Diplomado de Marketing Analytics de automatizaciones y uso de IA de alto impacto

Cristian David Giraldo Posada
Samuel David Díaz Alzate

Trabajo de grado presentado para optar al título de Ingeniero de Sistemas y Telecomunicaciones

Tutor: Carlos Betancourt Correa, Doctor (PhD) en Ciencias de la educación

Universidad de Manizales
Facultad de Ciencias e Ingeniería
Ingeniería de Sistemas y Telecomunicaciones
Manizales, Caldas, Colombia
2025

Cita	(Diaz Álzate & Giraldo Posada, 2025)
Referencia **Estilo APA 7 (2020)**	Diaz Álzate S. D. & Giraldo Posada C. D. (2025). *Diplomado de Marketing Analytics de automatizaciones y uso de IA de alto impacto* [Trabajo de grado profesional]. Universidad de Manizales. RIDUM: Repositorio Institucional Universidad de Manizales.

Seleccione posgrado UManizales (A-Z), Seleccione cohorte posgrado

Seleccione centro de investigación UManizales (A-Z).

Seleccione grupo de investigación UManizales (A-Z)

Seleccione línea de investigación UManizales (A-Z).

Declaración de inteligencia artificial: el o los autores de este trabajo de grado declaran que han utilizado herramientas de inteligencia artificial (IA), tales como [mencionar herramientas utilizadas, por ejemplo, ChatGPT, Grammarly, Turnitin, Copilot, Gemini, entre otras], de manera ética y responsable, tal como se establece en el Acuerdo UManizales 002 (julio 26 de 2023) sobre propiedad intelectual e IA. Estas herramientas son empleadas como apoyo en la redacción, revisión gramatical y generación de ideas, pero en ningún caso sustituyen el análisis crítico, la argumentación académica ni la originalidad del trabajo. Asimismo, cualquier contenido generado con asistencia de IA está citado y referenciado adecuadamente, garantizando la integridad académica y el cumplimiento de los principios éticos de la investigación.

Biblioteca y Centro de Recursos: biblioteca.umanizales.edu.co

Repositorio Institucional: ridum.umanizales.edu.co

Universidad de Manizales: umanizales.edu.co

Revistas: revistasum.umanizales.edu.co

Fondo Editorial: editorialum.umanizales.edu.co

Tabla de contenido

Siglas, acrónimos y abreviaturas

AI	Artificial Intelligence
AMV	Arquitectura Mínima Viable
API	Application Programming Interface
CDP	Customer Data Platform
CLV	Customer Lifetime Value
Causal AI	Causal Artificial Intelligence
ETL	Extract, Transform, Load
IA	Inteligencia Artificial
LLM	Large Language Model
MAB	Multi-Armed Bandit
MLOps	Machine Learning Operations
MMM	Marketing Mix Modeling
n8n	Plataforma de automatización *low-code*
RL	Reinforcement Learning
RAG	Retrieval Augmented Generation
SDK	Software Development Kit
SDK/Azure	Herramientas de desarrollo de Azure
SQL	Structured Query Language
UI	User Interface
UX	User Experience
XAI	Explainable Artificial Intelligence

Resumen

Este documento presenta el diseño curricular del curso Marketing Analytics de automatizaciones y uso de inteligencia artificial de alto impacto, una propuesta formativa estructurada en cinco unidades progresivas orientadas al diseño, implementación y despliegue de soluciones analíticas aplicadas al marketing. El curso integra conceptos fundamentales de agentes inteligentes, arquitecturas de datos modernas y enfoques causales para la toma de decisiones, junto con técnicas avanzadas como sistemas multiagente, aprendizaje por refuerzo y buenas prácticas de MLOps orientadas a entornos productivos. La metodología combina microclases teóricas con talleres prácticos y retos autónomos, permitiendo que los participantes construyan entregables funcionales como flujos de automatización en n8n, prototipos de agentes en Copilot Studio, modelos de causalidad, tableros analíticos y pipelines reproducibles de despliegue. A lo largo del curso se promueve el pensamiento crítico, la experimentación guiada y el uso responsable de la inteligencia artificial, integrando herramientas contemporáneas tales como feature stores, sistemas RAG multimodales y frameworks de optimización. Esta propuesta está dirigida a profesionales que buscan fortalecer su capacidad para diseñar soluciones basadas en datos dentro del ámbito del marketing, con enfoque en autonomía operativa, escalabilidad y buenas prácticas éticas.

Palabras clave: marketing analytics, inteligencia artificial, automatización, agentes autónomos, causal ai, mlops, xai.

Abstract

This document presents the curricular design of the High-Impact AI and Automation Marketing Analytics Course, a structured program composed of five progressive units aimed at developing, implementing, and deploying analytical solutions within the marketing domain. The course integrates fundamental concepts of intelligent agents, modern data architectures, and causal inference for decision-making, along with advanced techniques such as multi-agent systems, reinforcement learning, and MLOps practices oriented toward production environments. Its methodology combines theoretical micro-lessons with practical workshops and autonomous challenges, enabling participants to build functional deliverables including automation workflows in n8n, agent prototypes in Copilot Studio, causal models, analytical dashboards, and reproducible deployment pipelines. Throughout the program, critical thinking, guided experimentation, and responsible use of artificial intelligence are emphasized, incorporating contemporary tools such as feature stores, multimodal RAG systems, and optimization frameworks. This course is designed for professionals seeking to strengthen their ability to build data-driven marketing solutions with a focus on operational autonomy, scalability, and ethical best practices.

*Keywords***:** marketing analytics, artificial intelligence, automation, autonomous agents, causal ai, mlops, xai.

Introducción

El entorno actual del marketing digital se caracteriza por una creciente complejidad en la gestión de datos, la toma de decisiones y la necesidad de automatizar procesos a gran escala. Dado que las organizaciones requieren sistemas capaces de operar de manera autónoma, predecir comportamientos y optimizar resultados en tiempo real, se hace necesario avanzar más allá de los modelos descriptivos tradicionales. En efecto, la integración entre analítica avanzada, inteligencia artificial y plataformas de automatización *low-code* ha abierto la posibilidad de diseñar soluciones que incrementan la eficiencia operativa, fortalecen la personalización y mejoran la atribución causal de las acciones de marketing.

Teniendo en cuenta este panorama, el presente documento se elabora con el propósito de estructurar y describir el diseño curricular del curso *Marketing Analytics de automatizaciones y uso de inteligencia artificial de alto impacto*. De igual manera, busca explicar las razones que justifican su creación, puesto que existe una demanda creciente de profesionales capaces de comprender, construir y desplegar sistemas analíticos basados en datos dentro de entornos reales.

A saber, el curso se fundamenta en unidades que abordan desde conceptos básicos de agentes inteligentes y arquitecturas de datos hasta técnicas avanzadas como inteligencia de grafos, sistemas multiagente, aprendizaje por refuerzo y prácticas de MLOps orientadas al despliegue responsable. Por consiguiente, se plantea como objetivo central que los participantes desarrollen competencias técnicas y aplicadas que les permitan responder a los desafíos contemporáneos del marketing.

Así pues, el documento se justifica en la necesidad de formar perfiles capaces de diseñar soluciones reproducibles, escalables y éticamente responsables. Simultáneamente, se emplea una metodología basada en la práctica, la experimentación guiada y la construcción de un portafolio de proyectos funcionales que evidencien la capacidad para resolver problemas complejos mediante tecnologías de vanguardia.

1 Planteamiento del problema

El acelerado crecimiento del marketing digital ha generado una demanda constante de soluciones basadas en datos capaces de responder con precisión a dinámicas cada vez más complejas. Sin embargo, simultáneamente se observa una brecha significativa entre el avance de las tecnologías de inteligencia artificial y automatización, y la capacidad real de los profesionales para implementar estas herramientas de manera efectiva. Aunque existen iniciativas formativas relacionadas con analítica, inteligencia artificial o automatización por separado, persiste la ausencia de programas integrales que articulen estos componentes dentro de un enfoque práctico orientado a la resolución de problemas reales del marketing.

