos enfoques: un programa basado en árboles de decisión predictivos y un programa que utiliza fórmulas matemáticas exactas. Para garantizar una comparación equitativa, se realizaron 10 simulaciones con cada programa utilizando los mismos datos de entrada.

Este análisis permite evaluar la capacidad del modelo predictivo para replicar los resultados exactos del enfoque tradicional en un contexto determinístico, validando su precisión y aplicabilidad cuando los parámetros son conocidos y constantes.

Tabla 9

Resultados comparativos EOQ: árbol de decisiones vs. programa

Modelo clásico de cantidad económica de pedido							
D	K	H	L	Árbol de decisiones		Programa	
				Cantidad pedido	Punto de reorden	Cantidad pedido	Punto de reorden
784	68,13	0,27	1	315,67	29,01	629,01	154,99
659	49,38	0,07	2	263,63	71,77	964,24	353,76
1753	90,26	2,50	10	258,79	201,88	355,78	96,70
1316	96,73	2,34	10	283,83	15,56	329,85	295,88
935	44,51	2,61	1	188,90	40,39	178,58	42,11
863	81,26	2,94	11	210,05	131,27	218,42	101,10
1831	57,60	2,40	5	210,38	138,51	296,46	711,50
1483	61,12	1,39	8	289,96	194,21	361,14	307,67
1133	93,30	2,34	4	286,30	237,97	300,58	23,36
1847	16,39	0,36	3	309,04	332,187	410,10	209,74

Análisis de precisión mediante diferencias cuadráticas

En la Tabla 10 se presentan las diferencias cuadráticas entre las predicciones de cantidad de pedido y punto de reorden, generadas tanto por el modelo basado en árboles de decisión como por el enfoque con fórmulas matemáticas exactas. Este análisis tiene como propósito evaluar la precisión relativa de cada método, cuantificando el grado de desviación respecto a los valores obtenidos por el modelo analítico tradicional.

Tabla 10

MSE de modelo clásico de cantidad económica de pedido

Diferencia cuadrática (cantidad de pedido)	Diferencia cuadrática (punto de reorden)
98181,9556	15870,9604
490854,3721	79518,3601
9407,0601	11062,8324
2117,8404	78579,3024
106,5024	2,9584
70,0569	910,2289
7409,7664	328317,5401
5066,5924	12873,1716
203,9184	46057,4521
10213,1236	14,993
62363,11883	58818,60742

Si bien previamente se evidenciaron discrepancias entre los resultados de ambos enfoques, se decidió realizar un análisis

detallado del error cuadrático medio (MSE), a fin de proporcionar una medida estadística robusta sobre la dispersión entre las predicciones y los valores de referencia.

En la columna correspondiente a la Diferencia Cuadrática de la Cantidad de Pedido, se observan variaciones significativas, con un valor mínimo de 106,50 y un máximo de 490854,37. Estos resultados indican que, aunque algunas predicciones del árbol de decisión se acercan al valor exacto, existen también casos con discrepancias marcadas, que afectan negativamente la consistencia general del modelo. La presencia de valores atípicos incrementa la dispersión global, aunque se evidencia una tendencia hacia cierta estabilidad en la mayoría de los casos.

De forma similar, la columna correspondiente a la diferencia cuadrática del punto de reorden muestra también una alta variabilidad, con valores que oscilan entre 2,96 y 328317,54. Aunque en este caso los extremos son menos elevados que en la cantidad de pedido, el grado de dispersión sigue siendo considerable, lo que refleja limitaciones importantes en la capacidad predictiva del modelo basado en árboles de decisión en cuanto a este indicador.

Los errores cuadráticos medios (MSE) calculados fueron de 62363,12 para la cantidad de pedido y 58818,61 para el punto de reorden. Estos valores sugieren que, en promedio, las predicciones realizadas mediante el árbol de decisión presentan una desviación estadísticamente significativa respecto a los resultados exactos. Si bien el modelo puede ser considerado como una herramienta de aproximación inicial en escenarios complejos, sus limitaciones en términos de precisión deben tenerse en cuenta al compararlo

con los métodos analíticos clásicos, especialmente en contextos donde la exactitud es crítica para la toma de decisiones logísticas.

Comparación del modelo EOQ con descuentos por volumen

En la Tabla 11 se presenta un análisis comparativo del modelo de cantidad económica de pedido (EOQ) con descuentos por volumen, también conocido como modelo con discontinuidad de precios. Para ello, se aplicaron dos enfoques: por un lado, el programa basado en árboles de decisión predictivos, y por otro, el programa original fundamentado en fórmulas matemáticas exactas.

Tabla 11

Resultados comparativos EOQ con descuento: árbol de decisiones vs. programa

Cantidad económica de pedido con discontinuidades de precio										
D	K	c1	c2	H	Q	L	Árbol de decisiones		Programa	
							Cantidad pedido	Punto de reorden	Cantidad pedido	Punto de reorden
137	528	8,44	6,63	0,11	953	1	1063,00	176,00	1146,00	137,00
335	410	13,02	9,06	0,15	332	4	1434,00	272,00	1353,00	1340,00
172	301	18,71	8,80	0,08	715	2	1063,00	92,00	1137,00	344,00
867	198	11,86	8,81	0,39	666	2	837,00	305,00	938,00	795,00
815	296	11,46	4,91	0,21	72	4	1052,00	1414,00	1515,00	228,00
745	274	19,09	3,41	0,03	827	3	3356,00	1161,00	3688,00	2235,00
947	377	16,68	7,69	0,32	757	1	1084,00	332,00	1493,00	947,00
244	488	15,74	7,15	0,33	176	1	507,00	171,00	849,00	244,00

229	272	17,04	7,70	0,16	329	3	1510,00	220,00	882,38	687,00
683	510	6,39	5,88	0,23	431	3	1610,00	595,00	1740,00	308,00

El objetivo de este análisis fue evaluar la precisión y concordancia entre ambos enfoques en el contexto específico de políticas de descuento por cantidad, donde los precios unitarios varían en función del volumen de pedido. Este tipo de modelo introduce una mayor complejidad en el proceso de decisión, ya que no solo se debe determinar la cantidad óptima de pedido, sino también identificar correctamente el tramo de precios correspondiente.

Para asegurar una comparación justa y consistente, se realizaron 10 simulaciones en cada programa utilizando los mismos conjuntos de datos de entrada. Esto permitió analizar cómo cada enfoque se comporta frente a escenarios con estructuras de precios escalonadas, y hasta qué punto el modelo basado en inteligencia artificial es capaz de replicar los resultados del método tradicional.

Los resultados obtenidos permiten observar el grado de aproximación del modelo predictivo respecto al modelo exacto, así como identificar posibles limitaciones o ventajas del enfoque basado en árboles de decisión en la resolución de este tipo de problemas logísticos.

Análisis de la precisión del modelo EOQ con descuentos por volumen

En la Tabla 12 se presentan las diferencias cuadráticas entre las predicciones de cantidad de pedido y punto de reorden, obtenidas mediante los dos enfoques analizados: el programa basado en árboles de decisión predictivos y el programa desarrollado con fórmulas matemáticas exactas. Este análisis

cuantitativo permite evaluar la precisión de cada método, midiendo el grado de desviación entre los valores estimados por el modelo predictivo y los resultados de referencia generados mediante el enfoque analítico.

Tabla 12

MSE de cantidad económica de pedido con discontinuidades de precio

	Diferencia cuadrática (cantidad de pedido)	Diferencia cuadrática (punto de reorden)
	6889	1521
	6561	1140624
	5476	63504
	10201	240100
	214369	1406596
	110224	1153476
	167281	378225
	116964	5329
	393906,8644	218089
	16900	82,369
MSE	104877,1864	468983,3

Este procedimiento resulta particularmente relevante en el contexto del modelo EOQ con descuentos por volumen, dado que introduce mayores niveles de complejidad debido a la existencia de tramos de precios escalonados, lo cual exige una correcta identificación de las condiciones más convenientes para cada nivel de demanda.

En la columna de diferencia cuadrática (cantidad de pedido) se observan valores que oscilan entre un mínimo de 6889 y un máximo de 393906,86. Estos resultados evidencian que, aunque en algunos casos las predicciones generadas por el árbol

de decisión son relativamente precisas, también se presentan discrepancias notables en escenarios más complejos. El rango de variación indica una tendencia general a la consistencia, pero con la presencia de errores significativos que podrían deberse a la forma en que el modelo interpreta los tramos de descuento y su impacto en la optimización de la cantidad a pedir.

Por otro lado, la columna de diferencia cuadrática (punto de reorden) muestra una variabilidad aún más acentuada, con valores que van desde 5329 hasta un máximo de 1400.596, lo cual refleja una alta dispersión y, por ende, menor precisión en las estimaciones de esta variable por parte del modelo predictivo. Este comportamiento sugiere que la estimación del punto de reorden es especialmente sensible a errores en condiciones más atípicas o fuera del rango aprendido por el modelo durante su entrenamiento.

Los valores de error cuadrático medio (MSE) obtenidos refuerzan esta interpretación. para la cantidad de pedido, el MSE calculado fue de 104877,19, lo que indica que las predicciones presentan, en promedio, una diferencia cuadrática significativa respecto a los valores exactos. En el caso del punto de reorden, el MSE asciende a 468983,30, lo que representa un error promedio aún más elevado, y evidencia que esta variable presenta mayores limitaciones en su predicción bajo el enfoque de árboles de decisión.

En conclusión, aunque el modelo predictivo basado en inteligencia artificial muestra potencial como herramienta de apoyo para estimaciones iniciales, los resultados reflejan que su precisión es considerablemente inferior a la del método exacto en escenarios de mayor complejidad como los modelos con descuentos por volumen. Por tanto, se identifica un área crítica de mejora, especialmente en lo que respecta a la predicción del punto de reorden.

Modelos de inventarios probabilísticos

En entornos logísticos reales, la incertidumbre y variabilidad en la demanda hacen que los modelos determinísticos tradicionales sean insuficientes para una gestión óptima de inventarios. Por ello, los modelos probabilísticos surgen como herramientas clave que incorporan la aleatoriedad de la demanda y otros factores, permitiendo decisiones más precisas y adaptativas.

En particular, el modelo probabilístico de cantidad económica de pedido (EOQ) integra variables estadísticas como la desviación estándar de la demanda para calcular no solo el tamaño óptimo del pedido, sino también el punto de reorden y el *stock* de seguridad necesario para mantener niveles adecuados de servicio.

Esta sección presenta un análisis comparativo entre dos enfoques para la estimación de estos parámetros bajo incertidumbre: un sistema basado en árboles de decisión predictivos, representando técnicas modernas de inteligencia artificial, y el método clásico que emplea fórmulas matemáticas exactas. La comparación se realiza a través de múltiples simulaciones, permitiendo evaluar la precisión, consistencia y aplicabilidad de los modelos predictivos en escenarios probabilísticos.

Evaluación del modelo EOQ probabilístico

La gestión eficiente de inventarios es crucial para optimizar costos y garantizar el nivel de servicio en cadenas de suministro. Dentro de los modelos clásicos, la cantidad económica de pedido (EOQ) se utiliza para determinar el tamaño óptimo de los pedidos que minimizan los costos totales de inventario. Sin embargo,

en entornos reales, la demanda suele ser incierta y variable, lo que hace necesario emplear modelos probabilísticos que incorporen esta variabilidad para una toma de decisiones más precisa.

En este contexto, la Tabla 13 presenta un análisis comparativo del modelo EOQ bajo un enfoque probabilístico. Se confrontan los resultados obtenidos por un sistema basado en árboles de decisión predictivos —una técnica de inteligencia artificial que permite manejar escenarios con incertidumbre— con los resultados generados por un programa que utiliza fórmulas matemáticas exactas, el método tradicional para el cálculo del EOQ probabilístico.

Tabla 13
Resultados Comparativos modelo probabilizado: árbol de decisiones vs. programa

Modelo probabilizado de cantidad económica de pedido							
L	D	Σ	A	Árbol de decisiones		Programa	
				Punto de reorden	Stock de seguridad	Punto de reorden	Stock de seguridad
2	159	35	0,039	537,58	137,77	295,59	72,99
6	113	29	0,023	897,22	133,18	819,74	141,74
7	168	36	0,049	1315,03	135,27	1333,60	157,60
8	193	31	0,011	1717,23	160,03	1744,82	200,82
9	155	23	0,034	1467,6	141,71	1520,93	125,93
8	186	39	0,015	1680,33	174,24	1727,38	239,38
2	121	27	0,044	313,9	53,4	307,14	65,14
1	174	33	0,018	235,53	59,56	243,20	69,20
6	142	25	0,027	918,27	133,18	969,99	117,99
1	108	38	0,041	150,01	84,9	174,09	66,09

El propósito de esta comparación es evaluar la precisión, consistencia y capacidad del modelo predictivo para replicar los resultados del método exacto, especialmente en situaciones donde las condiciones de entrada son más complejas y variables. La demanda incierta implica que el modelo debe adaptarse a datos dinámicos y menos estructurados, incluyendo variables aleatorias como la desviación estándar, lo que representa un reto adicional para los modelos computacionales.

Para garantizar la equidad y validez del análisis, se realizaron diez simulaciones independientes en cada programa utilizando los mismos datos de entrada. Esta metodología permitió una comparación uniforme, asegurando que las diferencias observadas reflejen las características propias de cada enfoque y no variaciones en los datos de entrada.

Este estudio es fundamental para determinar el potencial y las limitaciones del uso de modelos predictivos basados en inteligencia artificial en la gestión de inventarios bajo incertidumbre, destacando la necesidad de modelos robustos y adaptativos que puedan operar eficazmente en entornos logísticos reales.

Análisis de precisión del modelo EOQ probabilístico: diferencias cuadráticas

Con el propósito de evaluar la precisión del modelo EOQ probabilístico desarrollado mediante árboles de decisión, se realizó un análisis comparativo utilizando diferencias cuadráticas. Este enfoque permitió medir el grado de desviación entre las predicciones generadas por el modelo basado en inteligencia artificial y los resultados obtenidos por un programa exacto que utiliza fórmulas matemáticas tradicionales.

El análisis se enfocó en dos variables logísticas clave: el punto de reorden y el *stock* de seguridad, en contextos donde la demanda es incierta y variable.

La Tabla 14 presenta las diferencias cuadráticas obtenidas para cada variable en 10 simulaciones realizadas con los mismos datos de entrada. Esta metodología permitió asegurar la equidad en la evaluación entre ambos métodos. Las diferencias cuadráticas permiten analizar no solo la precisión, sino también la consistencia del modelo predictivo frente al enfoque tradicional.

Tabla 14

MSE del modelo probabilizado de cantidad económica de pedido

	Diferencia cuadrática (punto de reorden)	Diferencia cuadrática (<i>stock</i> de seguridad)
	58559,1601	4196,4484
	6003,1504	73,2736
	344,8449	498,6289
	761,2081	1663,8241
	2844,0889	249,0084
	2213,7025	4243,2196
	45,6976	137,8276
	58,8289	92,9296
	2674,9584	230,7361
	579,8464	354
MSE	7408,54862	1173,97124

En el caso del punto de reorden, las diferencias cuadráticas oscilaron entre 344,84 y 58559,16. Este amplio rango evidencia

que, aunque algunas predicciones son cercanas a los valores reales, existen casos con discrepancias significativas. Esto sugiere que el modelo presenta cierta inestabilidad al estimar esta variable, especialmente en escenarios complejos o con alta incertidumbre.

Por otro lado, las diferencias cuadráticas del *stock* de seguridad fueron notablemente más bajas, variando entre 73,27 y 4196,45. Estos resultados reflejan una mayor consistencia y precisión en las predicciones del modelo para esta variable, lo cual indica que el enfoque basado en árboles de decisión se adapta mejor al cálculo del inventario adicional necesario para cubrir la variabilidad en la demanda.

Los errores cuadráticos medios (MSE) refuerzan este análisis: mientras que el MSE del punto de reorden fue de 7408,55, el del *stock* de seguridad fue de solo 1173,97. Esta diferencia sustancial sugiere que el modelo predictivo tiene un desempeño más confiable al estimar el *stock* de seguridad, aunque requiere mejoras para alcanzar una mayor precisión en la predicción del punto de reorden bajo condiciones probabilísticas.

Análisis del modelo probabilístico de cantidad económica de pedido

La gestión de inventarios en contextos donde la demanda es incierta y variable representa un desafío importante para las organizaciones. En estos escenarios, los modelos probabilísticos se convierten en herramientas esenciales para optimizar las decisiones de compra y minimizar costos, considerando la fluctuación y la aleatoriedad en la demanda.

En la Tabla 15, se presenta un análisis comparativo del modelo probabilístico de cantidad económica de pedido (EOQ), en el que se evalúan los resultados obtenidos por un sistema basado en árboles de decisión predictivos frente a los generados

por un programa inicial que utiliza fórmulas matemáticas exactas. Esta comparación busca determinar la efectividad del modelo predictivo para aproximar soluciones confiables bajo condiciones de incertidumbre.

Para garantizar una comparación justa y consistente, se realizaron diez simulaciones en cada uno de los dos programas, empleando idénticos datos de entrada en todas las ejecuciones. De esta manera, se aseguraron condiciones homogéneas para ambos métodos, lo que permite atribuir cualquier diferencia en los resultados a las características inherentes de cada enfoque metodológico.

Tabla 15
Resultados Comparativos modelo probabilístico: Árbol de Decisiones vs Programa

Modelo probabilístico de cantidad económica de pedido									
f(x)	D	K	h	P	E{x}	Árbol de decisiones		Programa	
						Cantidad pedido	Punto de reorden	Cantidad pedido	Punto de reorden
54	1127	94	2	10	54	394,632	32,073	327,05	50,86
43	1010	107	1	9	43	319,438	94,95	466,01	40,79
59	873	112	3	11	59	252,761	33,582	257,69	54,25
40	1189	87	27	20	40	478,586	57,447	89,59	35,93
146	958	101	1	12	146	249,67	7,85	442,72	140,37
55	1076	119	3	9	55	214,54	38,46	294,69	49,97
48	892	96	2	11	48	335,856	67,174	294,071	45,12
57	1145	103	1	10	57	301,67	58,599	486,87	54,57
52	1034	88	2	8	52	319,48	32,54	303,56	48,18
45	967	263	3	12	45	317,82	46,82	414,17	40,18

Análisis de las diferencias cuadráticas en el modelo probabilístico

La precisión en la estimación de parámetros clave como la cantidad a pedir y el punto de reorden es fundamental para una gestión eficiente de inventarios, especialmente cuando se trabaja con modelos probabilísticos que consideran la incertidumbre en la demanda. Evaluar la consistencia de estas predicciones es vital para determinar la viabilidad de diferentes enfoques metodológicos.

En la Tabla 16 se presentan las diferencias cuadráticas entre las predicciones generadas por un modelo probabilístico basado en árboles de decisión y los valores calculados por un programa tradicional con fórmulas matemáticas exactas. Este análisis comparativo busca medir la precisión y estabilidad de cada método para estimar la cantidad a pedir y el punto de reorden.

Tabla 16

MSE del modelo probabilístico de cantidad económica de pedido

Diferencia cuadrática (cantidad a pedir)	Diferencia cuadrática (punto de reorden)
4567,326724	352,951369
21483,35118	2933,3056
24,295041	427,166224
151317,888	462,981289
37268,3025	17561,5504
6424,0225	132,4801
1745,986225	486,378916
34299,04	16,232841
253,4464	244,6096
9283,3225	44

Respecto a la cantidad a pedir, los valores de diferencia cuadrática muestran una dispersión considerable, con un rango que va desde 24,29 hasta 151317,89. Esta variabilidad sugiere que, aunque en algunos casos las predicciones están relativamente cercanas a los valores reales, existen escenarios donde las estimaciones presentan errores importantes y poca consistencia, lo cual puede afectar negativamente la toma de decisiones en la gestión de inventarios.

Por otro lado, las diferencias cuadráticas correspondientes al punto de reorden son menores y más uniformes, oscilando entre 16,23 y 17561,55. Esto indica que, en términos generales, las predicciones del punto de reorden tienden a ser más confiables y precisas, aunque se registran algunas desviaciones puntuales que deben ser consideradas.

Finalmente, los errores cuadráticos medios (MSE) refuerzan estas observaciones. El MSE para la cantidad a pedir es de 26666,70, evidenciando una desviación promedio significativa en estas predicciones, mientras que el MSE para el punto de reorden es mucho menor, con un valor de 2266,17, lo que refleja una mayor precisión y estabilidad en las estimaciones de este parámetro.

Propuesta de mejora del programa predictivo

El análisis estadístico realizado evidenció que el programa basado en árboles de decisión no logra ofrecer predicciones exactas ni se aproxima consistentemente a los resultados que se obtienen mediante cálculos matemáticos precisos. Estas discrepancias, aunque significativas, no son inesperadas, dado que este

programa se fundamenta en procesos de predicción y clasificación que dependen de datos históricos y patrones previos, en lugar de fórmulas matemáticas explícitas.

Es importante entender que este tipo de modelos predictivos está diseñado para generar estimaciones aproximadas y no resultados exactos. Por lo tanto, exigir una precisión del 100 % sería poco razonable y no se ajusta a la naturaleza inherente de los algoritmos basados en aprendizaje automático, que buscan capturar tendencias y comportamientos en los datos más que replicar cálculos exactos.

Con el fin de mejorar la precisión y reducir las diferencias frente a los métodos matemáticos exactos, se propone la implementación de redes neuronales como alternativa a los árboles de decisión. Esta técnica ofrece la capacidad de aprender patrones más complejos y relaciones no lineales presentes en los datos, algo que puede superar las limitaciones observadas en los modelos actuales.

El enfoque basado en redes neuronales tiene el potencial de incrementar notablemente la exactitud de las predicciones, generando un modelo más robusto y adaptable a la variabilidad inherente en los datos de inventarios. Así, se espera que esta mejora permita una gestión predictiva más fiable, acercando los resultados a los obtenidos mediante métodos tradicionales.

Programa con redes neuronales

Méndez (2004) desarrolló un sistema basado en redes neuronales artificiales para la gestión inteligente de inventarios. El modelo aborda tanto problemas de predicción (cantidad óptima de pedido, demanda futura) como de clasificación (decisiones tipo reabastecer/no reabastecer). La metodología incluye el preprocesamiento de los datos, la definición de la

arquitectura de la red (capas ocultas, funciones de activación), el entrenamiento con retropropagación y la validación cruzada. Los autores comparan el desempeño de la red neuronal con otros modelos como árboles de decisión y concluyen que las redes neuronales presentan mayor precisión en escenarios complejos debido a su capacidad de modelar relaciones no lineales.

En esta sección se presenta el desarrollo de un programa basado en redes neuronales, que abarca tanto tareas de predicción como de clasificación. Se explica detalladamente el código implementado, paso a paso, acompañando cada fragmento con una descripción clara de su funcionamiento y propósito dentro del modelo.

El objetivo de este enfoque es mejorar la precisión y la robustez en la gestión de inventarios, superando las limitaciones identificadas en los modelos previos basados en árboles de decisión. Las redes neuronales permiten captar relaciones no lineales y patrones complejos en los datos, contribuyendo así a obtener resultados más ajustados a la realidad.

A continuación, se expone el proceso completo de implementación, desde la preparación y preprocesamiento de los datos, pasando por la definición de la arquitectura de la red, hasta el entrenamiento, evaluación y uso del modelo para realizar predicciones y clasificaciones sobre nuevos conjuntos de datos.

Redes neuronales por predicción

Paul y Azeem (2011) propusieron un sistema basado en redes neuronales para optimizar el modelo clásico de cantidad económica de pedido (EOQ). La investigación parte del modelo EOQ tradicional, pero integra redes neuronales para ajustar dinámicamente los parámetros clave (demanda, costos de pedido y almacenamiento) frente a condiciones cambiantes.

El enfoque permite mejorar la toma de decisiones en inventarios, adaptándose a entornos con alta incertidumbre y proporcionando una alternativa a los métodos determinísticos clásicos. El sistema fue implementado en Python utilizando TensorFlow y Keras.

Para proceder con el desarrollo del programa basado en redes neuronales, se decidió tomar como referencia el modelo clásico de cantidad económica de pedido (EOQ). Este modelo fue seleccionado debido a su amplia aplicabilidad en la gestión de inventarios y su reconocimiento como una herramienta fundamental para optimizar los costos asociados a pedidos y almacenamiento.

Utilizar el EOQ como punto de partida permite simplificar el desarrollo del programa y, a su vez, asegurar que el modelo implementado sea representativo y relevante para escenarios empresariales reales, facilitando así la validación y comparación de resultados con enfoques tradicionales.

Inicio del modelo: importación de bibliotecas y preparación del entorno

Antes de construir y entrenar un modelo de redes neuronales para calcular la cantidad óptima a pedir, es fundamental preparar el entorno de trabajo con las herramientas adecuadas. Esta etapa inicial garantiza que el código pueda manejar datos, realizar cálculos matemáticos y preparar los datos correctamente para el entrenamiento del modelo.

La Figura 49 muestra el fragmento de código donde se importan las bibliotecas esenciales para el desarrollo del proyecto. Se incluye:

- NumPy, una biblioteca poderosa para el manejo de arreglos (*arrays*) y operaciones matemáticas.

- Pandas, que facilita la carga y manipulación de datos tabulares, como los contenidos en archivos Excel.
- `train_test_split` de la biblioteca `sklearn`, que permite dividir el conjunto de datos en subconjuntos de entrenamiento y prueba. Esta separación es clave para evaluar el rendimiento del modelo de forma objetiva.
- `StandardScaler`, también de `sklearn`, que se utiliza para estandarizar los datos, asegurando que cada característica tenga una media de 0 y una desviación estándar de 1. Esta normalización es especialmente importante en redes neuronales, ya que mejora la eficiencia del entrenamiento y evita que ciertas variables dominen el aprendizaje por tener escalas mayores.

Estas bibliotecas y funciones conforman la base sobre la cual se construirá el modelo de predicción de inventarios utilizando inteligencia artificial.

Figura 49

Fragmento de código: importación de las bibliotecas necesarias para el correcto funcionamiento del programa – NumPy

```
[ ] import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

Carga y limpieza de datos desde Excel

Para trabajar con datos reales, es fundamental importar y preparar correctamente la información antes de iniciar el entrenamiento del modelo.

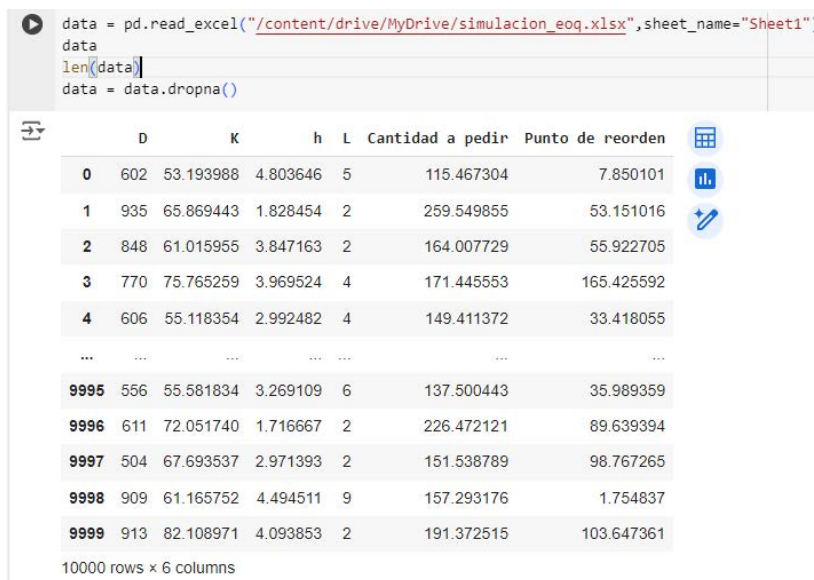
La Figura 50 muestra un fragmento de código donde se utiliza la función `pd.read_excel` para leer un archivo Excel, seleccionando la hoja específica llamada `Sheet1`. Esta operación permite cargar los datos estructurados en formato tabular, facilitando su manipulación posterior.

Posteriormente, se aplica `data.dropna()`, que elimina todas las filas que contienen valores nulos o vacíos. Este paso es crucial para evitar problemas durante el entrenamiento, ya que los datos incompletos pueden afectar negativamente el rendimiento y la precisión del modelo.

Así, se asegura que el conjunto de datos esté limpio y listo para los siguientes procesos de análisis y modelado.

Figura 50

Fragmento de código: código que utiliza `pd.read_excel` para leer el archivo Excel



```
data = pd.read_excel("/content/drive/MyDrive/simulacion_eoq.xlsx", sheet_name="Sheet1")
data
len(data)
data = data.dropna()
```

	D	K	h	L	Cantidad a pedir	Punto de reorden
0	602	53.193988	4.803646	5	115.467304	7.850101
1	935	65.869443	1.828454	2	259.549855	53.151016
2	848	61.015955	3.847163	2	164.007729	55.922705
3	770	75.765259	3.969524	4	171.445553	165.425592
4	606	55.118354	2.992482	4	149.411372	33.418055
...	...	...	...	...	...	...
9995	556	55.581834	3.269109	6	137.500443	35.989359
9996	611	72.051740	1.716667	2	226.472121	89.639394
9997	504	67.693537	2.971393	2	151.538789	98.767265
9998	909	61.165752	4.494511	9	157.293176	1.754837
9999	913	82.108971	4.093853	2	191.372515	103.647361

10000 rows x 6 columns

Definición de variables predictoras y variable objetivo

En el proceso de construcción del modelo, es esencial identificar y separar las variables que se utilizarán como entradas y la variable que se desea predecir.

La Figura 51 muestra cómo se define el conjunto de variables predictoras, denominadas X . Estas variables corresponden a las características que el modelo utilizará para realizar la predicción y en este caso incluyen:

- Demanda (D)
- Costo de pedido (K)
- Costo de mantenimiento (h)
- Tiempo de entrega (L)

Por otro lado, la variable objetivo, denominada y , corresponde a la columna que contiene la cantidad a pedir. Esta es la variable que el modelo intenta predecir con base en las características anteriores.

Esta separación clara entre variables de entrada y salida es fundamental para entrenar modelos de aprendizaje supervisado, ya que permite al modelo aprender las relaciones entre las características y el resultado esperado.

Figura 51

Fragmento de código: definición de variables predictoras y variable objetivo

```
[ ] X = data[["D","K","h","L"]]  
    y = data["Cantidad a pedir"]  
    X.shape
```

 (10000, 4)

División de los datos en conjuntos de entrenamiento y prueba

Para evaluar correctamente el rendimiento de un modelo de aprendizaje automático, es necesario dividir los datos disponibles en dos grupos: uno para entrenar el modelo y otro para probarlo.

En la Figura 52 se observa el uso de la función `train_test_split`, que realiza esta separación de manera automática. El conjunto de entrenamiento (`X_train`, `y_train`) se emplea para ajustar los parámetros internos de la red neuronal, mientras que el conjunto de prueba (`X_test`, `y_test`) se utiliza posteriormente para medir qué tan bien el modelo predice datos que no ha visto antes.

En este caso, se establece un parámetro `test_size=0.2`, lo que significa que el 20 % de los datos se reserva para la evaluación, dejando el 80 % restante para el entrenamiento. Además, el parámetro `random_state=42` asegura que esta división sea reproducible, es decir, que cada vez que se ejecute el código

se obtenga la misma partición, lo que facilita la comparación y validación de los resultados entre diferentes experimentos.

Esta etapa es clave para evitar sobreajuste y garantizar una evaluación objetiva del modelo.

Figura 52

Fragmento de código: división de los datos en conjuntos de entrenamiento y prueba

```
[ ] X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

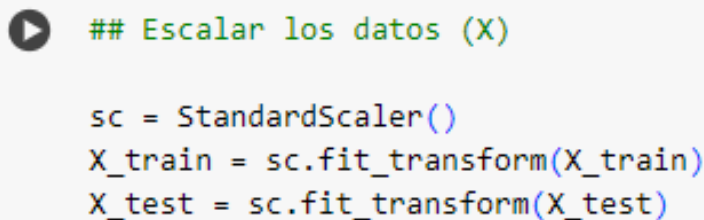
Estandarización de datos para redes neuronales

En el entrenamiento de redes neuronales, uno de los pasos cruciales es la normalización o estandarización de los datos. Este proceso permite que todas las variables de entrada se encuentren en la misma escala, lo cual es fundamental para evitar que alguna característica con valores numéricos mucho mayores influya desproporcionadamente en el aprendizaje del modelo.

La Figura 53 ilustra el inicio de este proceso mediante el uso de la clase `StandardScaler`. Esta herramienta transforma los datos para que tengan una media de 0 y una desviación estándar de 1. En primer lugar, se aplica la función `fit_transform` sobre el conjunto de entrenamiento (`X_train`), lo que permite ajustar el escalador a dichos datos y transformarlos simultáneamente. Luego, la función `transform` se emplea sobre el conjunto de prueba (`X_test`), utilizando los mismos parámetros calculados con los datos de entrenamiento. De esta forma, se garantiza que ambos conjuntos estén representados en una escala uniforme y comparable.