Dado que las organizaciones requieren sistemas autónomos que integren predicción, optimización y toma de decisiones en tiempo real, se vuelve indispensable contar con talento capaz de diseñar y desplegar arquitecturas modernas que incluyan agentes inteligentes, causalidad aplicada, sistemas multiagente, *feature stores*, aprendizaje por refuerzo y prácticas robustas de MLOps. No obstante, la mayoría de los profesionales encuentra dificultades para dominar estas herramientas debido a la falta de formación estructurada, actualizada y centrada en la práctica.

Así, el problema central que aborda este documento consiste en la necesidad de diseñar un curso que responda a esta brecha formativa, permitiendo el desarrollo de competencias aplicadas en automatización y uso de inteligencia artificial en el ámbito del marketing. En síntesis, se plantea la creación de un programa que combine fundamentos teóricos, experimentación guiada y construcción de proyectos funcionales orientados a entornos reales de implementación.

1.1 Antecedentes

El desarrollo de soluciones basadas en analítica avanzada e inteligencia artificial ha sido ampliamente documentado en la literatura académica y técnica durante las últimas dos décadas. Diversos estudios han demostrado la importancia de integrar modelos predictivos, automatización y toma de decisiones asistida por datos en sectores como el marketing digital, dado que estas tecnologías permiten mejorar la segmentación, optimizar inversiones publicitarias y aumentar la eficiencia operativa. Investigaciones recientes también han señalado el avance de los sistemas

autónomos y los agentes inteligentes como herramientas clave para automatizar flujos de trabajo y personalizar interacciones a gran escala.

Por otra parte, la evolución de las arquitecturas de datos incluyendo customer data platforms, feature stores y bases vectoriales ha permitido consolidar entornos más robustos para la implementación de modelos de aprendizaje automático y sistemas de recomendación. Simultáneamente, enfoques como la inteligencia de grafos, la causalidad aplicada y el aprendizaje por refuerzo han incrementado su presencia en estudios sobre optimización y atribución de impacto dentro del marketing, ampliando el repertorio de metodologías disponibles para resolver problemas complejos.

Aunque estas líneas de investigación han progresado de manera significativa, se identifican vacíos en la oferta formativa que integren todos estos componentes dentro de un enfoque práctico y orientado a proyectos. Los programas existentes suelen abordar la analítica, la inteligencia artificial o la automatización de manera independiente, dejando de lado la articulación entre ellos y la necesidad de formar profesionales capaces de construir sistemas autónomos completos. Estos antecedentes evidencian la pertinencia de diseñar un curso integral que responda a las exigencias actuales del marketing basado en datos.

2 Justificación

La rápida adopción de herramientas de analítica avanzada, automatización y agentes inteligentes en el ámbito del marketing ha generado una demanda urgente de perfiles profesionales capaces no solo de comprender modelos y tecnologías, sino de diseñar, integrar y desplegar soluciones operativas en entornos reales. Dado que la oferta formativa existente tiende a segmentar conocimientos —separando analítica, inteligencia artificial y automatización en cursos disociados— se detecta una brecha formativa: la ausencia de programas integrales orientados a la práctica que articulen arquitectura de datos, automatizaciones low-code, modelos causales, sistemas multiagente y prácticas de MLOps.

Por consiguiente, la creación de este curso se justifica en la necesidad de formar profesionales con competencias aplicadas que permitan responder a problemáticas contemporáneas del marketing digital: optimización del gasto publicitario, personalización a escala, atribución causal y automatización de procesos. Además, la propuesta contribuye a la transferencia tecnológica y a la competitividad organizacional, al promover metodologías reproducibles, criterios de gobernanza de datos y principios de explicabilidad y ética en despliegues productivos.

Finalmente, esta iniciativa apoya las metas institucionales de vinculación con el entorno productivo y la formación de talento pertinente al mercado; de igual manera, entrega un portafolio de evidencias y buenas prácticas que facilitan la evaluación, la replicabilidad y la adopción en contextos empresariales reales.

3 Objetivos

3.1 Objetivo general

Diseñar e implementar un programa formativo práctico que capacite a profesionales para construir, evaluar y desplegar soluciones de Marketing Analytics basadas en automatizaciones y agentes de inteligencia artificial, garantizando trazabilidad, gobernanza y criterios éticos a lo largo del ciclo de producción.

3.2 Objetivos específicos

- Diseñar arquitecturas de datos modernas aplicadas al marketing (CDP, feature stores, bases vectoriales) que permitan la operación de modelos reproducibles y la integración de pipelines de inferencia.
- Desarrollar e implementar flujos de automatización y agentes inteligentes utilizando herramientas low-code y LLMs integrados, orientados a casos de uso de marketing.
- Aplicar métodos de inferencia causal y modelos de uplift para evaluar incrementalidad y atribución en campañas de marketing.
- Construir y comparar modelos predictivos y de optimización (forecasting, MMM, MAB/RL) para soportar decisiones de inversión y asignación en marketing.
- Diseñar y desarrollar prototipos de sistemas multiagente y políticas de decisión autónoma que respondan a escenarios en tiempo real.
- Implementar prácticas de MLOps, monitoreo, versionado y despliegue reproducible, incorporando criterios de explicabilidad, fairness y gobernanza.
- Comunicar hallazgos técnicos y ejecutivos mediante entregables claros y reproducibles (dashboards, resúmenes ejecutivos, documentación técnica) que faciliten la adopción por stakeholders.

4 Supuestos de trabajo

Se asume que la implementación del curso permitirá a los participantes desarrollar competencias prácticas suficientes para construir y desplegar al menos un entregable funcional.

Se asume que la metodología basada en práctica guiada y retos autónomos facilita la transferencia de habilidades técnicas al contexto laboral real.

Se asume que las arquitecturas y herramientas seleccionadas (feature stores, RAG multimodal, n8n, Copilot Studio y prácticas MLOps) son pertinentes y viables dentro del entorno operativo objetivo.

4.1 Preguntas de diseño y validación

- ¿En qué medida los participantes son capaces de reproducir los pipelines y artefactos propuestos dentro de un entorno productivo?
- ¿Qué evidencia cuantitativa y cualitativa permite afirmar que la formación mejora la capacidad de atribución y optimización en campañas reales?
- ¿Qué ajustes metodológicos y de herramientas son necesarios para garantizar escalabilidad y gobernanza en despliegues reales?

5 Marco teórico

El campo del marketing analytics integra técnicas estadísticas, minería de datos y aprendizaje automático para transformar señales de interacción en decisiones de negocio medibles. Su transición desde reportes descriptivos hacia sistemas que soportan decisiones automatizadas obedece tanto a la mayor disponibilidad de datos como a mejoras metodológicas en modelado predictivo y en la instrumentación de experimentos (Provost & Fawcett, 2013; Hastie, Tibshirani, & Friedman, 2009). En entornos digitales, la analítica persigue objetivos concretos: segmentación dinámica, atribución de impactos y optimización del gasto publicitario. Por consiguiente, la formación en marketing analytics exige no solo conocimiento teórico, sino habilidades para operacionalizar modelos en pipelines reproducibles (Provost & Fawcett, 2013).

La operacionalización de modelos de IA requiere arquitecturas que soporten captura, unificación, versionado y servicio de características (features) en producción. Conceptos como *customer data platforms, feature stores* y bases vectoriales permiten separar la etapa de experimentación del servicio en línea, reduciendo la fricción entre prototipo y producción (Sculley et al., 2015). Estas arquitecturas demandan pipelines ETL/ELT robustos, catálogos de datos y mecanismos de gobernanza para asegurar calidad y trazabilidad de las variables alimentadas a modelos (Provost & Fawcett, 2013; Sculley et al., 2015).

Los agentes inteligentes y las plataformas *low-code* facilitan la orquestación de flujos para casos de uso en marketing como la gestión de leads, la personalización y la orquestación de campañas. La integración de modelos de lenguaje y orquestadores reduce tiempo de desarrollo y facilita iteraciones rápidas, siempre que se establezcan controles de gobernanza, logs y métricas que permitan auditoría y reproducibilidad (Molnar, 2020; Ribeiro, Singh, & Guestrin, 2016).

Para estimar el efecto real de intervenciones de marketing es necesario aplicar métodos de inferencia causal que superen las limitaciones de la correlación. Enfoques del marco de potencial outcomes y herramientas prácticas como uplift modeling, controles sintéticos y métodos contrafactuales ofrecen vías para estimar incrementalidad y atribución con mayor rigor (Rubin, 1974; Pearl, 2009). La incorporación de causalidad en los flujos analíticos permite decisiones presupuestales más robustas y la validación empírica de políticas de intervención.

Las técnicas de *forecasting* y el *Marketing Mix Modeling* (MMM) proporcionan estimaciones de elasticidades y efectos de canales a nivel agregado, mientras que algoritmos

adaptativos (Multi-Armed Bandits) y políticas derivadas de Reinforcement Learning permiten optimizar decisiones en tiempo real bajo incertidumbre (Sutton & Barto, 2018). La complementariedad entre modelos explicativos y algoritmos de optimización es crucial para cubrir tanto la necesidad de interpretación como la de rendimiento operacional.

Los sistemas multiagente facilitan la modelización de entornos donde múltiples actores (consumidores, campañas, canales) interactúan y generan comportamientos emergentes. En marketing complejo, estas arquitecturas permiten simular y evaluar estrategias distribuidas, así como diseñar políticas coordinadas para escenarios dinámicos (Sutton & Barto, 2018).

MLOps engloba prácticas de versionado de datos y modelos, pipelines reproductibles, testing automatizado y monitoreo en producción. Estas prácticas reducen la deuda técnica y facilitan la trazabilidad y mantenimiento de soluciones analíticas (Sculley et al., 2015). Integrar MLOps en la formación asegura que los participantes comprendan no solo la construcción de modelos, sino su operación responsable y sostenible en entornos empresariales.

Las técnicas de explicabilidad (interpretabilidad local y global) y los marcos de fairness son fundamentales para evaluar riesgos y construir mecanismos de apelación en decisiones automatizadas. Herramientas como LIME y SHAP han demostrado utilidad práctica para explicar predicciones y detectar comportamientos inesperados en modelos complejos (Ribeiro et al., 2016; Lundberg & Lee, 2017). La formación en XAI contribuye a la aceptabilidad organizacional y al cumplimiento de criterios éticos en despliegues reales.

La convergencia de los campos anteriores sugiere que una propuesta curricular en Marketing Analytics debe articular: (a) fundamentos teóricos de causalidad y modelado; (b) habilidades operativas en arquitecturas y herramientas; (c) experiencia práctica mediante proyectos reproducibles; y (d) competencias en MLOps y ética. La metodología basada en *learning by doing*, talleres guiados y rúbricas técnicas favorece la transferencia de competencias y la generación de evidencia verificable (Provost & Fawcett, 2013; Molnar, 2020).

6 Metodología

6.1. Enfoque metodológico

El presente trabajo corresponde a un diseño curricular aplicado con enfoque cualitativo y descriptivo. Su propósito no es contrastar hipótesis experimentales, sino desarrollar y justificar una propuesta formativa basada en evidencia técnica y pedagógica. Para tal fin se combina revisión documental, análisis de necesidades, diseño instruccional y validación técnica de prototipos.

6.2. Tipo de estudio y diseño

Se trata de un estudio de tipo propuesta o diseño curricular. El diseño se apoya en marcos teóricos de marketing analytics, aprendizaje automático y MLOps, y en criterios pedagógicos centrados en el aprendizaje activo. El curso se concibe como un programa modular y progresivo, orientado a la construcción de artefactos reproducibles.

6.3. Procedimiento metodológico

- Los pasos seguidos para la elaboración del diseño curricular fueron:
- Identificación de necesidades formativas mediante análisis de contexto y revisión de competencias requeridas en el sector.
- Revisión de literatura especializada sobre marketing analytics, causalidad, agentes inteligentes, MLOps y XAI.
- Selección de herramientas y tecnologías pertinentes (n8n, Copilot Studio, feature stores, frameworks RL).
- Definición de competencias y resultados de aprendizaje por unidad.
- Diseño de estructura de unidades, actividades, talleres y retos autónomos.
- Elaboración de rúbricas y criterios de evaluación enlazados a evidencias reproducibles.
- Validación técnica y pedagógica mediante pilotos internos y revisión por pares.

  Cada etapa incorpora criterios de trazabilidad y gobernanza para garantizar reproducibilidad y evaluación objetiva.

6.4. Estructura curricular propuesta

6.4.1. Organización del curso

El curso está organizado en cinco unidades progresivas más una unidad introductoria (Unidad 0). La progresión sigue un orden lógico: fundamentos y arquitectura, datos y causalidad, predicción y modelado, agentes y sistemas en tiempo real, y MLOps y

despliegue. La metodología combina microclases, talleres guiados y retos autónomos con entregables reproducibles.

6.4.2. Matriz de unidades

Cada unidad contempla los siguientes elementos obligatorios:

- Nombre de la unidad
- Objetivo específico
- Contenidos principales
- Estrategias de enseñanza
- Actividades de aprendizaje (talleres y retos)
- Evidencias y productos esperados
- Criterios de evaluación y rúbrica asociada

La matriz se presenta en un anexo o como tabla según prefiera la plantilla; en el cuerpo se incluye el detalle narrativo de cada unidad.

6.4.3 Detalle de las unidades

Detalle de la Unidad 0: Fundamentos de agentes con IA y automatizaciones (10 horas)

Resumen ejecutivo

Unidad introductoria de 10 horas que combina fundamentos conceptuales y práctica aplicada para comprender la anatomía, la arquitectura minima viable y los flujos básicos de automatización de un agente inteligente. Se aborda cómo los agentes de IA pueden ejecutar acciones autónomas conectando datos, reglas y modelos, y cómo herramientas low-code como n8n y Copilot Studio permiten construir prototipos funcionales sin programación avanzada. El entregable final es un flujo de automatización funcional en n8n y un agente operativo en Copilot Studio que demuestre comprensión del ciclo de percepción, decisión y acción.

Objetivos de la unidad

- Comprender la anatomía de un agente inteligente: componentes, tipos y flujo de información.
- Diseñar la arquitectura mínima viable de un agente con enfoque modular: percepción, decisión, acción.

- Implementar automatizaciones básicas en n8n y vincularlas a Copilot Studio como interfaz natural de interacción.
- Aplicar principios de gobernanza, trazabilidad y seguridad en el diseno de flujos automatizados.
- Documentar y probar un flujo de agente funcional con checklist de evaluación técnica y ética.

Resultados de aprendizaje (medibles)

Al finalizar la unidad, el participante podrá:

1. Explicar la anatomía y el ciclo de vida de un agente autónomo, diferenciando entre agente simple, colaborativo y multiagente.
2. Construir un flujo de automatización básico en n8n que integre un trigger, una lógica de decisión y una acción ejecutable.
3. Diseñar un agente funcional en Copilot Studio conectado a una API o flujo de n8n, con al menos un prompt operativo y un comando accionable.
4. Implementar un mecanismo básico de registro de logs y control de errores.
5. Entregar un video o demostración funcional con checklist de cumplimiento técnico y ético.