Figura 53

Fragmento de código: proceso de escalado de los datos utilizando la clase StandardScaler

A code snippet in a light gray box with a play button icon on the left. The code is in Python and demonstrates how to scale data using the StandardScaler class from sklearn.preprocessing. It shows the initialization of the scaler, fitting it to the training data, and then transforming both training and testing data.

```
## Escalar los datos (X)

sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

Construcción de una red neuronal con tensorflow

El uso de bibliotecas especializadas facilita significativamente el diseño y entrenamiento de redes neuronales. Entre ellas, TensorFlow se destaca por su capacidad para manejar tareas de aprendizaje profundo de forma eficiente y flexible.

La Figura 54 muestra cómo se implementa un modelo utilizando la estructura Sequential de TensorFlow, la cual permite construir redes neuronales añadiendo capas de manera lineal, es decir, una después de otra. En este caso, se incorpora una capa Dense, también conocida como capa completamente conectada. Esto implica que cada neurona en esta capa recibe información de todas las neuronas de la capa anterior, lo que asegura una comunicación completa entre capas. Este tipo de capa es esencial para que el modelo pueda capturar patrones complejos y realizar un procesamiento eficaz durante su etapa de aprendizaje.

Figura 54

Fragmento de código: implementación de un modelo utilizando la estructura Sequential de TensorFlow

```
▶ ## Crear la red neuronal

import tensorflow as tf

[ ] from tensorflow.keras.models import Sequential
    from tensorflow.keras.layers import Dense
```

Estructura de la red neuronal para la predicción de la cantidad a pedir

Una vez definido el enfoque general del modelo, es necesario detallar su arquitectura para comprender cómo procesa la información y genera predicciones útiles en el contexto de gestión de inventarios.

La Figura 55 presenta la estructura completa de la red neuronal. El modelo comienza con una capa de entrada que recibe cuatro características clave del sistema de inventario: la demanda (D), el costo de pedido (K), el costo de mantenimiento (h) y el tiempo de entrega (L). Esta capa utiliza la función de activación ReLU (rectified linear unit), ampliamente empleada en redes neuronales por su capacidad para introducir no linealidad, lo que permite al modelo aprender relaciones complejas en los datos.

Luego, se integran dos capas ocultas, cada una compuesta por 16 neuronas, que también emplean la función ReLU. Estas capas intermedias permiten que el modelo profundice su comprensión y representación de patrones internos más sofisticados.

Finalmente, el modelo concluye con una capa de salida compuesta por una sola neurona, ya que el objetivo final es predecir un único valor continuo: la cantidad óptima a pedir.

Figura 55

Fragmento de código: estructura completa de la red neuronal

```
## Red neuronal
model = Sequential()
model.add(Dense(4,activation = "relu",input_shape = (X_train.shape[1],)))
model.add(Dense(16,activation = "relu"))
model.add(Dense(16,activation = "relu"))
model.add(Dense(1))    ## si NN para predicción solo una neurona en la capa de salida
```

Configuración del entrenamiento del modelo: métricas y método de optimización

El éxito en el entrenamiento de modelos predictivos depende en gran medida de la adecuada selección del método de optimización y las métricas que guían el proceso de ajuste de parámetros. En este sentido, la elección del optimizador y la función de pérdida influye directamente en la velocidad, estabilidad y precisión del aprendizaje.

En la Figura 56 se presenta la configuración utilizada para el entrenamiento del modelo, donde se seleccionó el optimizador Adam (optimizer="adam"). Este optimizador es reconocido por su capacidad para manejar grandes conjuntos de datos y redes profundas, adaptándose dinámicamente a los cambios en el gradiente y facilitando un proceso de entrenamiento más eficiente y estable.

Como función de pérdida se eligió el error cuadrático medio (MSE) (loss="mse"), ideal para problemas de regresión, ya que penaliza fuertemente los errores grandes al calcular la

media de los cuadrados de las diferencias entre predicciones y valores reales.

Adicionalmente, se incorpora la métrica error absoluto medio (MAE) (`metrics=["mae"]`), que complementa la evaluación del desempeño midiendo la media de las diferencias absolutas entre predicciones y valores reales. MAE es menos sensible a valores atípicos, proporcionando una perspectiva equilibrada de la precisión del modelo.

Figura 56

Fragmento de código: importación de bibliotecas esenciales en Python

```
 ## Compilar la red  
model.compile(optimizer="adam", loss = "mse", metrics=["mae"])
```

Proceso de entrenamiento del modelo: parámetros y configuración

El entrenamiento de redes neuronales es un proceso iterativo que busca ajustar los parámetros internos del modelo para minimizar el error en las predicciones. La configuración de este proceso, incluyendo el número de ciclos (épocas), la división de datos para validación y la frecuencia de reporte, es fundamental para garantizar un aprendizaje efectivo y evitar problemas como el sobreajuste o el subentrenamiento.

En la Figura 57 se detalla la configuración inicial del entrenamiento de la red neuronal. El parámetro `epochs=1000` indica que el modelo realizará 1000 ciclos completos a través de todos los datos de entrenamiento, permitiendo así un proceso profundo de ajuste y optimización.

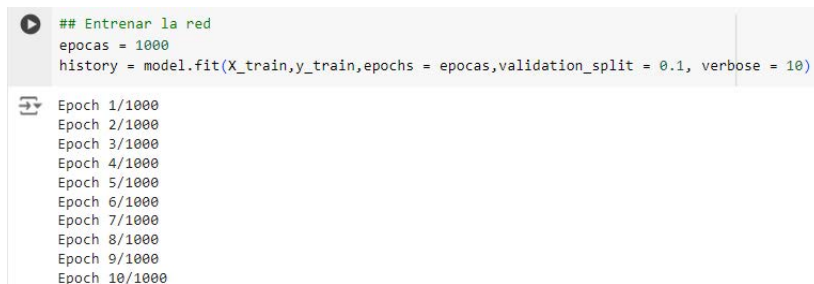
Asimismo, se establece un `validation_split=0.1`, que reserva el 10% de los datos de entrenamiento para evaluación continua

durante el proceso. Esto permite monitorear el rendimiento del modelo en datos no vistos, facilitando la detección temprana de sobreajuste

Por último, el parámetro `verbose=10` configura el sistema para mostrar actualizaciones periódicas cada 10 épocas, proporcionando información continua sobre la evolución del entrenamiento y ayudando a identificar comportamientos relevantes o posibles inconvenientes.

Figura 57

Fragmento de código: configuración del optimizador y las métricas de evaluación



```
## Entrenar la red
epocas = 1000
history = model.fit(X_train,y_train,epochs = epocas,validation_split = 0.1, verbose = 10)
```

Epoch 1/1000
Epoch 2/1000
Epoch 3/1000
Epoch 4/1000
Epoch 5/1000
Epoch 6/1000
Epoch 7/1000
Epoch 8/1000
Epoch 9/1000
Epoch 10/1000

Evaluación del modelo con datos de prueba

Una vez entrenado el modelo, es fundamental verificar su capacidad para generalizar correctamente a nuevos datos, es decir, su desempeño en situaciones no vistas durante el entrenamiento. Para ello, se evalúa su rendimiento sobre un conjunto de prueba independiente.

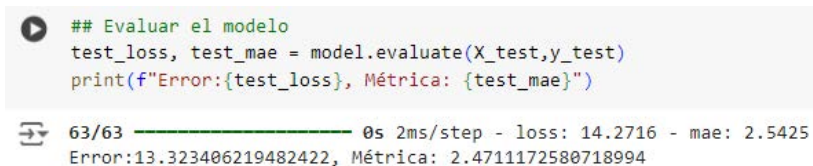
La Figura 58 muestra el uso de la función `model.evaluate`, que permite medir objetivamente el comportamiento del modelo aplicando métricas de error sobre los datos de prueba. Esta función calcula principalmente dos indicadores:

- Pérdida (*loss*): corresponde al error cuadrático medio (MSE), que cuantifica la distancia promedio al cuadrado entre las predicciones del modelo y los valores reales. Un valor más bajo indica un mejor ajuste del modelo.
- error absoluto medio (MAE): representa la media de las diferencias absolutas entre las predicciones y los valores reales, expresado en las mismas unidades del problema, lo que permite una interpretación directa del error promedio.

Estos dos indicadores ofrecen una evaluación clara y completa sobre la precisión y capacidad de generalización del modelo ante datos que no ha procesado previamente, ayudando a identificar posibles sobreajustes o limitaciones en su entrenamiento.

Figura 58

Fragmento de código: evaluación del modelo con datos de prueba



```

## Evaluar el modelo
test_loss, test_mae = model.evaluate(X_test,y_test)
print(f"Error:{test_loss}, Métrica: {test_mae}")

```

63/63 ————— 0s 2ms/step - loss: 14.2716 - mae: 2.5425
Error:13.323406219482422, Métrica: 2.4711172580718994

Resultados de evaluación del modelo en datos no vistos

Después del entrenamiento, es indispensable comprobar cómo se desempeña el modelo con datos completamente nuevos. Esta validación garantiza que el modelo no solo funcione bien con los datos de entrenamiento, sino que también pueda generalizar correctamente.

La Figura 59 ilustra nuevamente el uso de la función `model.evaluate`, esta vez enfocada en analizar los resultados obtenidos al aplicar el modelo sobre el conjunto de prueba. Al ejecutarse, esta función devuelve dos resultados fundamentales:

- Error o pérdida: representa qué tan bien se ajustan las predicciones del modelo a los valores reales del conjunto de prueba. Generalmente, corresponde al error cuadrático medio (MSE).
- Error absoluto medio (MAE): indica la diferencia promedio entre las predicciones del modelo y los valores reales, proporcionando una medida intuitiva de precisión.

Ambas métricas ofrecen una visión clara del rendimiento del modelo al enfrentarse a datos no utilizados durante el entrenamiento, lo que permite evaluar su efectividad en escenarios reales.

Figura 59

Fragmento de código: resultados de evaluación del modelo en datos no vistos

```
## Predicciones
predicciones = model.predict(X_test)
comparacion = pd.DataFrame(predicciones, columns = ["Predicciones"])
comparacion["Reales"] = y_test.values
comparacion
```

63/63 ————— 0s 2ms/step

	Predicciones	Reales
0	336.204865	331.642932
1	334.244629	324.859727
2	216.256821	216.104975
3	171.386948	171.732504
4	169.021378	169.055259
...	...	...
1995	135.422989	135.420480
1996	154.011978	153.647337
1997	224.910172	222.625089
1998	190.109329	189.905165
1999	215.622879	214.235450

2000 rows x 2 columns

Conjuntos de prueba: entradas y salidas para la evaluación del modelo

Una parte esencial del proceso de evaluación de modelos predictivos consiste en separar los datos que se utilizarán exclusivamente para validar su rendimiento. Esto permite obtener una estimación objetiva de su capacidad para generalizar.

La Figura 60 muestra cómo se estructuran los conjuntos de prueba que se utilizan en esta etapa. Por un lado, el conjunto X_{test} contiene las características o variables independientes, es decir, las entradas del modelo durante la prueba. Estas incluyen: la demanda (D), el costo de pedido (K), el costo de mantenimiento (h) y el tiempo de entrega (L). Dado que estos datos no fueron utilizados en el entrenamiento, su análisis permite comprobar si el modelo puede predecir de forma efectiva en nuevos escenarios.

Por otro lado, el conjunto y_{test} representa las etiquetas reales que el modelo intenta predecir, es decir, los valores verdaderos de la cantidad a pedir. Comparar estas etiquetas con las predicciones generadas por el modelo es fundamental para evaluar su precisión y efectividad.

Esta separación entre entradas (X_{test}) y salidas reales (y_{test}) asegura una evaluación rigurosa y representa una práctica estándar en el desarrollo de modelos predictivos aplicados a la gestión de inventarios.

Figura 60

Fragmento de código: conjuntos de prueba: entradas y salidas para la evaluación del modelo

 `y_test`



Cantidad a pedir	
6252	331.642932
4684	324.859727
1731	216.104975
4742	171.732504
4521	169.055259
...	...
6412	135.420480
8285	153.647337
7853	222.625089
1095	189.905165
6929	214.235450

2000 rows x 1 columns

dtype: float64

 `[ ] X_test`

 `array([[1.39358173, 1.29584255, -1.23259334, 1.50411605],
 [0.49003643, 0.57153546, -1.52040476, -0.05032503],
 [1.00341444, 1.72820186, 0.71398265, 1.11550578],
 ...,
 [0.48319139, 1.48015747, 0.14642603, -0.4389353],
 [0.64062731, -0.32025249, 0.24244234, -0.4389353],
 [1.19507557, -0.33014776, -0.16641005, 0.72689551]])`

Predicción de la cantidad a pedir con nuevos datos

Una vez que el modelo ha sido entrenado y evaluado, el siguiente paso lógico es utilizarlo para realizar predicciones con nuevos datos. Esto permite poner en práctica el modelo en contextos reales de toma de decisiones.

La Figura 61 muestra un fragmento de código en el que se realiza una predicción utilizando la red neuronal entrenada. El proceso inicia con la creación de un arreglo de NumPy (`np.array`) que contiene un nuevo conjunto de valores para las variables de entrada: demanda (D) = 784, costo de pedido (K) = 68.13, costo de mantenimiento (h) = 0.27 y tiempo de entrega (L) = 1.

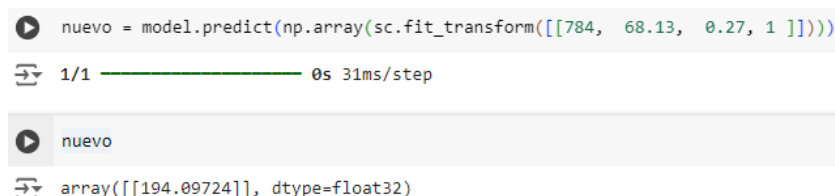
Antes de realizar la predicción, estos valores deben ser escalados para mantener la misma distribución y escala que los datos usados en el entrenamiento. Para ello, se aplica la función `sc.fit_transform` del objeto `StandardScaler`. Este paso es crucial, ya que garantiza que los nuevos datos sean compatibles con el modelo entrenado, que fue calibrado con datos estandarizados.

A continuación, se utiliza la función `model.predict` para generar la predicción de la cantidad óptima a pedir a partir de los datos escalados. El valor predicho se almacena en la variable `nuevo`.

Finalmente, el resultado se imprime en consola, dando como salida un valor aproximado de 19409724, que representa la cantidad sugerida a solicitar según las condiciones de entrada especificadas.

Figura 61

Fragmento de código: predicción de la cantidad a pedir con nuevos datos



```
nuevo = model.predict(np.array(sc.fit_transform([[784, 68.13, 0.27, 1 ]]]))
```

1/1 ————— 0s 31ms/step

```
nuevo
```

```
array([[194.09724]], dtype=float32)
```

Redes neuronales por clasificación

Vidal Rodríguez y Herrera Acosta (2005) abordan la aplicación de redes neuronales artificiales para tareas de clasificación en la gestión de inventarios con política (S-s). El modelo propuesto clasifica los estados del inventario y determina automáticamente si se debe realizar un pedido o no, basándose en variables como la demanda estocástica y el nivel actual de existencias. La investigación compara el desempeño del modelo neuronal con otros enfoques tradicionales, mostrando que las redes neuronales ofrecen mejoras en precisión y adaptabilidad frente a variaciones en el comportamiento de la demanda.

Se decidió también experimentar con redes neuronales para tareas de clasificación, con el propósito de evaluar las posibles mejoras que este enfoque podría aportar. En este caso, la clasificación se aplicó exclusivamente al modelo probabilístico con política S-s, por lo que la explicación siguiente se centra específicamente en este modelo y en el código asociado. En la Figura 62, el código inicia con la importación de diversas bibliotecas fundamentales para el desarrollo del proyecto:

- NumPy, utilizada para realizar operaciones matemáticas y manipular matrices de datos.

- matplotlib.pyplot, que permite la generación de gráficos y la visualización de datos para facilitar el análisis.
- TensorFlow, la biblioteca principal empleada para la construcción, entrenamiento y evaluación del modelo de red neuronal.
- Pandas, que se utiliza para la manipulación y análisis de datos, específicamente para cargar la información desde un archivo Excel.

Estas bibliotecas conforman las herramientas básicas necesarias para la manipulación de datos, la visualización gráfica y el desarrollo del modelo de clasificación basado en redes neuronales.

Figura 62

Fragmento de código: importación de diversas bibliotecas fundamentales para el desarrollo del proyecto