Duración y estructura (10 horas)

- Bloque A: Introducción y anatomía de los agentes inteligentes (2 h)
- Bloque B: Arquitectura mínima viable y flujos lógicos (2 h)
- Bloque C: Automatizaciones con n8n: percepción y accion (3 h)
- Bloque D: Integracion con Copilot Studio: interfaz y orquestacion (2 h)
- Bloque E: Gobernanza, seguridad y evaluacion etica de agentes (1 h)

Cada bloque incluye: microclase (20 a 40 minutos), taller guiado (60 a 120 minutos segun bloque) y reto autonomo (20 a 45 minutos). Al final de la unidad se verifica cumplimiento mediante checklist de evidencia.

Recursos y setup (prioridad)

Requisitos minimos:

- Cuenta en n8n (self-hosted o cloud).
- Acceso a Microsoft Copilot Studio (cuenta Microsoft).
- Navegador moderno, acceso a APIs publicas sencillas.
- Dataset o API simulada para inputs.

- Editor de texto y espacio de trabajo n8n.

Sandboxes y alternativas:

- n8n Cloud version gratuita.
- Webhook Tester o JSONPlaceholder para simulacion de triggers.
- Plantilla institucional de Copilot preconfigurada.

Bloque A — Anatomia de un agente inteligente (2 h)

Objetivo: Comprender los fundamentos conceptuales y estructurales de un agente inteligente.

Contenido clave:

- Definicion de agente y tipos: reactivo, deliberativo, colaborativo.
- Ciclo de vida: percepcion, razonamiento, accion.
- Arquitectura conceptual: sensores, entorno, politica de decision, actuadores.
- Ejemplos en marketing y analitica: agentes recolectores de datos, de recomendacion y de optimizacion.

Taller guiado A1 (60 min):

- Elaboracion de un mapa conceptual del flujo de informacion entre componentes.
- Identificacion de oportunidades de automatizacion en contexto real del participante.

Reto autonomo A2 (30 a 45 min):

- Documentar el ciclo percepcion-accion de un agente relevante para su organizacion.

Checklist A:

- Infografia o diagrama del ciclo de vida de un agente.
- Documento explicativo de 1 pagina con ejemplo contextualizado.

Bloque B — Arquitectura minima viable y flujos logicos (2 h)

Objetivo: Disenar la arquitectura base de un agente con componentes interconectados.

Contenido clave:

- AMV de un agente: estructura modular (trigger, logica, accion).
- Flujo logico de decision: condicionales, bucles y manejo de estados.
- Representacion visual en flujos low-code (n8n).

Taller guiado B1 (60 min):

- Creacion de diagrama logico (pseudocodigo o mapa de flujo) para un agente de respuesta automatizada.

Reto autonomo B2 (30 a 45 min):

- Implementar el diagrama funcional con pseudocodigo o un archivo JSON de flujo.

Checklist B:

- Diagrama AMV con tres componentes (input, decision, accion).
- Flujo logico exportado como imagen o archivo JSON.

Bloque C — Automatizaciones con n8n: percepcion y accion (3 h)

Objetivo: Construir flujos de automatizacion funcionales en n8n.

Contenido clave:

- Creacion de nodos: trigger (webhook, schedule, event).
- Nodos de procesamiento: function, if, http request, data transform.
- Acciones: envio de correo, actualizacion en Notion o Sheets, mensaje en Slack o Teams.
- Control de errores y logging basico.

Taller guiado C1 (90 a 120 min):

- Construir un flujo: deteccion de evento, evaluacion condicional y accion en canal.
- Prueba de ejecucion y revision de logs.

Reto autonomo C2 (30 a 45 min):

- Crear flujo con tres nodos y logica condicional. Documentar caso de uso.

Checklist C:

- Flujo ejecutable en n8n (archivo JSON exportado).
- Captura del dashboard de ejecucion.

Bloque D — Integracion con Copilot Studio: interfaz y orquestacion (2 h)

Objetivo: Conectar Copilot Studio con flujos n8n para habilitar agentes conversacionales o de comando.

Contenido clave:

- Introduccion a Copilot Studio: intents, actions y variables.
- Conexion de acciones Copilot a n8n via webhook o API.
- Ejemplo: comando "actualizar tablero" o "consultar clima".
- Validaciones y control de errores.

Taller guiado D1 (60 min):

- Configurar Copilot Studio con una accion que dispare un flujo en n8n.

Reto autonomo D2 (30 a 45 min):

- Crear un Copilot con un comando personalizado y probar la ejecucion.

Checklist D:

- Copilot funcional con un intent y accion conectada.
- Evidencia: video corto o captura del flujo en ejecucion.

Bloque E — Gobernanza, seguridad y evaluacion etica (1 h)

Objetivo: Incorporar principios de gobernanza, transparencia y seguridad en el diseno de agentes.

Contenido clave:

- Control de acceso y registro de actividad.
- Trazabilidad y logs de decisiones.
- Revision etica: sesgos, privacidad, consentimiento.
- Buenas practicas de documentacion y auditoria.

Taller guiado E1 (30 a 45 min):

- Crear checklist de riesgos y medidas de mitigacion.

Checklist E:

- Documento de buenas practicas eticas.
- Configuracion basica de logging y auditoria.

Entregables finales de la unidad (ZIP)

- Flujo n8n exportado (.json).
- Copilot configurado (captura o archivo de intent).
- Documento PDF de 1 a 2 paginas con diagrama AMV, evidencias y reflexion sobre gobernanza.
- Checklist de validacion etica y tecnica.

Rúbrica de evaluación (100 puntos)

- Comprension conceptual de agentes y AMV: 25 puntos
- Implementacion tecnica (flujo n8n y Copilot): 25 puntos
- Diseno de gobernanza y registro de decisiones: 15 puntos
- Documentacion y reproducibilidad: 15 puntos
- Presentacion ejecutiva y claridad de resultados: 20 puntos

Aprobacion minima: 60 puntos de 100.

Snippets y plantillas utiles (pegables, en texto simple)

Ejemplo de flujo basico en n8n (pseudocodigo)

nodes:

- webhook: event new_message
- if: condition message contains "alerta"
- httpRequest: action send Slack notification

Ejemplo de intent en Copilot Studio (texto)

Intent: Consultar estado

Action: Llama flujo n8n "GetStatus" via webhook

Response: "El sistema esta operativo y actualizado al {fecha}"

Pruebas de aceptacion (minimas)

- Ejecutar flujo n8n con exito y registro en log.
- Copilot responde al menos a un intent con una accion real.
- El ZIP contiene todos los archivos listados.
- Checklist etico completado.

Si todas las pruebas se ejecutan y producen los artefactos listados, resultado: APROBADO.

Riesgos y mitigaciones

- Riesgo: falta de familiaridad con n8n. Mitigacion: proveer video tutorial inicial.
- Riesgo: fallos de conexion a APIs. Mitigacion: usar mocks o endpoints locales.
- Riesgo: errores en Copilot Studio por permisos. Mitigacion: usar sandbox institucional.
- Riesgo: automatizaciones sin control. Mitigacion: checklist etico obligatorio.

Recomendaciones pedagogicas

- Incluir prework sobre automatizacion low-code y flujos logicos basicos.
- Promover trabajo colaborativo en la creacion de un flujo compartido.
- Evaluar en sesion practica los resultados de ejecucion de cada agente.
- Finalizar con una mini-demostracion (pitch tecnico-ejecutivo de 3 a 5 minutos).

Unidad I: Fundamentos, Arquitecturas de Datos y Causal AI (10 horas)

Resumen ejecutivo

Unidad de 10 horas enfocada en los fundamentos conceptuales y prácticos que sustentan el Marketing Analytics autónomo: arquitecturas modernas de datos, estrategias de first-party data y el uso de Causal AI para estimar impacto real en marketing. La unidad combina teoría y práctica mediante el diseño de una arquitectura tipo CDP (Customer Data Platform) con Feature Store y base vectorial, y la implementación de un experimento causal simple para estimar incrementalidad. El entregable final incluye un esquema arquitectónico funcional y un notebook con simulación causal básica.

Objetivos de la unidad

- Comprender los principios del marketing basado en datos y la transición hacia arquitecturas centradas en datos propios (first-party data).
- Diseñar una arquitectura moderna de datos para IA: CDP + Feature Store + base vectorial.
- Entender los fundamentos de Causal AI y su relevancia en la medición de impacto y atribución.
- Aplicar un modelo causal (uplift o contrafactual simple) con datos de marketing.
- Documentar el flujo de datos, dependencias y métricas de calidad del pipeline.

Resultados de aprendizaje (medibles)

Al finalizar la unidad, el participante podrá:

1. Diagramar una arquitectura CDP + Feature Store + base vectorial, identificando flujos de entrada, procesamiento y salida.
2. Construir un esquema básico de Feature Store (en Python o pseudocódigo) con tres features de comportamiento y uno contextual.
3. Implementar un notebook simple con simulación causal (modelo contrafactual o uplift) y reportar resultados de incrementalidad.
4. Evaluar la calidad de datos mediante métricas: completitud, unicidad y consistencia.
5. Entregar un documento técnico-ejecutivo con el diseño arquitectónico y los resultados del experimento causal.

Duración y estructura (10 horas)

- Bloque A: Fundamentos del marketing basado en datos y first-party data (2 h)
- Bloque B: Arquitectura moderna de datos: CDP y Feature Stores (2.5 h)

- Bloque C: Bases vectoriales y RAG para marketing (2 h)
- Bloque D: Fundamentos de Causal AI y modelado de impacto (2.5 h)
- Bloque E: Integración y documentación del pipeline (1 h)

Cada bloque incluye: microclase (20 a 40 minutos), taller guiado (60 a 90 minutos) y reto autónomo (20 a 45 minutos). Al final de la unidad se verifica cumplimiento mediante checklist de evidencia.

Recursos y setup (prioridad)

Requisitos mínimos:

- Python 3.9 o superior y librerías: pandas, numpy, scikit-learn, featuretools, sentence-transformers, causalml, matplotlib.
- Entorno de notebooks: JupyterLab o Google Colab.
- Dataset sintético mínimo de 1000 filas con variables: id_cliente, canal, inversion, conversiones, revenue, fecha.
- Opcional: Weaviate o Pinecone (free tier) para base vectorial, o FAISS local como simulación.
- Editor de diagramas: Draw.io, Excalidraw o Miro.

Sandboxes y alternativas low-cost:

- Dataset "Uplift Modeling Synthetic" de causalml o generadores sintéticos en Python.
- Feature Store simulado con pandas y estructuras en disco.
- RAG simulado con embeddings locales usando SentenceTransformers y búsqueda semántica con FAISS.

Bloque A — Fundamentos del marketing basado en datos y first-party data (2 h)

Objetivo: Comprender la importancia estratégica de los datos propios y el cambio de paradigma tras la eliminación de cookies.

Contenido clave:

- Transición de third-party a first-party data.
- Beneficios de propiedad y control de datos en marketing.
- Elementos de una estrategia de datos: fuentes, consentimiento, activacion.
- Estructura mínima de una arquitectura basada en datos propios.

Taller guiado A1 (60 min):

- Mapear fuentes de datos propias (CRM, web, social, offline).

- Construir flujo conceptual de ingesta, unificación y activación.

Reto autónomo A2 (30 a 45 min):

- Diseñar un esquema simple de CDP con tres capas: ingesta, unificación y activación.

Checklist A:

- Diagrama CDP con tres capas.
- Lista de fuentes y tipo de dato (identificable / no identificable).

Bloque B — Arquitectura moderna de datos: CDP y Feature Stores (2.5 h)

Objetivo: Comprender la estructura técnica de una arquitectura moderna para IA de marketing.

Contenido clave:

- CDP: ingestión, identidad, unificación y activación.
- Feature Store: concepto, ventajas y estructura.
- Definición de feature, entidad y metadato.
- Versionado y reutilización de features en modelos.

Taller guiado B1 (90 min):

- Construir un mini Feature Store con pandas.
- Crear tres features: frecuencia de compra, valor promedio, dias desde ultima compra.

Reto autónomo B2 (30 a 45 min):

- Agregar un feature contextual (por ejemplo, canal dominante o categoria de producto).

Checklist B:

- Notebook con función create_feature_store(df) que retorne features listos para modelar.
- CSV con tabla de features y metadatos.

Bloque C — Bases vectoriales y RAG para marketing (2 h)

Objetivo: Introducir los principios de búsqueda semántica y su aplicación en marketing.

Contenido clave:

- Concepto de embeddings y espacio vectorial.
- Bases vectoriales: Weaviate, Pinecone, FAISS.
- RAG: Retrieval-Augmented Generation y su integración con modelos generativos.
- Aplicaciones: recomendación, clasificación de contenido, respuesta contextual.

Taller guiado C1 (60 a 90 min):

- Crear embeddings de texto (productos o mensajes publicitarios) con SentenceTransformers.
- Búsqueda de similitud semántica entre mensajes o audiencias.

Reto autónomo C2 (30 a 45 min):

- Construir función get_similar_texts(query, k=3) y documentar resultados.

Checklist C:

- Notebook embeddings.ipynb con resultados de similitud.
- Tabla o captura con top-3 resultados más similares.

Bloque D — Fundamentos de Causal AI y modelado de impacto (2.5 h)

Objetivo: Comprender cómo estimar efectos causales e incrementalidad mediante modelos contrafactuales.

Contenido clave:

- Causalidad versus correlación.
- Diseño contrafactual y ATE (Average Treatment Effect).
- Uplift modeling, matching y propensity scores.
- Herramientas: causalml, DoWhy (introducción conceptual).

Taller guiado D1 (90 a 120 min):

- Simular dataset con tratamiento (exposicion a campaña) y resultado.
- Calcular ATE y uplift usando causalml.
- Visualizar incrementalidad con intervalos de confianza.

Reto autónomo D2 (30 a 45 min):

- Implementar un test de robustez (placebo o permutation test).

Checklist D:

- Notebook causal_ai.ipynb con código ejecutable.
- Reporte corto (1 a 2 paginas) con conclusiones sobre el efecto causal.

Bloque E — Integración y documentación del pipeline (1 h)

Objetivo: Integrar todos los componentes (CDP, Feature Store, Causal AI) y documentar el flujo de datos.

Contenido clave:

- Flujo end-to-end: ingesta, procesamiento, modelado y decisión.
- Métricas de calidad de datos.
- Documentación y versionado de artefactos.

Taller guiado E1 (45 a 60 min):

- Diagramar el pipeline completo con herramienta visual.

Checklist E:

- Diagrama de flujo del pipeline.
- Métricas de calidad calculadas: completitud, duplicados, consistencia.

Entregables finales de la unidad (ZIP)

- Notebook Feature Store (feature_store.ipynb).
- Notebook Causal AI (causal_ai.ipynb).
- Diagrama arquitectónico (imagen o PDF).
- Documento ejecutivo (1 a 2 paginas) con flujo de datos, resultados y recomendaciones.
- Checklist de calidad de datos y evidencia causal.

Rúbrica de evaluación (100 puntos)

- Comprensión de arquitectura y flujo de datos: 25 puntos
- Implementación técnica (Feature Store + Causal AI): 30 puntos
- Análisis e interpretación causal: 20 puntos
- Calidad de documentación y reproducibilidad: 15 puntos
- Presentación ejecutiva y conclusiones: 10 puntos

Aprobación mínima: 60 puntos de 100.