```
[ ] import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
import pandas as pd
```

Carga y procesamiento de datos para modelos de inventario

En la gestión de inventarios, contar con datos precisos y bien estructurados es fundamental para desarrollar modelos que permitan optimizar costos y niveles de *stock*. La correcta importación y procesamiento de la información facilita el análisis y la aplicación de técnicas matemáticas para la toma de decisiones.

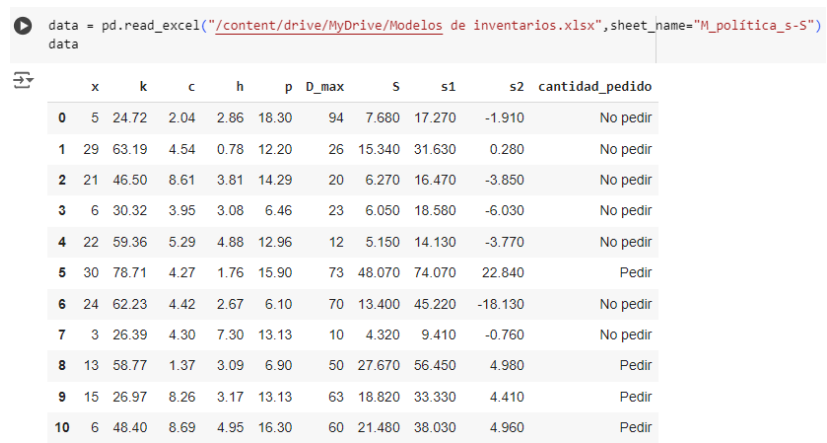
En la Figura 63 se presenta el fragmento de código utilizado para la carga y procesamiento de datos de inventario mediante la biblioteca Pandas de Python. Este código importa un archivo Excel con una estructura tabular que contiene variables clave para los modelos, tales como:

- **x**: identificador del producto,
- **k**: costo de ordenar,
- **c**: costo unitario,
- **h**: costo de mantenimiento,
- **p**: costo de penalización,
- **D_max**: demanda máxima,
- **S**: nivel máximo de inventario,
- **s1** y **s2**: puntos de reorden,
- y la **cantidad óptima de pedido**.

Estos parámetros servirán como insumos fundamentales para el desarrollo y aplicación de los modelos de gestión de inventarios explicados en los capítulos posteriores del documento.

Figura 63

Fragmento de código: carga y procesamiento de datos de inventario mediante la biblioteca pandas de Python

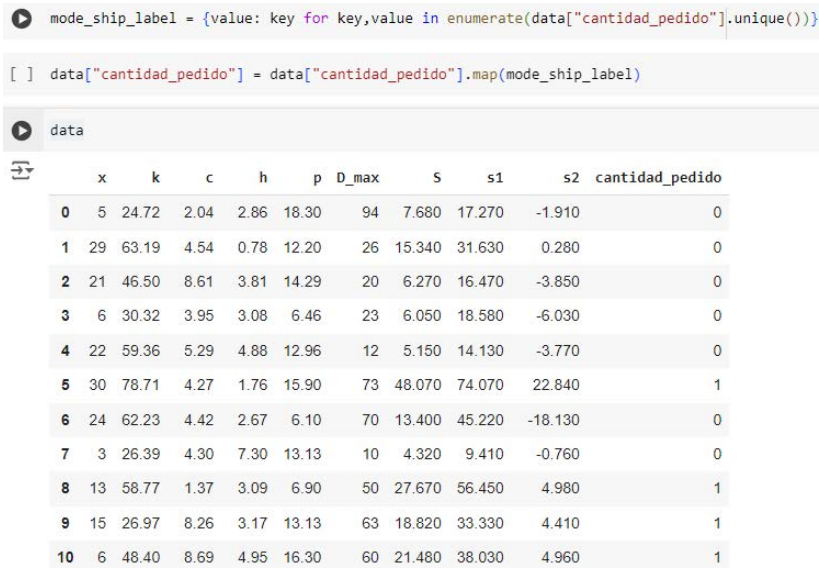


Preparación de datos para modelos predictivos en gestión de inventarios

El desarrollo de modelos predictivos para la gestión de inventarios se basa en la adecuada transformación y preparación de los datos. Para que las técnicas de aprendizaje automático, como las redes neuronales, funcionen correctamente, es necesario convertir las variables categóricas en formatos numéricos y estructurar apropiadamente las matrices de características. En la figura 64 se muestra el proceso de codificación binaria aplicado a la variable categórica «cantidad pedido» del conjunto de datos. Este procedimiento convierte las categorías en valores numéricos binarios, asignando el valor 1 a la categoría “pedir” y 0 a la categoría "no pedir". Esta transformación es crucial para que la red neuronal pueda interpretar correctamente la variable objetivo durante la fase de clasificación, facilitando un entrenamiento efectivo del modelo predictivo.

Figura 64

Fragmento de código: proceso de codificación binaria aplicado a la variable categórica “cantidad_pedido”



Construcción de la matriz de características para entrenamiento de la red neuronal

Para entrenar una red neuronal en problemas de gestión de inventarios, es fundamental organizar y estructurar correctamente los datos de entrada. Esto implica seleccionar las variables relevantes y disponerlas en una matriz que represente cada instancia del problema con sus atributos correspondientes.

En la Figura 65 se presenta el código que construye la matriz de características de entrada, denominada *X*, a partir de diversas variables extraídas del dataframe *data*. Entre estas variables se incluyen identificadores (*x*), costos de ordenamiento

(k), costos unitarios (c), y otros parámetros críticos para el modelo de inventarios.

Además, se realiza una transposición de la matriz (X.T), para asegurar que la estructura de datos sea compatible con el formato esperado por el algoritmo de la red neuronal, donde cada fila corresponde a un vector de atributos para una instancia específica. Esta organización es clave para que el modelo pueda aprender correctamente las relaciones entre variables y realizar predicciones precisas.

Figura 65

Fragmento de código: código que construye la matriz de características de entrada

```
X = np.asarray([data["x"],
                 data["k"],
                 data["c"],
                 data["h"],
                 data["p"],
                 data["D_max"],
                 data["S"],
                 data["s1"],
                 data["s2"]])

[ ] X = X.T

[ ] X

array([[ 5.0000e+00,  2.4720e+01,  2.0400e+00,  2.8600e+00,  1.8300e+01,
         9.4000e+01,  7.6800e+00,  1.7270e+01, -1.9100e+00],
       [ 2.9000e+01,  6.3190e+01,  4.5400e+00,  7.8000e-01,  1.2200e+01,
         2.6000e+01,  1.5340e+01,  3.1630e+01,  2.8000e-01],
       [ 2.1000e+01,  4.6500e+01,  8.6100e+00,  3.8100e+00,  1.4290e+01,
         2.0000e+01,  6.2700e+00,  1.6470e+01, -3.8500e+00],
       [ 6.0000e+00,  3.0320e+01,  3.9500e+00,  3.0800e+00,  6.4600e+00,
         2.3000e+01,  6.0500e+00,  1.8580e+01, -6.0300e+00],
       [ 2.2000e+01,  5.9360e+01,  5.2900e+00,  4.8800e+00,  1.2960e+01,
         1.2000e+01,  5.1500e+00,  1.4130e+01, -3.7700e+00],
       [ 3.0000e+01,  7.8710e+01,  4.2700e+00,  1.7600e+00,  1.5900e+01,
         7.3000e+01,  4.8070e+01,  7.4070e+01,  2.2840e+01],
       [ 2.4000e+01,  6.2230e+01,  4.4200e+00,  2.6700e+00,  6.1000e+00,
         7.0000e+01,  1.3400e+01,  4.5220e+01, -1.8130e+01],
       [ 3.0000e+00,  2.6390e+01,  4.3000e+00,  7.3000e+00,  1.3130e+01,
         1.0000e+01,  4.3200e+00,  9.4100e+00, -7.6000e-01],
       [ 1.3000e+01,  5.8770e+01,  1.3700e+00,  3.0900e+00,  6.9000e+00,
         5.0000e+01,  2.7670e+01,  5.6450e+01,  4.9800e+00],
       [ 1.5000e+01,  2.6970e+01,  8.2600e+00,  3.1700e+00,  1.3130e+01,
         6.3000e+01,  1.8820e+01,  3.3330e+01,  4.4100e+00],
```

Definición y preparación del vector objetivo para la clasificación binaria

En el desarrollo de modelos predictivos para la gestión de inventarios, es fundamental definir claramente la variable objetivo que guiará el proceso de clasificación. Esta variable debe reflejar de forma precisa la acción recomendada sobre la reposición de inventarios, permitiendo que el modelo aprenda a distinguir cuándo realizar un pedido y cuándo no. La codificación binaria de esta variable facilita el procesamiento por parte de las redes neuronales y mejora la eficacia del entrenamiento.

En la Figura 66 se presenta el vector y , que corresponde a la columna "cantidad_pedido" extraído del conjunto de datos. Esta columna indica la acción recomendada respecto a la reposición de inventario, codificada de manera binaria: un valor de 1 significa que se debe "pedir" el producto, mientras que un valor de 0 señala que no es necesario realizar un pedido en ese momento.

Esta codificación binaria es esencial para que el modelo pueda clasificar cada instancia en una de las dos categorías, optimizando así las decisiones de pedido basadas en los patrones aprendidos de datos históricos. La correcta definición y extracción de esta variable objetivo es un paso clave para garantizar un desempeño óptimo y una mayor precisión en el modelo predictivo.

Figura 66

Fragmento de código: vector y , que corresponde a la columna “cantidad_pedido” extraído del conjunto de datos.

```
y = np.asarray(data["cantidad_pedido"])

[ ] y

array([0, 0, 0, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1,
       0, 0, 0, 0, 1, 1])
```

División de datos para entrenamiento y evaluación del modelo predictivo

Un paso crucial en el desarrollo de modelos de aprendizaje automático es la partición adecuada del conjunto de datos en subconjuntos para entrenamiento y prueba. Esta división permite entrenar el modelo con una parte de los datos y validar su desempeño con datos no vistos, garantizando así la generalización y robustez del modelo. La correcta separación y verificación de estos conjuntos es fundamental para evitar sobreajustes y evaluar objetivamente la capacidad predictiva.

En la Figura 67 se muestra la aplicación de la función `train_test_split` de la biblioteca `sklearn`, que facilita esta partición de manera sencilla y efectiva. De forma predeterminada, esta función asigna el 75 % de los datos para entrenamiento y el 25 % restante para pruebas, aunque estos porcentajes pueden ajustarse según las necesidades del análisis.

El resultado de esta función son cuatro conjuntos: X_{train} y y_{train} , que contienen las características y etiquetas, respectivamente, utilizadas para entrenar la red neuronal; y X_{test} y y_{test} , que corresponden a las características y etiquetas

utilizadas para evaluar el desempeño del modelo una vez finalizado el entrenamiento.

Para verificar las dimensiones de estos conjuntos, se emplea un comando que devuelve una tupla con su tamaño. `X_train.shape` indica las dimensiones del conjunto de entrenamiento, donde la primera dimensión corresponde al número de ejemplos y la segunda al número de características, por ejemplo. `X_test.shape` refleja las dimensiones del conjunto de prueba. Finalmente, `y_train.shape` muestra las dimensiones del vector de etiquetas para el entrenamiento, que suele ser unidimensional, representando el número total de ejemplos de entrenamiento.

Figura 67

Fragmento de código: aplicación de la función `train_test_split` de la biblioteca `sklearn`

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y)

[ ] X_train.shape,X_test.shape, y_train.shape

⇒ ((37, 9), (13, 9), (37,))
```

Importancia de la estandarización en el preprocesamiento de datos

En el desarrollo de modelos predictivos, la escala de las variables de entrada puede afectar significativamente el rendimiento y la convergencia de los algoritmos de aprendizaje automático. Cuando las características tienen rangos muy distintos, algunas pueden dominar el proceso de entrenamiento simplemente por tener valores numéricos mayores, lo que puede sesgar

el modelo. Por ello, la estandarización de datos, que consiste en ajustar las variables para que tengan media cero y desviación estándar uno, es una práctica esencial para mejorar la estabilidad y precisión del entrenamiento.

En la Figura 68 se presenta el fragmento de código donde se importa la clase `StandardScaler` de la biblioteca `Scikit-learn`, herramienta utilizada para transformar los datos conforme a estos criterios. Para ello, se crea una instancia del escalador con `sc = StandardScaler()`, que permite ajustar y transformar los datos de forma consistente.

Figura 68

Fragmento de código: clase `StandardScaler` de la biblioteca `Scikit-learn`

```
▶ ##Escalar los datos

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

El método `fit_transform` se aplica al conjunto de entrenamiento con `X_train = sc.fit_transform(X_train)`. En esta etapa, el escalador calcula la media y desviación estándar del conjunto de entrenamiento y transforma los datos para que cumplan con la estandarización. En cambio, para el conjunto de prueba, es fundamental utilizar el método `transform` en lugar de `fit_transform`, como se muestra en `X_test = sc.transform(X_test)`. Esto asegura que la transformación se realice con los parámetros calculados únicamente sobre el conjunto de entrenamiento,

evitando así la filtración de información y preservando la validez de la evaluación del modelo.

Es importante señalar que usar incorrectamente `fit_transform` en el conjunto de prueba, como ocurre en algunos ejemplos, puede introducir sesgos que afecten negativamente la evaluación del desempeño del modelo.

Diseño de la arquitectura de la red neuronal para clasificación en gestión de inventarios

La construcción de un modelo de red neuronal eficaz es fundamental para lograr predicciones precisas en problemas de clasificación, como la toma de decisiones en la gestión de inventarios. Utilizar herramientas especializadas y definir cuidadosamente la arquitectura de la red permite capturar patrones complejos en los datos y mejorar la capacidad del modelo para generalizar sobre nuevas instancias.

En la Figura 69 se muestra el código para la construcción de una red neuronal empleando las bibliotecas TensorFlow y Keras, dos de las plataformas más utilizadas en el desarrollo de aprendizaje profundo. El proceso comienza con la importación de los módulos necesarios, incluyendo `active_count` del paquete `threading` y TensorFlow importado como `tf`.

Figura 69

Fragmento de código: código para la construcción de una red neuronal empleando las bibliotecas TensorFlow y Keras

```
from threading import active_count
import tensorflow as tf

## input layer 9 neurons
inputlyr = tf.keras.layers.Dense(units = 9)

## hidden layers with ReLU
hdnlyr01 = tf.keras.layers.Dense(units = 16, activation = tf.nn.relu)
hdnlyr02 = tf.keras.layers.Dense(units = 16, activation = tf.nn.relu)

## output layer
outputlyr = tf.keras.layers.Dense(units = 2, activation = tf.nn.softmax)

## Crear la NN

model = tf.keras.Sequential([inputlyr, hdnlyr01, hdnlyr02, outputlyr])
```

- La arquitectura se define mediante un modelo secuencial de Keras, que facilita la adición ordenada de capas. La capa de entrada consta de nueve neuronas, correspondientes a las nueve características independientes que alimentan el modelo, creada con `tf.keras.layers.Dense(units=9)`.
- A continuación, se incorporan dos capas ocultas idénticas, cada una con dieciséis neuronas y la función de activación ReLU (`tf.nn.relu`). Esta función introduce no linealidad, permitiendo que la red aprenda representaciones complejas y robustas de los datos. Estas capas se definen con `tf.keras.layers.Dense(units=16, activation=tf.nn.relu)`.
- Finalmente, la capa de salida está compuesta por dos neuronas con función de activación softmax (`tf.nn.softmax`), que convierte las salidas en probabilidades para cada clase, facilitando la interpretación de resultados en problemas de clasificación binaria.

- Esta estructura arquitectónica permite que la red neuronal aprenda patrones relevantes de los datos y produzca predicciones probabilísticas confiables que apoyan la toma de decisiones eficientes en la gestión de inventarios.

Compilación del modelo de red neuronal para clasificación en gestión de inventarios

Antes de iniciar el proceso de entrenamiento de una red neuronal, es esencial configurar adecuadamente el modelo mediante su compilación. Esta etapa define los elementos clave que guiarán la optimización y evaluación durante el aprendizaje, asegurando que el modelo se ajuste correctamente a los datos y pueda generalizar con precisión.

En la Figura 70 se presenta el código correspondiente a la compilación del modelo de red neuronal. En primer lugar, se especifica el uso del optimizador Adam, un algoritmo altamente eficiente y popular que ajusta los pesos y sesgos de manera adaptativa. Adam combina las ventajas del descenso de gradiente estocástico con momentos de primer y segundo orden, favoreciendo una convergencia rápida y estable del modelo.

La función de pérdida seleccionada es `SparseCategoricalCrossentropy`, adecuada para problemas de clasificación con múltiples clases, como es el caso particular de dos clases en este escenario. Esta función mide la discrepancia entre las predicciones probabilísticas del modelo y las etiquetas reales, orientando la optimización hacia la minimización de esta diferencia.

Finalmente, se incluye como métrica la precisión (*accuracy*), que permite monitorear el rendimiento del modelo durante las fases de entrenamiento y validación, reflejando el

porcentaje de predicciones correctas sobre el total de ejemplos evaluados.

Figura 70

Fragmento de código: código correspondiente a la compilación del modelo de red neuronal

```
▶ model.compile(  
    optimizer = tf.keras.optimizers.Adam(),  
    loss = tf.keras.losses.SparseCategoricalCrossentropy(),  
    metrics = ["accuracy"]  
)
```

Evaluación del rendimiento del modelo de red neuronal en datos de prueba

Una vez finalizado el entrenamiento de un modelo de red neuronal, es crucial evaluar su desempeño con datos que no hayan sido utilizados durante el proceso de aprendizaje. Esto permite medir la capacidad del modelo para generalizar correctamente y ofrecer predicciones fiables en escenarios reales.

En la Figura 71 se muestra el proceso de evaluación del modelo utilizando el conjunto de datos de prueba, compuesto por las variables `X_test` y `y_test`. Para este fin, se emplea el método `model.evaluate()`, que recibe los datos de prueba y calcula dos métricas principales: la pérdida (`test_loss`), que representa el error del modelo, y la precisión (`test_acc`), que indica el porcentaje de predicciones acertadas.

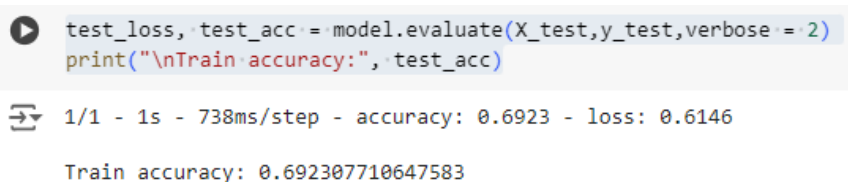
El parámetro `verbose=2` facilita la visualización detallada del progreso durante la evaluación, proporcionando una ejecución transparente y controlada. Al concluir, se imprime en consola el

valor de la precisión de prueba, permitiendo una rápida apreciación del desempeño del modelo ante datos nuevos.

Este paso es fundamental para validar la efectividad del modelo y orientar decisiones sobre ajustes necesarios en la arquitectura o en los parámetros para mejorar su rendimiento en la gestión de inventarios.

Figura 71

Fragmento de código: proceso de evaluación del modelo utilizando el conjunto de datos de prueba



```
test_loss, test_acc = model.evaluate(X_test, y_test, verbose = 2)
print("\nTrain accuracy:", test_acc)
```

1/1 - 1s - 738ms/step - accuracy: 0.6923 - loss: 0.6146

Train accuracy: 0.692307710647583

Predicción con el modelo de red neuronal para nuevos datos de inventario

Una vez entrenado y evaluado el modelo de red neuronal, es posible utilizarlo para predecir la clase de nuevas observaciones, lo que resulta fundamental para la toma de decisiones automatizada en la gestión de inventarios. Este proceso requiere que los datos de entrada estén adecuadamente preparados y escalados para garantizar que el modelo reciba la información en el formato esperado.

En la Figura 72 se presenta el código que realiza la predicción para un nuevo conjunto de datos, con las siguientes características:

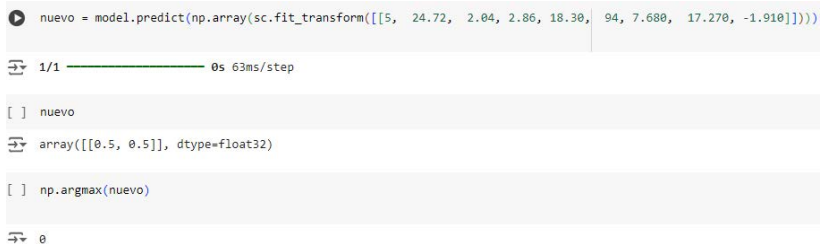
[5, 24.72, 2.04, 2.86, 18.30, 94, 7.680, 17.270, - 1.910].

El vector de características es estandarizado previamente mediante el método `fit_transform` de la instancia `StandardScaler` (`sc`) para mantener la coherencia en la escala respecto al conjunto de entrenamiento. Luego, el método `model.predict()` calcula la probabilidad de pertenencia a cada clase, devolviendo un vector de probabilidades que se almacena en la variable `nuevo`.

Finalmente, se utiliza la función `np.argmax(nuevo)` para identificar el índice del valor máximo en este vector, es decir, la clase con mayor probabilidad asignada por el modelo. Esta clase corresponde a la predicción final para la observación nueva, permitiendo así anticipar si se debe realizar un pedido o no en la gestión de inventarios.

Figura 72

Fragmento de código: código que realiza la predicción para un nuevo conjunto de datos



```
nuevo = model.predict(np.array(sc.fit_transform([[5, 24.72, 2.04, 2.86, 18.30, 94, 7.680, 17.270, -1.910]])))
```

1/1 ————— 0s 63ms/step