Snippets y plantillas útiles (pegables, en texto simple)

Ejemplo: creación de Feature Store (Python)

```
def create_feature_store(df):
df['freq_compra'] = df.groupby('id_cliente')['id_cliente'].transform('count')
df['avg_ticket'] = df.groupby('id_cliente')['revenue'].transform('mean')
df['days_since_last'] = (df['fecha'].max() - df['fecha']).dt.days
features = df[['id_cliente','freq_compra','avg_ticket','days_since_last']].drop_duplicates()
return features
```

Ejemplo: cálculo causal simple (causalml)

```
from causalml.inference.tree import UpliftTreeClassifier
uplift = UpliftTreeClassifier(max_depth=4, min_samples_leaf=100)
uplift.fit(X, treatment, y)
uplift.plot()
```

Pruebas de aceptación (mínimas)

- feature_store.ipynb ejecuta y genera CSV de features.
- causal_ai.ipynb produce estimación de ATE o uplift.
- Diagrama del pipeline entregado.
- Checklist de calidad de datos completado.

Si todas las pruebas se ejecutan y producen los artefactos listados, resultado: APROBADO.

Riesgos y mitigaciones

- Riesgo: falta de datos reales. Mitigación: uso de datasets sintéticos o sandbox.
- Riesgo: confusión entre causalidad y correlación. Mitigación: enfatizar diseño experimental y pruebas de robustez.
- Riesgo: complejidad técnica de Feature Stores. Mitigación: usar pandas como proxy y ejemplos simplificados.
- Riesgo: métricas inconsistentes. Mitigación: validación cruzada con checklist de calidad.

Recomendaciones pedagógicas

- Incluir prework sobre estructuras de datos y nociones básicas de causalidad.
- Fomentar trabajo en parejas para diseñar y comparar arquitecturas.
- Evaluar resultados en una sesión de revisión técnica y de negocio.
- Promover reflexión final sobre ética en uso de datos y atribución.

Unidad II: Predicción Avanzada, Marketing Mix Modeling (MMM) y Graph Intelligence (15 horas)

Resumen ejecutivo

Unidad práctica de 15 horas enfocada en metodologías y herramientas para forecast de demanda y performance (Prophet, BSTS), modelado de Marketing Mix bayesiano para estimar elasticidades y optimizar presupuesto, diseño de experimentos para atribución e incrementalidad (A/B, geo-lift), y aplicación de Graph Intelligence para analizar comunidades, influencia y similitud entre clientes y contenidos. El entregable final incluye notebooks reproducibles, modelos, un dashboard ejecutivo con insights accionables y un protocolo de validación causal.

Objetivos de la unidad

- Implementar modelos de forecasting con Prophet y BSTS y comparar su desempeño en series de marketing y ventas.
- Construir un modelo MMM bayesiano que estime elasticidades por canal y proponga redistribución presupuestal optimizada.
- Diseñar y analizar experimentos (A/B y geo-lift) para estimar incrementalidad con robustez estadística.
- Aplicar técnicas de Graph Intelligence (detección de comunidades, centralidad, embeddings) para enriquecer segmentaciones y estrategias de targeting.

Resultados de aprendizaje (medibles)

Al finalizar la unidad, el participante podrá:

1. Entregar un notebook reproducible que ejecute forecasting con Prophet y un modelo BSTS, mostrando métricas comparadas (MAE, RMSE, coverage de intervalos).
2. Presentar un notebook con un MMM bayesiano que incluya estimaciones de elasticidad por canal y una simulación de redistribución presupuestal con impacto estimado en KPI.
3. Documentar un protocolo experimental (A/B o geo-lift) con cálculo de tamaño muestral, análisis de resultados, intervalos de confianza y medidas de incrementalidad.
4. Entregar un análisis de grafo (script o notebook) que identifique comunidades, nodos influyentes y recomendaciones de targeting basadas en embeddings o medidas de similitud.

Duración y estructura (15 horas)

- Bloque A: Fundamentos de series temporales y Prophet (3 h)

- Bloque B: BSTS y modelos bayesianos de series (3 h)
- Bloque C: Marketing Mix Modeling (Bayesiano) y optimización de presupuesto (3.5 h)
- Bloque D: Diseño de experimentos y atribución/incrementalidad (2.5 h)
- Bloque E: Graph Intelligence aplicada al marketing (3 h)

Cada bloque incluye: microclase (20 a 40 minutos), taller guiado (60 a 90 minutos), reto autónomo (20 a 45 minutos) y checklist de evidencia.

Recursos y setup (prioridad)

Requisitos mínimos:

- Python 3.9+ y librerías: pandas, numpy, prophet, pymc3 o pymc, scikit-learn, networkx, python-igraph (opcional), node2vec o stellargraph, sentence-transformers, matplotlib/plotly, arviz.
- Entorno de notebooks: JupyterLab o Google Colab.
- Dataset sugerido: 1,000 a 5,000 filas con columnas date, channel, spend, impressions, clicks, conversions, revenue, geo, product_id, customer_id.
- Opcional: Neo4j Desktop o servicios vectoriales para grafos y embeddings.

Sandboxes y alternativas low-cost:

- Datasets sintéticos o sample datasets de Kaggle/UCI adaptados.
- Plantillas preconfiguradas para ingestión y preprocesado.

Bloque A — Series temporales aplicadas: Prophet (3 h)

Objetivo: Entender y aplicar Prophet para series con estacionalidad, festividades y efectos promocionales.

Contenido clave:

- Conceptos: tendencia, estacionalidad, multiplicativo vs aditivo, holidays/events, changepoints.
- Prophet: parámetros relevantes (changepoint_prior_scale, seasonality_mode, yearly/weekly/daily).
- Evaluación: holdout rolling window, métricas MAE, RMSE, MAPE y coverage de intervalos predictivos.

Taller guiado A1 (90 min):

1. Notebook: carga y preprocesamiento de datos (resample diario/semana según granularidad).

2. Construcción de modelo Prophet: agregar holidays y regressors (gasto por canal).
3. Forecast y backtest con rolling origin (3 folds) y cálculo de métricas.

Reto autónomo A2 (30 a 60 min):

- Incorporar efectos de eventos (promociones) y comparar con modelo base; documentar ganancias en cobertura o reducción de error.

Checklist A:

- Notebook A_prophet.ipynb con modelo, forecast y métricas.
- Gráfica de predicción vs real con intervalos.
- Export de resultados en CSV y screenshot de evidencia.

Bloque B — BSTS y modelos bayesianos de series (3 h)

Objetivo: Aplicar modelos bayesianos estructurales para forecasting, estimación de incertidumbre y contrafactuales.

Contenido clave:

- BSTS: componentes (trend, seasonal, local level, regression con coeficientes time-varying).
- Ventajas: incertidumbre bayesiana y estimación de efectos contrafactuales.
- Herramientas: PyMC/PyBATS en Python o bsts en R; ArviZ para diagnóstico.

Taller guiado B1 (90 a 120 min):

1. Notebook: especificación de un modelo BSTS con componentes y regresores.
2. Posterior sampling (MCMC) y diagnóstico: traceplots, R-hat, ESS.
3. Forecast y construcción de contrafactuales para evaluar efecto de campaña.

Reto autónomo B2 (30 a 45 min):

- Estimar el efecto contrafactual de una campaña específica y comparar con diff-in-diff.

Checklist B:

- Notebook B_bsts.ipynb con modelado bayesiano, diagnósticos y contrafactual.
- Reporte corto (1 a 2 páginas) con interpretación de incertidumbres.

Bloque C — Marketing Mix Modeling (Bayesiano) y optimización (3.5 h)

Objetivo: Construir un MMM bayesiano que estime elasticidades por canal y proponga redistribución presupuestal optimizada.