```
[ ] nuevo
```

```
array([[0.5, 0.5]], dtype=float32)
```

```
[ ] np.argmax(nuevo)
```

```
0
```

Resumen de la arquitectura del modelo de red neuronal

Para entender en detalle la estructura del modelo de red neuronal construido, es fundamental analizar un resumen que describa sus componentes y características principales. Esta información facilita la interpretación del diseño del modelo y su complejidad, aspectos importantes para su evaluación y posible ajuste.

En la Figura 73 se muestra el uso del comando `model.summary()`, que genera un reporte detallado de la arquitectura de la red neuronal. Este resumen incluye datos específicos de cada capa del modelo, tales como el tipo de capa, la forma de la salida que produce y el número de parámetros entrenables asociados a cada una.

En el reporte se identifican cuatro capas:

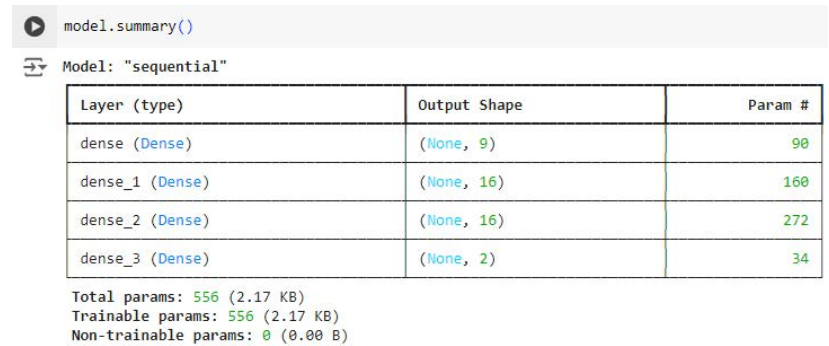
- La capa de entrada, denominada **dense**,
- Dos capas ocultas, llamadas **dense_1** y **dense_2**,
- Y finalmente, la capa de salida, designada como **dense_3**.

Además, el resumen indica que el modelo cuenta con un total de 556 parámetros entrenables, que corresponden a los pesos y sesgos que serán ajustados durante el proceso de entrenamiento para mejorar el desempeño del modelo.

Para un análisis más profundo y detallado del código completo utilizado en este programa, se remite al lector al Anexo 5, donde se incluyen todos los scripts y fragmentos de código correspondientes.

Figura 73

Fragmento de código: uso del comando `model.summary()`



4

CAPÍTULO

Comparación de
desempeño entre
redes neuronales y
árboles de decisión

Motivación

En este capítulo la incorporación de redes neuronales en el programa de gestión de inventarios surge como una evolución natural del proceso de mejora iniciado con el uso de árboles de decisión. Si bien estos últimos permitieron representar de forma lógica la toma de decisiones y obtener resultados satisfactorios, presentaron limitaciones en cuanto a precisión y capacidad de generalización frente a escenarios con alta variabilidad o incertidumbre en la demanda.

Ante ello, se desarrolló una versión optimizada del programa basada en redes neuronales artificiales, con el propósito de incrementar la exactitud de las predicciones y mejorar el desempeño global del sistema. Este enfoque busca evaluar si la integración de redes neuronales aporta una ventaja significativa en la estimación de parámetros clave, como la cantidad económica de pedido, el punto de reorden y el *stock* de seguridad, en comparación con los modelos previos basados en fórmulas matemáticas y árboles de decisión.

El motivo central de esta etapa radica en comprobar el valor añadido de las técnicas de aprendizaje automático dentro de la gestión de inventarios, no solo desde un punto de vista teórico,

sino también práctico y cuantitativo. De este modo, se pretende determinar si las redes neuronales constituyen una herramienta más precisa, eficiente y adaptable para la optimización inteligente de inventarios en entornos empresariales reales.

Resultados de mejora de programa

Una vez implementadas las mejoras en el programa basado en redes neuronales, resulta fundamental realizar una comparación exhaustiva similar a la efectuada previamente con el programa basado en árboles de decisión. Este análisis tiene como objetivo evaluar el grado de mejora que presenta el nuevo modelo respecto a la versión anterior, permitiendo determinar si las optimizaciones realizadas han generado un aumento significativo en la precisión de las predicciones.

Dicha comparación no solo valida la efectividad de las modificaciones introducidas, sino que también facilita una comprensión más profunda de las diferencias en rendimiento y capacidad predictiva entre los enfoques de redes neuronales y árboles de decisión.

Cabe destacar que este programa basado en redes neuronales se presenta únicamente como una propuesta de mejora al programa central del proyecto. Por lo tanto, la comparación de resultados se limitará exclusivamente a los modelos para los cuales se ha desarrollado código específico utilizando redes neuronales. El propósito es determinar si la incorporación de esta técnica representa un avance significativo frente a métodos previos, incluyendo aquellos fundamentados en árboles de decisión y fórmulas matemáticas clásicas.

Este análisis permitirá identificar si la integración de redes neuronales ofrece una ventaja competitiva en términos de

precisión, eficiencia y optimización de las predicciones, lo cual justifica su adopción dentro del marco general del libro.

Comparación de métodos para el cálculo de EOQ y punto de reorden

En el marco del análisis de mejoras mediante técnicas de inteligencia artificial aplicadas a la gestión de inventarios, se ha llevado a cabo una comparación entre tres enfoques distintos para el cálculo de la cantidad económica de pedido (EOQ) y el punto de reorden. Esta evaluación tiene como objetivo determinar el rendimiento relativo de cada método y su aplicabilidad práctica dentro de modelos clásicos de inventario.

Los métodos comparados corresponden a:

1. Un modelo tradicional basado en fórmulas matemáticas clásicas,
2. Un modelo fundamentado en árboles de decisión, y
3. Un modelo construido con redes neuronales artificiales.

El propósito principal de esta comparación es evaluar y contrastar los resultados obtenidos por cada uno de estos métodos, identificando sus fortalezas y limitaciones en cuanto a precisión y consistencia.

Para garantizar la rigurosidad del análisis, se realizaron diez simulaciones independientes para cada modelo, utilizando los mismos datos de entrada en todos los casos. Esta condición de uniformidad asegura que las diferencias observadas en los resultados se deban exclusivamente a las capacidades inherentes de cada enfoque.

En este contexto, la Tabla 17 presenta los resultados de esta comparación, proporcionando una visión clara de las diferencias y beneficios relativos de cada uno de los métodos considerados.

Tabla 17

Resultados comparativos EOQ: Árbol de decisiones vs programa vs. red neuronal

Modelo clásico de cantidad económica de pedido									
D	K	H	L	Árbol de decisiones		Programa		Red neuronal	
				C. de pedido	P. de reorden	C. de pedido	P. de reorden	C. de pedido	P. de reorden
784	68,13	0,27	1	315,67	29,01	629,01	154,99	421,01	175,43
659	49,38	0,07	2	263,63	71,77	964,24	353,76	380,46	176,84
1753	90,26	2,50	10	258,79	201,88	355,78	96,70	383,12	188,58
1316	96,73	2,34	10	283,83	15,56	329,85	295,88	339,75	177,95
935	44,51	2,61	1	188,90	40,39	178,58	42,11	184,86	89,01
863	81,26	2,94	11	210,05	131,27	218,42	101,10	219,39	104,13
1831	57,60	2,40	5	210,38	138,51	296,46	711,50	341,28	185,90
1483	61,12	1,39	8	289,96	194,21	361,14	307,67	391,81	180,71
1133	93,30	2,34	4	286,30	237,97	300,58	23,36	305,39	161,80
1847	16,39	0,36	3	309,04	332,187	410,10	209,74	496,39	198,85

Visualización comparativa del desempeño predictivo de los modelos

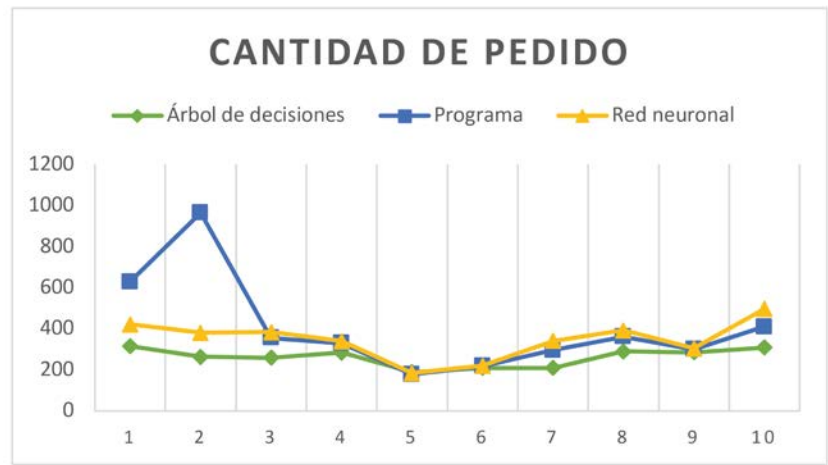
Con el objetivo de facilitar la interpretación de los datos presentados en la Tabla 17 y profundizar en el análisis comparativo de los métodos evaluados, se elaboraron gráficos de líneas que permiten observar de manera clara las desviaciones de cada programa

con respecto a los valores exactos. Estos gráficos constituyen una herramienta visual clave para analizar tanto el comportamiento como la precisión relativa de los tres enfoques estudiados.

La Figura 74 presenta una comparación gráfica de los resultados obtenidos a partir de los tres modelos: el programa tradicional, el modelo basado en árboles de decisión y el modelo de red neuronal. En esta figura, la línea azul representa los datos exactos, los cuales sirven como línea base para evaluar el grado de aproximación de cada enfoque.

Figura 74

EOQ comparación: cantidad de pedido

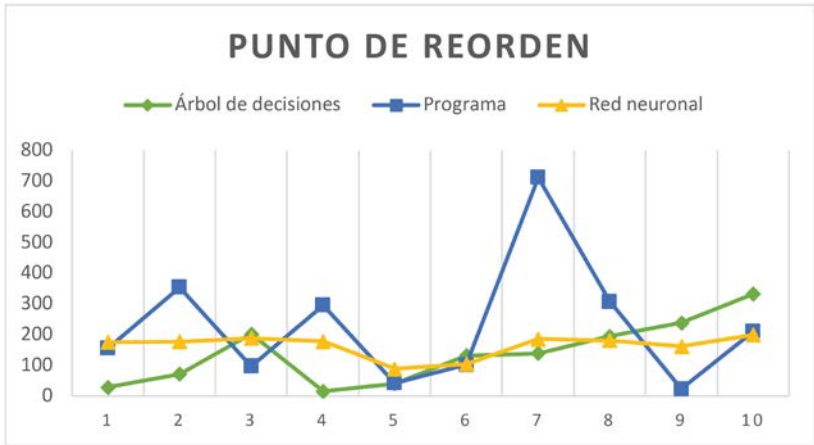


Desde una perspectiva visual, se destaca que el modelo de red neuronal sigue más fielmente la tendencia de los datos exactos, mostrando una aproximación más precisa en comparación con los otros dos métodos. Este comportamiento evidencia una mayor capacidad predictiva del modelo de red neuronal en la estimación de la cantidad económica de pedido (EOQ) y el punto de reorden.

Por su parte, la Figura 75 presenta nuevamente los resultados obtenidos por los tres programas, utilizando la misma referencia de los datos exactos (línea azul). Si bien ninguno de los modelos predictivos logra replicar con total fidelidad los valores reales, se observa que la red neuronal supera al modelo de árbol de decisión, mostrando una mejor capacidad para capturar los patrones subyacentes y ajustarse con mayor precisión a la tendencia general de los datos.

Figura 75

EOQ comparación: punto de reorden



Complementando esta evaluación visual, la Tabla 18 presenta los resultados de una comparación cuantitativa mediante el error cuadrático medio (ECM). Esta métrica permite medir de forma objetiva el grado de error entre las predicciones de cada modelo y los valores exactos, proporcionando así una base numérica para determinar cuál de los enfoques ofrece un mejor desempeño.

Tabla 18

MSE con tres enfoques del modelo clásico de cantidad económica de pedido

Comparación árbol de decisión y programa		Comparación red neuronal y programa	
Diferencia cuadrática (cantidad de pedido)	Diferencia cuadrática (punto de reorden)	Diferencia cuadrática (cantidad de pedido)	Diferencia cuadrática (punto de reorden)
98181,9556	15870,9604	43264	417,7936
490854,3721	79518,3601	340799,0884	31300,6864
9407,0601	11062,8324	747,4756	8441,9344
2117,8404	78579,3024	98,01	13907,4849
106,5024	2,9584	39,4384	2199,61
70,0569	910,2289	0,9409	9,1809
7409,7664	328317,5401	2008,8324	276255,36
5066,5924	12873,1716	940,6489	16118,8416
203,9184	46057,4521	23,1361	19165,6336
10213,1236	14993,26781	7445,9641	118,5921
MSE	62363,11883	58818,60742	39536,75348
RMSE	249,73	242,53	198,84

Comparación de métodos bajo modelos probabilísticos de inventario

para ampliar el alcance del análisis comparativo de técnicas aplicadas a la gestión de inventarios, se consideraron ahora escenarios más cercanos a condiciones reales, caracterizados por la incertidumbre en la demanda y en los tiempos de entrega. En

este contexto, se utilizó el modelo probabilístico de inventario, que permite calcular tanto el *stock* de seguridad como el punto de reorden ajustado en función de la variabilidad y el riesgo.

En este análisis se aplicaron tres enfoques distintos:

1. Un modelo basado en árboles de decisión.
2. Un programa tradicional, sustentado en fórmulas matemáticas determinísticas.
3. Un modelo desarrollado mediante redes neuronales.

El objetivo principal es evaluar la precisión, adaptabilidad y eficacia predictiva de cada método en la estimación de variables clave bajo condiciones de incertidumbre.

La Tabla 19 presenta la comparación de los resultados obtenidos por estos tres métodos en el cálculo del *stock* de seguridad y del punto de reorden probabilístico. Esta comparación permite observar no solo el desempeño de los enfoques en condiciones ideales (como en el modelo clásico), sino también su robustez y capacidad de adaptación frente a escenarios reales más complejos.

Para garantizar la objetividad del análisis, se realizaron diez simulaciones independientes por cada programa, utilizando idénticos datos de entrada. Esta uniformidad en las condiciones asegura que las diferencias observadas en los resultados reflejen exclusivamente las capacidades técnicas de cada método, y no estén influenciadas por la variabilidad de los datos.

Tabla 19

Resultados comparativos modelo probabilizado: árbol de decisiones vs programa vs red neuronal

Modelo probabilizado de cantidad económica de pedido									
L	D	Σ	A	Árbol de decisiones			Programa		
				P. de reorden	S. de seguridad	P. de reorden	S. de seguridad	P. de reorden	S. de seguridad
2	159	35	0,039	537,58	137,77	295,59	72,99	474,54	86,95
6	113	29	0,023	897,22	133,18	819,74	141,74	818,51	112,34
7	168	36	0,049	1315,03	135,27	1333,60	157,60	1330,90	157,81
8	193	31	0,011	1717,23	160,03	1744,82	200,82	1649,71	199,48
9	155	23	0,034	1467,6	141,71	1520,93	125,93	1491,20	126,24
8	186	39	0,015	1680,33	174,24	1727,38	239,38	1757,61	236,69
2	121	27	0,044	313,9	53,4	307,14	65,14	304,95	64,42
1	174	33	0,018	235,53	59,56	243,20	69,20	415,48	66,24
6	142	25	0,027	918,27	133,18	969,99	117,99	986,23	117,95
1	108	38	0,041	150,01	84,9	174,09	66,09	172,71	65,88

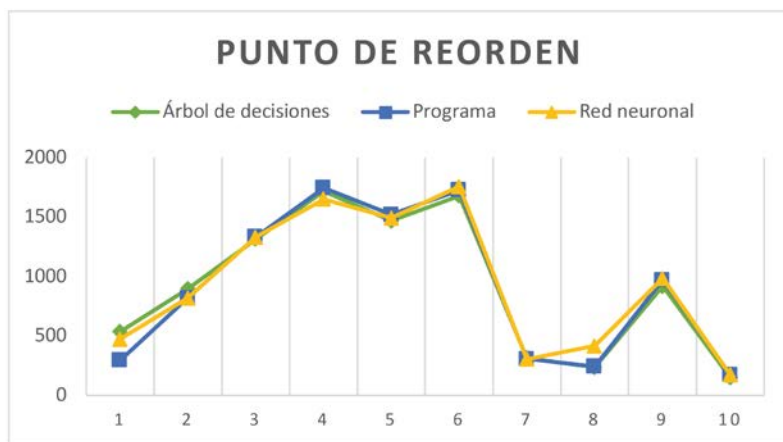
Representación gráfica de resultados en el modelo probabilístico

con el fin de facilitar la interpretación de los resultados presentados en la Tabla 19, se elaboraron gráficos de líneas que permiten observar de forma visual el grado de desviación de cada enfoque respecto a los valores exactos. Estos gráficos ofrecen una representación intuitiva de la precisión relativa de cada método y permiten identificar cuál se aproxima con mayor fidelidad a los valores de referencia establecidos para el modelo probabilístico de inventario.

La Figura 76 presenta los resultados obtenidos mediante los tres enfoques: el programa tradicional, el modelo basado en árboles de decisión y el modelo de red neuronal. En el gráfico, la línea azul representa los datos exactos, que sirven como punto de referencia para evaluar la precisión de cada método.

Figura 76

EOQ probabilizado: comparación punto de reorden



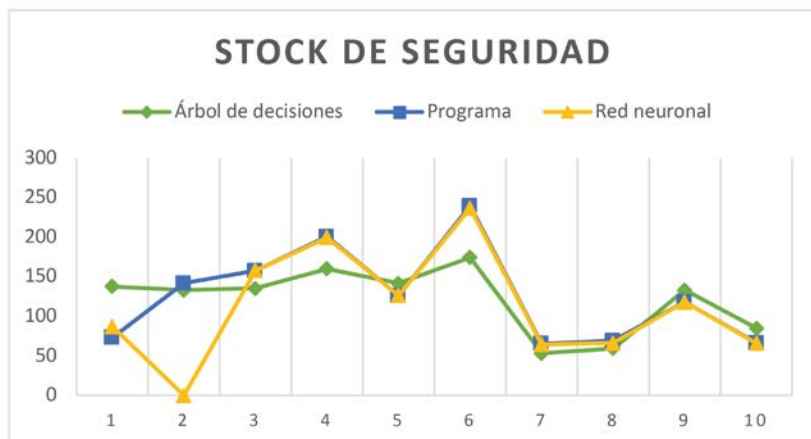
Desde un análisis visual, se observa que tanto el árbol de decisiones como la red neuronal logran seguir con alta precisión la tendencia de los datos exactos, mostrando un comportamiento significativamente más ajustado en comparación con el enfoque tradicional. Esta aproximación indica un desempeño predictivo robusto por parte de ambos modelos en el contexto de un sistema de inventario bajo condiciones de incertidumbre.

La Figura 77 muestra los resultados generados por los tres enfoques mencionados, utilizando como referencia nuevamente la línea azul de los datos exactos. En este caso, el gráfico compara la precisión en la estimación del *stock* de seguridad.

A partir del análisis visual, se aprecia que —salvo en los dos primeros puntos— el modelo de red neuronal mantiene una precisión cercana al 100 %, evidenciando un ajuste altamente preciso en el resto del conjunto de datos. Este comportamiento pone de manifiesto una superioridad significativa de las redes neuronales respecto al árbol de decisiones, particularmente en su capacidad para replicar patrones complejos y adaptarse a la variabilidad inherente de los modelos probabilísticos.

Figura 77

EOQ probabilizado: comparación stock de seguridad



Comparación cuantitativa de desempeño: error cuadrático medio (MSE)

Para complementar esta evaluación visual, la Tabla 20 presenta una comparación cuantitativa de los errores cuadráticos medios (MSE) obtenidos por cada uno de los enfoques analizados. Esta métrica estadística permite medir de forma objetiva la magnitud del error entre los valores predichos por cada modelo y los datos reales, expresando el promedio de las diferencias al cuadrado.

El uso del MSE proporciona una base sólida para determinar cuál de los métodos ofrece un mejor rendimiento predictivo, ya que penaliza con mayor severidad las predicciones más alejadas de los valores reales. De este modo, se refuerza el análisis gráfico previo y se brinda un respaldo numérico a las observaciones sobre la precisión y estabilidad de cada técnica, tanto en el cálculo del punto de reorden como del *stock* de seguridad bajo condiciones de incertidumbre.

Tabla 20

MSE con tres enfoques del modelo probabilizado

Comparación árbol de decisión y programa			Comparación red neuronal y programa	
	Diferencia cuadrática (punto de reorden)	Diferencia cuadrática (stock de seguridad)	Diferencia cuadrática (punto de reorden)	Diferencia cuadrática (stock de seguridad)
	58559,1601	4196,4484	32023,1025	194,8816
	6003,1504	73,2736	1,5129	864,36
	344,8449	498,6289	7,29	0,0441
	761,2081	1663,8241	9045,9121	1,7956
	2844,0889	249,0084	883,8729	0,0961
	2213,7025	4243,2196	913,8529	7,2361
	45,6976	137,8276	4,7961	0,5184
	58,8289	92,9296	29680,3984	8,7616
	2674,9584	230,7361	263,7376	0,0016
	579,8464	353,8161	1,9044	0,0441
<i>MSE</i>	7408,54862	1173,97124	7282,63798	107,77392
<i>RMSE</i>	86,07	34,26	85,35	10,38

Análisis final del desempeño predictivo en modelos de inventario

Tras la aplicación de los tres enfoques propuestos —programa tradicional, árbol de decisión y red neuronal— en modelos de inventario tanto determinísticos como probabilísticos, se procedió a evaluar cuantitativamente el rendimiento de cada uno a partir de los resultados obtenidos en múltiples simulaciones.

En el modelo clásico de cantidad económica de pedido (EOQ), el enfoque basado en redes neuronales evidenció el menor error cuadrático medio (MSE) tanto en la cantidad de pedido como en el punto de reorden, superando al modelo basado en árboles de decisión. Estos resultados reflejan una mejora significativa en la precisión de las predicciones, destacando la eficacia de las redes neuronales como herramienta de optimización para estimaciones en contextos determinísticos.

En cuanto al modelo probabilístico, la comparación presentada en la Tabla 20 también resalta el desempeño superior del modelo de red neuronal frente a los otros enfoques. De manera particular, la red neuronal mostró una mayor precisión en la estimación del *stock* de seguridad, con un MSE sustancialmente menor. Si bien en la estimación del punto de reorden los tres modelos presentan márgenes de error, la red neuronal mantiene una ligera ventaja en consistencia y exactitud.

En conjunto, los resultados permiten concluir que, aunque el modelo basado en árboles de decisión es funcional, presenta limitaciones importantes en cuanto a precisión predictiva, especialmente bajo condiciones de incertidumbre. Por el contrario, el modelo de redes neuronales se posiciona como la alternativa más robusta y adaptable, tanto en escenarios ideales como en contextos logísticos complejos y variables, demostrando un alto potencial de generalización y ajuste a patrones reales.

5 **CAPÍTULO**

Análisis comparativo,
validación estadística,
conclusiones y
recomendaciones

Motivación

En este capítulo se analizó el impacto del programa de gestión de inventarios desarrollado, evaluando su aplicación en entornos empresariales reales y comparando su desempeño con métodos tradicionales utilizados para el cálculo de parámetros logísticos clave.

Para validar estadísticamente la efectividad del programa, se aplicó una prueba de hipótesis *t* de Student, cuyo objetivo fue determinar si existían diferencias significativas entre los resultados obtenidos por el programa y aquellos calculados manualmente. Los resultados permitieron rechazar la hipótesis nula, confirmando que el programa genera resultados diferenciados y, en muchos casos, superiores frente a los enfoques convencionales.

Además, se implementaron procesos de optimización de inventarios, logrando una reducción considerable de los costos de almacenamiento, uno de los objetivos prioritarios del desarrollo. El programa fue también sometido a ajustes de parámetros, lo cual garantizó su funcionalidad en diversos contextos empresariales, incrementando su versatilidad y aplicabilidad.

El análisis visual mediante gráficos comparativos evidenció una mejora sustancial en la eficiencia de la gestión de

inventarios, especialmente gracias a la integración de herramientas de inteligencia artificial como los árboles de decisión y las redes neuronales. Estos resultados refuerzan el valor práctico del programa como una herramienta moderna, adaptable y eficaz para apoyar la toma de decisiones en entornos logísticos complejos.

Validación estadística mediante pruebas de hipótesis

Para evaluar la eficacia del programa de gestión de inventarios desarrollado, se aplicó una prueba de hipótesis *t* de Student, con el objetivo de determinar si existía una diferencia significativa entre dos conjuntos de datos: los valores empíricos obtenidos antes de la implementación del programa y los resultados generados posteriormente tras su aplicación.

El análisis se llevó a cabo utilizando un nivel de significancia de $p=0.05$, criterio ampliamente aceptado en estudios estadísticos para validar diferencias entre grupos. Idealmente, este tipo de análisis requiere al menos 100 observaciones por grupo para garantizar la robustez y representatividad de los resultados. Sin embargo, debido a la limitada disponibilidad de datos proporcionados por las empresas colaboradoras, solo se contaba con 10 observaciones iniciales.

Para superar esta restricción metodológica, se aplicó la técnica de remuestreo *bootstrap*, un método estadístico que permite generar múltiples muestras a partir de una muestra original, mediante selección aleatoria con reemplazo. Esta estrategia permitió ampliar la base de datos a 100 observaciones simuladas, manteniendo la distribución subyacente de los datos originales y habilitando un análisis estadístico válido.

El estudio se centró en el programa desarrollado con el enfoque de árboles de decisión, aplicado al modelo clásico de cantidad económica de pedido (EOQ), específicamente sobre la variable "cantidad a pedir". La elección de este enfoque se justifica en que, si se demuestra una diferencia significativa en este caso —donde el desempeño predictivo es moderado—, es razonable asumir que el modelo de redes neuronales, que previamente ha demostrado una mayor precisión, también arrojaría diferencias significativas y aún más precisas frente a los datos empíricos.

Los resultados del análisis confirmaron la existencia de una diferencia estadísticamente significativa entre los valores estimados por el programa y los datos reales en la variable "cantidad a pedir". Este hallazgo respalda la hipótesis de que el programa desarrollado mejora la precisión de los cálculos logísticos, y refuerza la validez del modelo basado en redes neuronales como una herramienta superior en términos de exactitud y eficiencia para la gestión de inventarios en entornos empresariales reales.

Evaluación de la normalidad en datos empíricos y simulados mediante muestreo bootstrap

Con el propósito de validar estadísticamente la eficacia del programa de gestión de inventarios desarrollado, se recopilaron datos reales de la variable "cantidad a pedir", obtenidos a través de entrevistas a diversas empresas. Estos valores representan estimaciones generadas mediante métodos tradicionales y constituyen la base empírica para la comparación con los resultados del nuevo sistema.

Dado el tamaño reducido de la muestra original, se aplicó la técnica de muestreo *bootstrap*, la cual permite generar múltiples observaciones a partir de una muestra inicial mediante selección aleatoria con reemplazo (Efron y Hastie, 2021). Este procedimiento permitió expandir la base de datos

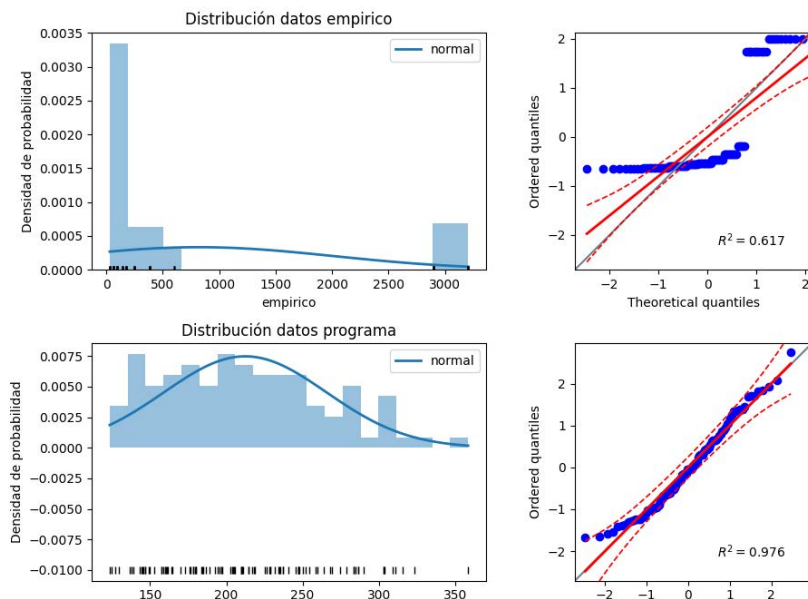
a 100 observaciones simuladas, conservando las características estadísticas de la distribución original. La muestra ampliada facilitó la construcción de una tabla comparativa entre los datos empíricos y los valores generados por el programa, detallada en el Anexo 6, donde también se explica el proceso de remuestreo.

Antes de aplicar la prueba t de Student, se verificó el cumplimiento del supuesto de normalidad para ambas muestras. Para ello, se emplearon tanto gráficos de cuantiles (Q-Q plots) como el cálculo del coeficiente de determinación (R^2), herramientas que permiten evaluar visual y cuantitativamente el ajuste de los datos a una distribución normal. El código utilizado para estos análisis, así como para la prueba t, se presenta en el Anexo 7.

Los resultados obtenidos se muestran en la Figura 76, la cual incluye las salidas del programa que confirman o rechazan los supuestos estadísticos necesarios para proceder con el análisis inferencial. Posteriormente, en la Figura 78, se ilustran los Q-Q plots de ambas muestras, junto con sus respectivos valores de R^2 .

Figura 78

Distribución de los datos simulados a una distribución normal



En el caso de los datos empíricos, se observa una clara desviación respecto a la normalidad. Esto se evidencia en la asimetría de la distribución y en la desviación de los puntos con respecto a la línea de referencia en el gráfico de cuantiles, especialmente en los extremos, lo que sugiere la presencia de colas pesadas o sesgo. Esta observación se ve reforzada por un coeficiente de determinación bajo ($R^2 = 0,617$), lo que indica una escasa correspondencia con una distribución normal.

En contraste, los datos simulados mediante el programa presentan una distribución notablemente más simétrica. En su gráfico de cuantiles, los puntos siguen de forma mucho más uniforme la línea de referencia, y el valor elevado del coeficiente

de determinación ($R^2 = 0,976$) confirma un ajuste satisfactorio a la normalidad.

Esta verificación resulta fundamental, ya que garantiza la validez del uso de la prueba t de Student sobre los datos generados por el programa. De este modo, se asegura la solidez metodológica del análisis comparativo, fortaleciendo la fiabilidad de los resultados obtenidos y el respaldo empírico a las mejoras introducidas por el nuevo sistema de gestión de inventarios.

Aplicación de la prueba shapiro-wilk y t de student para comparación de muestras

Además del análisis visual mediante los gráficos de cuantiles (Q-Q plots), se aplicó la prueba estadística de normalidad de Shapiro-Wilk con el fin de evaluar de manera formal si los datos analizados provienen de una distribución normal. Esta prueba proporciona un valor p, que indica la probabilidad de que los datos observados se ajusten a una distribución normal, bajo la hipótesis nula de normalidad (Razali y Wah, 2011).

Los resultados de esta prueba se presentan en la Tabla 21, donde se detallan los valores p obtenidos tanto para los datos empíricos como para los datos generados por el programa. Estos valores permiten determinar si se acepta o se rechaza la hipótesis nula de normalidad en cada caso, considerando un nivel de significancia de 0.05.

Tabla 21

Valor p normalidad

Test de normal dad Shapiro-Wilk			Normalidad
	Datos empíricos	Datos programas	
valor p	6,848007e-15	0,031024	Falso

Para los datos empíricos, el valor p obtenido fue de $6,85 \times 10^{-15}$, un valor extremadamente bajo y muy inferior al umbral establecido. Esto proporciona evidencia concluyente para rechazar la hipótesis nula, lo que indica que los datos no siguen una distribución normal.

Por su parte, los datos generados por el programa arrojaron un valor p de 0,0310, el cual, si bien es más cercano al umbral de 0,05, también se encuentra por debajo del nivel de significancia, por lo que se concluye igualmente que estos datos no cumplen estrictamente con el supuesto de normalidad.

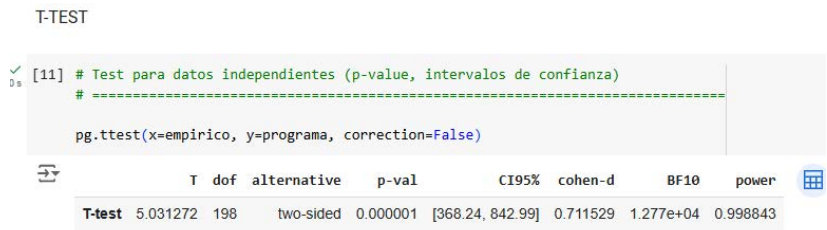
En consecuencia, ambos conjuntos de datos violan el supuesto de normalidad bajo la prueba de Shapiro-Wilk. Sin embargo, considerando que el tamaño de cada muestra es mayor a 30 observaciones, se recurre al teorema central del límite, el cual establece que la distribución de las medias muestrales tiende a ser normal cuando el tamaño de la muestra es suficientemente grande. Esto permite justificar el uso de la prueba t de Student, aún cuando los datos individuales no sigan una distribución normal.

Una vez evaluada la normalidad, se procedió a aplicar la prueba t de Student para muestras independientes, con el objetivo de comparar las cantidades a pedir observadas empíricamente con aquellas calculadas por el modelo de cantidad económica de pedido (EOQ), basado en árboles de decisión.

El resultado de esta prueba, representado en la Figura 79, arrojó un valor p de 0,00001, lo que constituye una evidencia estadísticamente significativa para rechazar la hipótesis nula de igualdad de medias. Esto implica que existe una diferencia significativa entre los valores empíricos y los generados por el modelo EOQ.

Figura 79

Resultados de la prueba t de Student para muestras independientes entre los datos empíricos y los generados por el modelo EOQ basado en árboles de decisión



Además, se calculó el tamaño del efecto mediante el estadístico d de Cohen, obteniendo un valor de 0,711529. Este valor corresponde a un efecto de magnitud alta, lo que refuerza la conclusión de que la diferencia entre ambos conjuntos de datos no solo es estadísticamente significativa, sino también sustancial desde una perspectiva práctica.

En conjunto, estos resultados respaldan la hipótesis de que el programa desarrollado aporta mejoras significativas en los cálculos logísticos, especialmente en la estimación de la cantidad óptima a pedir, contribuyendo así a una gestión de inventarios más eficiente.

Optimización de recursos

Los resultados de la prueba de hipótesis indicaron una diferencia estadísticamente significativa entre los dos conjuntos de datos, lo que condujo al rechazo de la hipótesis nula. Esta diferencia implica que las cantidades calculadas por el modelo EOQ basado en árboles de decisión no coinciden con las cantidades observadas en las prácticas actuales de las empresas.

Sin embargo, esta discrepancia no debe interpretarse de forma negativa. Por el contrario, puede entenderse como una evidencia positiva del potencial del programa para optimizar la gestión de inventarios. El hecho de que el modelo proponga cantidades distintas sugiere que introduce mejoras respecto a los métodos tradicionales, contribuyendo a reducir costos asociados al exceso de inventario, faltantes o ineficiencias en el reabastecimiento.

En este contexto, la validación estadística no solo respalda la solidez matemática del modelo, sino también su aplicabilidad práctica. La capacidad del programa para generar decisiones más eficientes en términos de cantidades a pedir demuestra su valor como herramienta para la optimización de recursos y la mejora del desempeño logístico en entornos empresariales reales.

Optimización de inventarios

Una gestión eficiente de inventarios es un factor clave para garantizar el buen funcionamiento operativo y financiero de cualquier empresa. Como se analizó en los primeros capítulos de este libro, muchas organizaciones aún gestionan sus inventarios de forma empírica, sin el respaldo de herramientas tecnológicas ni modelos sistematizados. Esta práctica conlleva a errores frecuentes como faltantes de productos o excesos de inventario, lo que puede generar costos innecesarios, pérdida de ventas y una disminución en la satisfacción del cliente.

Si bien algunas empresas logran mitigar parcialmente estos efectos gracias a su experiencia acumulada, el desarrollo e implementación de sistemas basados en modelos matemáticos ofrece una alternativa mucho más precisa y eficaz. En este contexto, el programa propuesto en este proyecto tiene como objetivo mejorar sustancialmente la toma de decisiones

relacionadas con el aprovisionamiento, reduciendo los márgenes de error y optimizando recursos.

En la Tabla 22 se resumen los principales problemas enfrentados por las empresas entrevistadas durante el diagnóstico inicial. Los datos muestran con claridad que todas las organizaciones presentan inconvenientes asociados al exceso o escasez de productos, en algunos casos con impactos significativos. Esta evidencia refuerza la necesidad de adoptar herramientas tecnológicas que faciliten una gestión más precisa y estratégica de los inventarios.

Tabla 22

Problemas identificados en las empresas analizadas

Empresa	Problemática
A TODP PEDAL	Sobrantes
Abastos el Nazareno	Faltantes
Águila Bike	Sobrantes y Faltantes
Comercial Enríquez	Sobrantes
Ferroalvid	Sobrantes y Faltantes
Mega Ofertas	Sobrantes
Movitech	Sobrantes
SALOMON	Sobrantes
Technet	Sobrantes
Tejidos Olarte	Sobrantes

Además, los datos recopilados permitieron realizar una prueba de hipótesis para comparar el desempeño entre la gestión empírica tradicional y los resultados obtenidos mediante el programa desarrollado. Los resultados mostraron diferencias estadísticamente significativas, lo cual indica que el sistema propuesto tiene un impacto positivo al acercarse a una gestión óptima de inventarios. Esto se debe a que el programa utiliza

técnicas sistemáticas para calcular de manera eficiente tanto la cantidad a pedir como el punto de reorden.

Estos parámetros —que son calculados automáticamente por el sistema— permiten a las empresas reducir errores, mejorar la eficiencia operativa y disminuir costos en todo el proceso de aprovisionamiento. Además, al establecer claramente cuándo y cuánto pedir, se reduce de forma considerable el tiempo que el personal dedica a la planificación de pedidos, especialmente en contextos donde se gestionan múltiples proveedores. De este modo, se eliminan tareas repetitivas y se optimizan los recursos humanos y materiales.

Para ejemplificar el impacto práctico del programa, se seleccionó uno de los casos analizados. En la Tabla 23 se presentan los datos de compras y ventas de la empresa Águila Bike, especializada en la comercialización de productos para bicicletas y motocicletas. Estos datos sirvieron como base para simular la aplicación del modelo en un entorno real.

Durante el trabajo de campo, se recopilaron detalles como las cantidades solicitadas por periodo, el tiempo promedio de rotación del inventario, los artículos más demandados y los precios unitarios. Esta información permitió demostrar cómo, al aplicar el modelo propuesto, se puede lograr una gestión mucho más eficiente, ajustada a las necesidades reales del negocio y alineada con una política de inventarios basada en datos concretos.

Tabla 23*Información empresa Águila Bike*

Productos	Compra		Venta		Precio
	Cantidad mensual	T. promedio de reabastecimiento	Cantidad mensual	Periodo entre compra y venta	
Llantas de bicicleta	1000	Cada mes	1000	1 mes y medio	15
Tubos de bicicleta	1000	Cade mes	1000	1 mes y medio	3
Equipamiento de ciclismo	100	Dos veces al año	100	1 año	25
Repuestos y Accesorios de moto	1000	Cada semana	1000	2 meses	5
Equipamiento de moto	100	Cuatro veces en el año	100	6 meses	50

Parametrización del modelo EOQ

En el análisis de las empresas participantes en este estudio, se identificó un patrón común: todas utilizan el modelo clásico de inventarios EOQ (cantidad económica de pedido) como base para sus decisiones de aprovisionamiento. Ninguna de ellas ha implementado sistemas más avanzados de gestión de inventarios, lo que evidencia la persistencia del modelo EOQ como estándar operativo en este sector.

A partir de esta observación, se utilizaron los datos recopilados durante las entrevistas para calcular los parámetros específicos del modelo EOQ en relación con los productos que maneja cada empresa. El propósito de este ejercicio fue doble: por un lado, evaluar el nivel actual de eficiencia en la gestión de inventarios y, por otro, identificar oportunidades de mejora

mediante la implementación del programa desarrollado en este proyecto.

Este análisis permite comparar el comportamiento de los inventarios —particularmente los niveles de sobrantes y los costos asociados— antes y después de aplicar el sistema propuesto. Así, se obtiene una base sólida para argumentar la pertinencia de sustituir una gestión empírica por un enfoque sistematizado y optimizado.

La Tabla 24 muestra los resultados de esta parametrización aplicada a la empresa Águila Bike, cuyas cifras de compras y ventas ya se presentaron previamente (ver Tabla 23). En ella se detallan los valores obtenidos para cada uno de los productos analizados, calculados a partir de la información recopilada directamente durante las entrevistas.

Tabla 24

Cálculo de cantidad óptima de pedido y punto de reorden

Productos	D	K	H	L	Cantidad a pedir	Punto de reorden
Llantas de bicicleta	666,66	70	0,2	0,033	683	22
Tubos de bicicleta	666,66	70	0,2	0,033	683	22
Equipamiento de ciclismo	100	12	0,2	0,0028	107	10
Repuestos y accesorios de moto	500	25	0,2	0,167	547	83
Equipamiento de moto	200	14	0,2	0,167	223	33

Por ejemplo, para el caso de las llantas de bicicleta, se reportó una adquisición de 1000 unidades cada mes y medio.

Utilizando una conversión proporcional simple, se estimó una demanda mensual de aproximadamente 666,66 unidades. Este procedimiento fue aplicado de forma sistemática para todos los productos considerados en el estudio.

Los costos de pedido, costos de almacenamiento y tiempos de entrega fueron también estimados a partir de la información proporcionada por la empresa, ajustándose a las unidades de tiempo específicas utilizadas para cada producto. Así, mientras que la demanda de llantas se calculó en función de meses, los datos relativos a equipos de ciclismo fueron expresados en años, de acuerdo con la frecuencia y volumen de aprovisionamiento informado.

La parametrización realizada constituye un insumo fundamental para evaluar la viabilidad de implementar modelos más sofisticados, que no solo mantengan la lógica del EOQ, sino que además la integren dentro de un sistema automatizado capaz de generar recomendaciones precisas y reducir los costos logísticos y operativos.

Optimización significativa

Una vez calculados los parámetros del modelo EOQ tradicional, se procedió a aplicar una versión mejorada del mismo utilizando un programa basado en redes neuronales. Este enfoque híbrido incorpora inteligencia artificial para ajustar los valores de cantidad óptima de pedido y punto de reorden en función del comportamiento histórico de la demanda. El objetivo fue realizar una comparación directa entre la gestión actual de inventarios y el desempeño proyectado por el sistema propuesto.

La finalidad de este análisis fue evaluar el grado de optimización que puede lograrse al reemplazar el método empírico o tradicional por un enfoque sistemático y adaptativo.

Para ello, se tomaron los datos reales reportados por la empresa y se contrastaron con los resultados generados por el programa, específicamente en cuanto a niveles de sobrantes y costos asociados.

La Tabla 25 presenta esta comparación. En ella se detallan, para cada producto, los niveles de excedente real y los proyectados por el modelo, así como el impacto económico que representan ambos escenarios.

- La columna e1 muestra el excedente real de unidades en inventario, calculado como la diferencia entre la cantidad comprada y la cantidad vendida, expresado por mes o por año, según corresponda. Estos datos revelan que la empresa mantiene niveles significativos de sobrantes en la mayoría de sus productos.
- Por su parte, la columna e2 indica los excedentes proyectados al implementar la cantidad óptima de pedido sugerida por el programa. Estos valores se obtienen al restar la demanda de cada producto a la cantidad recomendada por el modelo. Se observa una reducción significativa en los sobrantes, lo que implica una mejora directa en la eficiencia del inventario.

Por ejemplo, en el caso de las llantas para bicicleta, el excedente actual es de 333,34 unidades, mientras que con el modelo este se reduciría a 16,34 unidades. Esta diferencia evidencia una mejora sustancial en la precisión del aprovisionamiento, disminuyendo los recursos ociosos y liberando espacio en almacén.

- Finalmente, la columna Costos refleja el impacto financiero de estos excedentes. En la subcolumna Real, se presenta el costo mensual asociado al exceso de unidades en el inventario actual. Por ejemplo, el excedente de llantas implica un gasto de 5000.10

dólares mensuales. En contraste, al aplicar el modelo, ese costo se reduce a 245,10 dólares, según lo reportado en la subcolumna Programa.

Este análisis confirma que el uso de modelos de optimización basados en inteligencia artificial puede traducirse en ahorros significativos, tanto en términos económicos como operativos. La reducción de sobrantes no solo disminuye los costos de almacenamiento, sino que también mejora la capacidad de respuesta y la toma de decisiones en la gestión de inventarios.

Tabla 25

Optimización: comparación real vs. programa

Productos	Excedentes		Costos	
	e1	e2		Programa
Llantas de bicicleta	333,34	16,34	5000,1	245,1
Tubos de bicicleta	333,34	16,34	1000,02	49,02
Equipamiento de ciclismo	100	7	2500	175
Repuestos y accesorios de moto	3500	47	17500	235
Equipamiento de moto	200	23	10000	1150
			36000,12	1854,12
			Optimización	94,84

Impacto de la optimización en excedentes y costos

Una vez aplicada la optimización mediante el programa basado en redes neuronales, se procedió a comparar gráficamente los resultados con los datos reales proporcionados por la empresa. Este análisis visual permite evaluar con mayor claridad el efecto de la herramienta sobre la reducción de excedentes y la eficiencia general del inventario.

Las Figuras 80 y 81 ilustran la diferencia entre los valores reales y los proyectados tras la optimización. En ambas representaciones se aprecia un patrón consistente: todos los productos analizados experimentan una reducción notable en sus excedentes, lo que refleja una mejora tangible en la gestión de inventarios.

Figura 80

Comparación costos real vs. programa



Figura 81

Comparaciones excedentes real vs programa



Este hallazgo no solo confirma la efectividad del modelo propuesto, sino que también evidencia su impacto directo en la reducción de costos y en la optimización del uso de recursos. Al disminuir el volumen de inventario innecesario, se logra una utilización más eficiente del espacio, una reducción en los costos de almacenamiento y un menor capital inmovilizado.

Para cuantificar este beneficio, se analizaron los costos asociados al exceso de inventario antes y después de aplicar el programa. El resultado fue el siguiente:

Costo actual (sin optimización): 36000,12 dólares

Costo proyectado (con programa): 1854,12 dólares

Con estos valores, se calculó el porcentaje de optimización alcanzado utilizando la siguiente fórmula:

$$\text{Optimización} = \left(\frac{\text{Costo antes} - \text{Costo ahora}}{\text{Costo antes}} \right) \times 100 \quad (41)$$

Al aplicar la fórmula, se obtuvo un porcentaje de optimización superior al 94 % en la gestión de inventarios. Esta cifra representa una mejora significativa y tangible que puede lograrse mediante la implementación del programa desarrollado, sin necesidad de cambios estructurales en la operación de la empresa.

Además, este nivel de mejora se encuentra dentro del margen de error estimado del 5 % para el programa, lo que refuerza la fiabilidad del modelo y su potencial de aplicación práctica en diferentes contextos empresariales.

Evaluación del desempeño del modelo EOQ basado en redes neuronales

Para asegurar la confiabilidad del modelo propuesto, se evaluó su desempeño durante las fases de entrenamiento y validación. En este proceso, se empleó una métrica de error que permite medir la capacidad del modelo para generar predicciones precisas respecto a la cantidad óptima de pedido y el punto de reorden.

La Figura 82 presenta la evolución de dicha métrica a lo largo del proceso de entrenamiento. Esta gráfica permite visualizar cómo el modelo, al ser expuesto a los datos históricos, ajusta progresivamente sus parámetros para minimizar el error entre las predicciones y los valores reales.

La tendencia descendente de la curva de error observada en la figura indica que el modelo mejora su desempeño con cada iteración, logrando una mayor precisión en sus estimaciones

conforme avanza el aprendizaje. Esta evolución valida la capacidad de generalización del modelo y refuerza su utilidad como herramienta efectiva para la optimización de inventarios en entornos empresariales reales.

Figura 82

Fragmento de código: métrica de error que permite medir la capacidad del modelo para generar predicciones precisas respecto a la cantidad óptima de pedido y el punto de reorde