Contenido clave:

- Formulación del MMM: respuesta como función del gasto por canal con saturación (adstock, S-shaped), control variables.

- Estimación bayesiana: priors informados, MCMC y análisis de posterior.
- Optimización: asignación de presupuesto para maximizar KPI (ventas, ROAS) bajo constraints.

Taller guiado C1 (120 a 150 min):

1. Notebook: preparar datos agregados por semana (spend per channel, sales, control vars).
2. Implementar adstock transformation y parametrización.
3. Especificar modelo bayesiano con PyMC y ejecutar MCMC; extraer elasticidades.
4. Simular escenarios y formular optimización (scipy.optimize o CVXPy).

Reto autónomo C2 (30 a 45 min):

- Diseñar tres escenarios de redistribución presupuestal y reportar impacto esperado y bandas de incertidumbre.

Checklist C:

- Notebook C_mmm.ipynb con modelo, posteriors y simulaciones.
- Script optimize_budget.py que use summaries de posterior y retorne asignación óptima.
- PDF ejecutivo con recomendaciones y estimación de impacto.

Bloque D — Atribución e incrementalidad: A/B y geo-lift (2.5 h)

Objetivo: Diseñar experimentos causales robustos y calcular incrementalidad con medidas de confianza.

Contenido clave:

- Diseño de experimentos: randomización, unidad de asignación, covariate balance, evitar spillover.
- Geo-lift: diseño, validación y análisis, control de confounders espaciales y temporales.
- Cálculo de tamaño muestral y potencia.
- Análisis: estimador de efecto incremental, CI y robustness checks.

Taller guiado D1 (90 min):

1. Notebook: función de cálculo de sample size y ejemplo práctico.
2. Simulación de A/B y geo experiments en datos sintéticos; cálculo de ATE y uplift con bootstrap CI.
3. Implementar checks: tablas de balance, pre-trend plots, permutation tests.

Reto autónomo D2 (30 a 45 min):

- Proponer diseño para un test geo real: seleccionar geos, justificar asignación y calcular MDE.

Checklist D:

- Notebook D_experiments.ipynb con scripts de sample size, análisis y robustness.
- Plan experimental de 1 página para caso real del participante.

Bloque E — Graph Intelligence aplicada al marketing (3 h)

Objetivo: Usar grafos para identificar comunidades, influencia y similitud, y traducir resultados a acciones de targeting y contenido.

Contenido clave:

- Representación de grafos: nodos (clientes, productos, contenidos), aristas (co-purchase, interacciones, similitud).
- Algoritmos: detección de comunidades (Louvain, Leiden), medidas de centralidad (PageRank, betweenness), embeddings (node2vec, GraphSAGE).
- Aplicaciones: lookalike audiences, influencer detection, product affinity, churn propagation.

Taller guiado E1 (90 min):

1. Notebook: construir grafo con NetworkX o Neo4j usando interacciones user-product.
2. Ejecutar detección de comunidades, calcular centralities y visualizar subgrafo.
3. Generar embeddings con node2vec y calcular nearest neighbors para lookalike.

Reto autónomo E2 (30 a 45 min):

- Crear segmentación basada en comunidades y proponer tres acciones de marketing (campaign targeting, content seeding, influencer outreach).

Checklist E:

- Notebook E_graphs.ipynb con grafo, comunidades, centralities y embeddings.
- CSV con segmentos y top influencers, más un action plan breve.

Entregables finales de la unidad (por participante)

1. ZIP con notebooks: A_prophet.ipynb, B_bsts.ipynb, C_mmm.ipynb, D_experiments.ipynb, E_graphs.ipynb.
2. Scripts: optimize_budget.py, detect_effects.py, compute_sample_size.py.
3. PDF ejecutivo (máx. 2 páginas) con insights accionables y recomendaciones presupuestarias basadas en MMM.

4. Dashboard o mockup (screenshot) con visualización de forecast, elasticities y segments.
5. Rúbrica completada y checklist de evidencia.

Rúbrica de evaluación (100 puntos)

- Correctitud técnica de modelos (Prophet, BSTS, MMM): 30 puntos
- Calidad del análisis causal y diseño experimental: 20 puntos
- Uso de Graph Intelligence y traducción a acciones: 15 puntos
- Reproducibilidad (notebooks, scripts, tests): 15 puntos
- Presentación ejecutiva y recomendaciones de negocio: 20 puntos

Aprobación mínima: 60 puntos de 100.

Función básica para cálculo de tamaño de muestra
(Incluir como compute_sample_size.py con fórmulas estándar para MDE y potencia)

Pruebas de aceptación (mínimas)

1. Forecasting test: ejecutar A_prophet.ipynb — generar forecast.csv y metrics.json (MAE, RMSE).
2. MMM test: ejecutar C_mmm.ipynb — generar posterior_elasticities.csv y optimize_budget.py produce allocation.csv.
3. Experiment analysis test: ejecutar D_experiments.ipynb — generar experiment_report.pdf con ATE y CI.
4. Graph test: ejecutar E_graphs.ipynb — generar segments.csv y top_influencers.csv.

Si todas las pruebas generan los artefactos listados, resultado: APROBADO.

Riesgos y mitigaciones

- Datos insuficientes o ruidosos: usar agregación (weekly), regularización bayesiana y simulaciones para estimar incertidumbre.
- Heterogeneidad técnica en estudiantes: proporcionar notebooks paso a paso y dataset sandbox.
- Dependencia de APIs externas: ofrecer datasets sintéticos y mocks.
- Interpretación errónea de causalidad: insistir en checks de robustez (pre-trends, placebos).

Recomendaciones pedagógicas

- Proveer prework (lectura y 2 notebooks básicos sobre pandas y series temporales) para homogeneizar nivel.

- Entrega intermedia: solicitar draft de proposal y subset de notebooks a mitad del módulo para feedback.
- Sesión final de revisión ejecutiva: cada participante presenta el PDF ejecutivo en 5 a 7 minutos.
- Facilitar office hours técnicos para MCMC y debugging de modelos bayesianos.

Unidad III: Agentes Autónomos y Sistemas en Tiempo Real (25 horas)

Resumen ejecutivo

Unidad de 25 horas orientada a la implementación práctica de agentes autónomos, arquitecturas en tiempo real e inteligencia artificial multimodal aplicadas al marketing y a operaciones. Los participantes desarrollarán un sistema funcional compuesto por agentes generativos conectados a datos, modelos y acciones automatizadas. Se abordan arquitecturas RAG con LLMs finamente ajustados, generación de contenido operativa, uso de Feature Stores online, procesamiento en streaming y toma de decisión en milisegundos. El entregable final es un prototipo ejecutable de agente autónomo que toma decisiones o genera acciones basadas en datos en tiempo real, acompañado de su dashboard de monitoreo y métricas de desempeño.

Objetivos de la unidad

- Comprender los principios y arquitectura de los sistemas autónomos aplicados al marketing.
- Diseñar y construir agentes generativos conectados a datos y modelos en tiempo real.
- Aplicar RAG multimodal para integrar texto, imagen y audio en la toma de decisiones.
- Implementar mecanismos de inferencia y decisión online con Feature Stores y procesamiento de streams.
- Documentar la orquestación completa: pipeline, agentes, control y monitoreo.

Resultados de aprendizaje (medibles)

Al finalizar la unidad, el participante podrá:

1. Implementar un agente autónomo que procese inputs en tiempo real y ejecute acciones basadas en datos.
2. Integrar modelos generativos (texto o imagen) en un flujo operativo con RAG multimodal.
3. Desarrollar un pipeline de decisiones con Feature Store online y arquitectura Lambda o Kappa.
4. Diseñar un dashboard de monitoreo que visualice decisiones, latencia y desempeño del agente.
5. Entregar un paquete reproducible (notebooks, scripts, flujo n8n/LLM) con métricas de respuesta y logs de decisiones.