```
## Evaluar el modelo
test_loss, test_mae = model.evaluate(X_test,y_test)
print(f"Error:{test_loss}, Métrica: {test_mae}")
```

63/63 ————— 0s 2ms/step - loss: 5.4271 - mae: 1.6695
Error:5.255565166473389, Métrica: 1.6799367666244507

Conclusiones

La gestión de inventarios en empresas que operan sin sistemas tecnológicos evidencia enfoques empíricos en los procesos de compras, almacenamiento y ventas. Mediante entrevistas y análisis documental, se identificó que la mayoría de estas organizaciones no emplea herramientas tecnológicas, lo que genera retrasos y disminuye la eficiencia operativa. Este estudio resalta la necesidad urgente de implementar soluciones tecnológicas que optimicen estos procesos, sobre todo en sectores donde el factor tiempo es crítico. En este sentido, el flujograma desarrollado en esta investigación proporciona una base sólida para comprender y mejorar los procesos logísticos de dichas empresas.

El análisis comparativo de diferentes modelos de gestión de inventarios demostró que no existe un modelo universal aplicable a todas las organizaciones. Cada empresa presenta

características particulares —relacionadas con su tamaño, sector económico y comportamiento de demanda— que exigen una selección cuidadosa y personalizada del modelo más adecuado. Los modelos detallados en la Tabla 7 facilitan la adaptación del sistema de inventarios, permitiendo optimizar recursos, reducir costos y mejorar la eficiencia operativa, aspectos fundamentales para el éxito en la gestión de inventarios.

La elección del lenguaje Python para el desarrollo del sistema basado en árboles de decisión y redes neuronales resultó ser acertada, gracias a su facilidad de uso, su extensa librería especializada y su capacidad de integración con otros sistemas. Python facilitó no solo el desarrollo y mantenimiento del software, sino también la implementación de modelos complejos, convirtiéndose en un aliado clave para el éxito del proyecto.

En total, se desarrollaron tres programas distintos orientados a optimizar la gestión de inventarios: uno basado en fórmulas matemáticas exactas, otro en árboles de decisión y un tercero utilizando redes neuronales. La base teórica y metodológica de estos programas se fundamentó en las fórmulas y modelos expuestos en la obra de Investigación de Taha (2004), que sirvió como referencia principal.

- El primer programa aborda todos los modelos de inventarios mediante fórmulas matemáticas, ofreciendo una solución directa y precisa.
- El segundo programa, basado en árboles de decisión, incorpora una estructura lógica para la toma de decisiones, aplicable también a todos los modelos estudiados.
- Finalmente, el programa de redes neuronales se diseñó como una mejora del modelo anterior, con mayor capacidad de adaptación y predicción para escenarios complejos, aunque limitado a ciertos modelos específicos.

- En conjunto, estos desarrollos representan un avance significativo, combinando el enfoque matemático tradicional con tecnologías innovadoras para lograr soluciones más eficientes y precisas en la gestión de inventarios.

Las simulaciones realizadas evidenciaron diferencias en el desempeño de los programas. Como era esperable, el programa basado en fórmulas matemáticas proporcionó resultados exactos y sirvió como referencia. El modelo de árbol de decisiones mostró una aproximación cercana a los datos reales, aunque con desviaciones notables, mientras que el modelo de redes neuronales presentó desviaciones menores, destacando su mayor precisión y consistencia. Esto confirma que la implementación de redes neuronales constituye un éxito, al reducir significativamente las imprecisiones y demostrar su valor para optimizar la gestión de inventarios en escenarios no lineales o con datos complejos.

Además, se evidencia una optimización de recursos significativa, alcanzando una mejora del 94 % en el ejemplo aplicado con la empresa Águila Bike, lo que valida la eficacia y aplicabilidad práctica del programa desarrollado.

Recomendaciones

El uso efectivo de los programas basados en árboles de decisión y redes neuronales depende fundamentalmente de la calidad y precisión de la base de datos utilizada. Por esta razón, se recomienda complementar ambos enfoques con el programa basado en fórmulas matemáticas, que sirve para generar una base de datos inicial robusta y coherente. En la práctica, se aconseja que, al implementar estos programas en una empresa, primero se utilice el programa matemático para construir una base de datos inicial que luego pueda ser cargada y procesada

por los otros dos programas. De esta manera, al alimentar los modelos predictivos con datos reales y específicos, se mejora considerablemente su rendimiento y precisión.

Actualmente, el sistema cuenta con una base de datos cuyos valores están dispersos, ya que fueron generados aleatoriamente sin un patrón definido, lo que afecta la exactitud de las predicciones. Para optimizar la capacidad predictiva de los modelos, es fundamental trabajar con datos reales y concretos directamente obtenidos de la operación de la empresa. Esto permitirá identificar patrones de comportamiento específicos y delimitar con mayor precisión las características particulares del negocio, lo que se traducirá en una mejora significativa en la predicción y adaptación del sistema a las necesidades reales.

En el proceso de entrenamiento de las redes neuronales, se recomienda utilizar al menos 1000 ciclos de entrenamiento. Aunque esto puede incrementar el tiempo de procesamiento, se ha observado que esta cantidad de ciclos asegura una mejor aproximación a los datos reales en los modelos donde se implementaron redes neuronales, mejorando la calidad y confiabilidad de los resultados obtenidos.

Los programas desarrollados se enfocan específicamente en el cálculo de la cantidad óptima de pedido y el punto de reorden, dada la limitación de otros programas en estos aspectos para modelos de inventarios más específicos. Por ello, se sugiere complementar este *software* con una aplicación básica de gestión de inventarios que permita el registro sistemático de ventas, inventarios y compras. Esta integración facilitará una gestión más integral y eficiente, potenciando los beneficios del programa y mejorando la eficacia en la toma de decisiones.

Referencias bibliográficas

- Bowersox, D. J., Closs, D. J., & Cooper, M. B. (2007). *Gestión logística de la cadena de suministro*. McGraw-Hill. https://gc.scalahed.com/recursos/files/r161r/w25775w/L1LT123_S1_01.pdf
- Breiman, L., Friedman, J., Olshen, R., & Stone, C. (1984). *Classification and regression trees*. Wadsworth International Group. <https://doi.org/10.1201/9781315139470>
- Cazares, E. (2025). *Aplicación de Técnicas de Aprendizaje Supervisado para la Predicción de Ventas en Hentel, mediante la Implementación de Modelos de Machine Learning con Python y Pandas*. Pandas [Tesis de maestría]. Universidad de las Américas. Repositorio Digital Universidad de las Américas. <https://dspace.udla.edu.ec/bitstream/33000/17596/1/UDLA-EC-TMINCD-2025-18.pdf>
- Chopra, S., & Meindl, P. (2016). *Administración de la cadena de suministro: Estrategia, planeación y operación*. Pearson. https://gc.scalahed.com/recursos/files/r161r/w24567w/Sunil_Chopral.pdf
- Efron, B., & Hastie, T. (2021). *Computer Age Statistical Inference: Algorithms, Evidence, and Data Science*. Cambridge University Press. https://books.google.com.ec/books?hl=es&lr=&id=q1ctEAAAQBAJ&oi=fnd&pg=PR15&dq=related:Olcl9TbtLk4J:scholar.google.com/&ots=OMfcP_K_ct&sig=BvoMdHi7-PQPIQj6IjYDRZDK-BA&redir_esc=y#v=onepage&q&f=false

- Géron, A. (2023). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow (3rd ed.)*. O'Reilly Media. https://www.rasa-ai.com/wp-content/uploads/2022/02/Aurélien-Géron-Hands-On-Machine-Learning-with-Scikit-Learn-Keras-and-Tensorflow_-Concepts-Tools-and-Techniques-to-Build-Intelligent-Systems-O'Reilly-Media-2019.pdf
- Paul, S. K., & Azeem, A. (2011). An artificial neural network model for optimization of finished goods inventory. *International Journal of Industrial Engineering Computations*, 2, 431-438. https://www.growingscience.com/ijiec/Vol2/IJIEC_2011_3.pdf
- JetBrains. (2022). *The State of Developer Ecosystem*. JetBrains. <https://www.jetbrains.com/lp/devecosystem-2022/>
- Kagermann, H. (2013). *Recomendaciones para implementar la iniciativa estratégica INDUSTRIA 4.0: Informe final del Grupo de Trabajo Industrie 4.0*. Acatech. https://www.acatech.de/wp-content/uploads/2018/03/Final_report__Industrie_4.0_accessible.pdf
- Qiu, J. (2024). An Analysis of Model Evaluation with Cross-Validation: Techniques, Applications, and Recent Advances. *Advances in Economics, Management and Political Sciences*, 99, 69-72. <https://doi.org/10.54254/2754-1169/99/2024OX0213>
- Vidal Rodríguez, S. M., & Herrera Acosta, R. J. (2005). Modelos de inventario probalísticos aplicando distribuciones discretas de probabilidad. *PROSPECTIVA*, 3,(1), 34-37. <https://www.redalyc.org/pdf/4962/496251105007.pdf>
- Stack Overflow. (2023). *Stack Overflow Developer Survey*. Stack. <https://survey.stackoverflow.co/2023/>

- Méndez, R. (2004). Aplicación de redes neuronales artificiales en la predicción y clasificación de niveles de inventario. *Revista Ingeniería UC*, 11(1), 27-34. <https://www.redalyc.org/pdf/707/70711104.pdf>
- Razali, N., & Wah, Y. (2011). Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests. *Journal of Statistical Modeling and Analytics*, 2(1), 21-33. <https://www.nrc.gov/docs/ml1714/ml17143a100.pdf>
- Parada Gutiérrez, O. (2009). Un enfoque multicriterio para la toma de decisiones en la gestión de inventarios. *Cuadernos de Administración*, 22(38), 169-187. <https://www.redalyc.org/pdf/205/20511730009.pdf>
- Stackscale. (2023, 13 de septiembre). *Los lenguajes de programación más populares de 2023*. Stackscale. <https://www.stackscale.com/es/blog/lenguajes-programacion-mas-populares/>
- Taha, H. A. (2004). *Investigación de operaciones*. Pearson Educación. <https://books.google.com.mx/books?id=3oHztjMSuL8C&printsec=frontcover#v=onepage&q&f=false>
- Van Rossum, G., & Drake, F. L. (2023). *The Python Language Reference Manual (5th ed.)*. Python Software Foundation. <https://docs.python.org/3/reference/index.html>
- Von Neumann, J., & Morgenstern, O. (1947). *Theory of games and economic behavior*. Princeton University Press. <https://uvammm.github.io/docs/theoryofgames.pdf>

Anexos

Anexo 1. Códigos por cada modelo de inventario

<https://github.com/LuisStiven1313/C-digos-por-cada-modelo-de-inventario>

Anexo 2. Código unificado modelos determinísticos

<https://github.com/LuisStiven1313/C-digo-unificado-modelos-determin-sticos>

Anexo 3. Código unificado modelos probabilísticos

<https://github.com/LuisStiven1313/C-digo-unificado-modelos-probabil-sticos>

Anexo 4. Modelos Determinísticos y probabilísticos

<https://github.com/LuisStiven1313/Modelos-Determin-sticos-y-Probabil-sticos>

Anexo 5. Códigos árbol de decisiones y redes neuronales

<https://github.com/LuisStiven1313/Codigo>

Anexo 6. Código muestreo bootstrap

<https://github.com/LuisStiven1313/C-digo-muestreo-bootstrap>

Anexo 7. Código normalidad y T-student

<https://github.com/LuisStiven1313/C-digo-normalidad-y-Tstudent/tree/main>

Anexo 8. Documentación de uso de los programas.

<https://github.com/luisstiven1313/documentaci-n-de-uso-de-los-programas>

Sobre los autores

Luis Stiven Ruano Piscal

Es ingeniero en Logística y Transporte por la Universidad Politécnica Estatal del Carchi. Su trayectoria académica y de investigación se centra en la aplicación de aprendizaje automático para la optimización de procesos. Su perfil técnico destaca por el diseño de programas avanzados para la gestión de inventarios en Python, orientada a la integración de modelos de predicción y de clasificación a través de herramientas como árboles de decisión y redes neuronales, para garantizar la precisión en el cálculo de la cantidad óptima de pedido y los puntos de reorden en entornos logísticos complejos. Cuenta con una destacada experiencia profesional como especialista en operaciones aduaneras y comercio exterior con sólida trayectoria en el uso de ECUAPASS, así como en la gestión de regímenes aduaneros, el control de cruces de fronteras, la supervisión logística de transporte y el aseguramiento del cumplimiento normativo en zonas primarias.

Henry Stalin Tapia Collaguazo

Es ingeniero en Logística y Transporte por la Universidad Politécnica Estatal del Carchi. A nivel académico e investigativo, se ha especializado en la aplicación de técnicas de aprendizaje automático para la optimización y mejora de procesos logísticos, ha desarrollado programas para la gestión y el control de inventarios, utilizando modelos basados en árboles de decisión y redes neuronales, enfocados en apoyar la toma de decisiones y mejorar la eficiencia de las operaciones logísticas. Actualmente, su experiencia profesional se ha desarrolla en el

sector del transporte, donde participa en la gestión de despachos y operaciones logísticas; además, brinda apoyo en procesos aduaneros de importación y exportación. Su perfil integra conocimientos de logística, transporte, comercio exterior y herramientas tecnológicas aplicadas a la optimización de procesos.

Iván Gabriel Mafla Bolaños

Es Ingeniero en Electrónica y Control por la Escuela Politécnica Nacional (Ecuador) y Magíster en Gerencia de Sistemas por la Universidad de las Fuerzas Armadas – ESPE (Ecuador). Su formación académica integra áreas de ingeniería, electrónica, control, gestión de sistemas y administración tecnológica, con orientación al análisis, diseño, gestión y la mejora de soluciones técnicas y organizacionales. Su perfil se relaciona con el fortalecimiento de procesos mediante el uso estratégico de sistemas, tecnologías de control y herramientas de gestión aplicadas a contextos académicos, técnicos e institucionales.

Luis Omar Alpala Alpala

Es Ingeniero Industrial y Magíster en Ingeniería Avanzada de Producción, Logística y Cadena de Suministro por la Universidad Politécnica de Valencia (España), y Doctor en Tecnologías de la Información y la Comunicación por la Universidad de Granada (España). Su trayectoria académica e investigativa se orienta a la logística, la producción, la gestión de la cadena de suministro, la simulación, las tecnologías de la información y la transformación digital aplicada a procesos industriales y educativos. Sus intereses se enfocan en el desarrollo de soluciones innovadoras para la mejora de operaciones, la toma de decisiones y la formación profesional mediante tecnologías emergentes.

Francisco Javier Montalvo Marquez

Es Ingeniero Mecánico por la Escuela Politécnica Nacional (Ecuador), Magíster en Ingeniería Mecánica con mención en Sistemas de Transporte por la Escuela Politécnica Nacional (Ecuador) y Magíster en Estadística Aplicada por la Universidad Politécnica Estatal del Carchi (Ecuador). Su formación académica integra áreas de ingeniería mecánica, sistemas de transporte, estadística aplicada, análisis de datos y gestión técnica de procesos. Su perfil se orienta al estudio, modelamiento y mejora de sistemas industriales y de transporte, con apoyo en herramientas cuantitativas para el análisis, la evaluación y la toma de decisiones en contextos académicos, técnicos e institucionales.

Daniel Mauricio Beltrán del Hierro

Es Ingeniero en Industrias Pecuarias por la Escuela Superior Politécnica de Chimborazo (Ecuador) y Magíster en Gestión de la Producción por la Universidad Técnica de Cotopaxi (Ecuador). Su trayectoria académica y profesional se articula en torno a la gestión de la producción, la mejora de procesos y el análisis de sistemas productivos. Actualmente, desarrolla su labor en el campo de la logística y el transporte, especialmente en el área de producción, con orientación hacia el fortalecimiento de operaciones, procesos técnicos y soluciones aplicadas en entornos académicos e institucionales.



En un entorno empresarial cada vez más dinámico y competitivo, la gestión de inventarios constituye un eje estratégico fundamental para la rentabilidad y agilidad de cualquier organización. El libro *Optimización inteligente de inventarios con árboles de decisión y redes neuronales*: de la teoría a la aplicación surge como una respuesta rigurosa y práctica para modernizar los procesos logísticos, especialmente en pequeñas y medianas empresas de Tulcán.

La presente obra ofrece una ruta clara, práctica y sólidamente fundamentada que transita desde los modelos clásicos determinísticos hasta complejos algoritmos probabilísticos y técnicas de *machine learning*, para enseñar a estudiantes, investigadores y empresarios a traducir conceptos en código funcional utilizando el lenguaje Python.

Su relevancia y valor reside en que, mediante un análisis comparativo, valida estadísticamente cómo la integración de redes neuronales permite alcanzar una mayor precisión y consistencia frente a los métodos tradicionales. De hecho, sus autores han logrado demostrar que el uso de estas herramientas inteligentes puede generar niveles de optimización superiores al 94 % y, en consecuencia, reducir significativamente los costos, en aspectos críticos como el almacenamiento y los recursos disponibles.

En definitiva, en la era de los datos, esta publicación constituye una herramienta indispensable para liderar la transformación digital en la cadena de suministro a través de la toma de decisiones basadas en evidencia.

Unidad de Producción y Difusión Académica y Científica

ISBN 978-9942-625-69-4