Duración y estructura (25 horas)

- Bloque A: Arquitectura de agentes autónomos y visión general (3 h)
- Bloque B: Agentes generativos y RAG multimodal (6 h)
- Bloque C: Decisiones en tiempo real y Feature Stores online (6 h)
- Bloque D: Computer Vision y análisis multimodal aplicado (5 h)
- Bloque E: Monitoreo, despliegue y gobernanza del sistema (5 h)

Cada bloque incluye: microclase (30 a 45 minutos), taller guiado (90 a 120 minutos), reto autónomo (30 a 60 minutos) y checklist de evidencia.

Recursos y setup (prioridad)

Requisitos mínimos:

- Python 3.10 o superior; librerías recomendadas: langchain, openai o transformers, fastapi, feast (Feature Store), streamlit, kafka o equivalente simulado.
- n8n o Copilot Studio para orquestación de agentes.
- Dataset mínimo: logs o interacciones de clientes (fecha, canal, input, respuesta, feedback).
- Acceso a API de LLM (OpenAI, Azure o alternativa local).
- GPU opcional para tareas multimodales (CLIP, BLIP, Whisper).

Sandboxes y alternativas low-cost:

- OpenAI Free Tier o modelos locales ligeros.
- Feast local como Feature Store de prueba.
- Simulación de streaming con asyncio o colas en memoria.
- Mocks de API para pruebas de integración.

Bloque A — Arquitectura de agentes autónomos y visión general (3 h)

Objetivo: Comprender los componentes de un sistema autónomo y su ciclo percepción-decisión-acción.

Contenido clave:

- Anatomía del agente autónomo: sensores, actuadores, razonamiento.
- Orquestación con flujos n8n y frameworks tipo LangChain.
- Comunicación entre agentes: mensajes, colas y protocolos.
- Casos de uso en marketing: segmentación automática, pricing dinámico, respuesta automatizada al cliente.

Taller guiado A1 (90 min):

- Diseñar diagrama de arquitectura de un agente que recibe inputs y genera outputs accionables.

Reto autónomo A2 (30 a 45 min):

- Crear flujo conceptual de un agente autónomo aplicado a un caso real de marketing.

Checklist A:

- Diagrama de arquitectura con roles definidos.
- Documento explicativo del flujo percepción-acción.

Bloque B — Agentes generativos y RAG multimodal (6 h)

Objetivo: Integrar modelos generativos y sistemas RAG multimodales en agentes autónomos.

Contenido clave:

- Arquitectura RAG: retrieval, augmentation y generation.
- Integración con bases vectoriales (Weaviate, Pinecone o FAISS).
- Fine-tuning ligero de LLMs y prompt engineering para tareas específicas.
- RAG multimodal: combinar texto, imagen y metadatos.
- Casos prácticos: generación autónoma de copy, resumen de conversaciones, enriquecimiento de contexto.

Taller guiado B1 (120 min):

- Construir pipeline RAG con embeddings vectoriales y generación contextualizada.

Reto autónomo B2 (60 min):

- Integrar un modelo de generación (copy o respuesta a cliente) con contexto RAG.

Checklist B:

- Notebook rag_agent.ipynb ejecutable.
- Evidencia de generación multimodal (texto y, si aplica, imagen o audio).

Bloque C — Decisiones en tiempo real y Feature Stores online (6 h)

Objetivo: Implementar decisiones autónomas en tiempo real basadas en features recientes y modelos de baja latencia.

Contenido clave:

- Arquitecturas Lambda y Kappa: diferencias y casos de uso.
- Feature Store online (Feast u alternativa): definición, actualización y consumo de features.
- Procesamiento de streams: Kafka, Pub/Sub o simulación con asyncio.

- Micro-models para decisiones hipercontextuales.
- Integración con agentes vía APIs y triggers.

Taller guiado C1 (120 min):

- Configurar pipeline de ingestión en tiempo real y simular toma de decisión autónoma.

Reto autónomo C2 (60 min):

- Crear flujo de actualización de features en tiempo real y registrar decisiones del agente.

Checklist C:

- Notebook realtime_decision.ipynb con flujo completo.
- Logs de decisiones almacenados (JSON o CSV).

Bloque D — Computer Vision y análisis multimodal aplicado (5 h)

Objetivo: Integrar análisis visual y multimodal en agentes generativos y de decisión.

Contenido clave:

- Embeddings multimodales (CLIP, BLIP, OpenAI Vision).
- Tareas: reconocimiento de objetos, análisis de escena, extracción de metadatos visuales.
- Aplicaciones: validación de contenido visual, análisis de engagement, creatividad adaptativa.
- Flujo multimodal: imagen o video → embedding → contexto → acción.

Taller guiado D1 (120 min):

- Implementar pipeline visual: detección, embeddings y generación de respuesta textual.

Reto autónomo D2 (45 a 60 min):

- Crear demo de agente visual que describa una imagen o evalúe contenido publicitario.

Checklist D:

- Notebook vision_agent.ipynb ejecutable.
- Evidencia: imagen + texto generado o clasificación.

Bloque E — Monitoreo, despliegue y gobernanza del sistema (5 h)

Objetivo: Monitorear y gobernar un sistema de agentes autónomos en producción.

Contenido clave:

- Métricas clave: latencia, precisión, cobertura, tasa de éxito y tasa de error.
- Dashboards de monitoreo: Streamlit, Grafana o Power BI.
- Registro de decisiones y explicabilidad básica (logs y resúmenes SHAP).

- Gobernanza: políticas de permisos, revisión humana y consideraciones éticas.

Taller guiado E1 (90 a 120 min):

- Construir dashboard con métricas de performance y registro de eventos.

Reto autónomo E2 (30 a 45 min):

- Incorporar trazabilidad (logging) y resumen explicativo de decisiones recientes.

Checklist E:

- Dashboard ejecutable (Streamlit o notebook).
- Registro de logs y métricas básicas disponibles.

Entregables finales de la unidad (ZIP)

- Notebooks: rag_agent.ipynb, realtime_decision.ipynb, vision_agent.ipynb.
- Logs de decisiones y métricas en CSV o JSON.
- Dashboard o mockup de monitoreo (Streamlit o Power BI).
- Documento técnico-ejecutivo (2 a 3 páginas) con arquitectura, resultados y reflexiones éticas.

Rúbrica de evaluación (100 puntos)

- Diseño y arquitectura de agentes autónomos: 20 puntos
- Implementación técnica (RAG, Feature Store, Vision): 30 puntos
- Capacidad de decisión en tiempo real y orquestación: 20 puntos
- Monitoreo, métricas y reproducibilidad: 15 puntos
- Documentación y presentación ejecutiva: 15 puntos

Aprobación mínima: 60 puntos de 100.

Pruebas de aceptación (mínimas)

- rag_agent.ipynb ejecuta consulta contextualizada con éxito.
- realtime_decision.ipynb produce decisiones registradas en log.
- vision_agent.ipynb genera descripción o clasificación visual válida.
- Dashboard presenta métricas actualizadas.

Si todas las pruebas se ejecutan y producen los artefactos listados, resultado: APROBADO.

Riesgos y mitigaciones

- Saturación de API o costos elevados: Mitigación: cache local de embeddings y límites de consultas.

- Latencia en respuestas de modelos: Mitigación: modelos ligeros, asincronismo o micro-models.
- Errores en Feature Store: Mitigación: actualización incremental, validaciones y logs.
- Decisiones erróneas sin control humano: Mitigación: supervisión, umbrales de seguridad y revisión ética.

Recomendaciones pedagógicas

- Dividir equipos por tipo de agente (generativo, visual, decisional).
- Proveer ejemplos base de pipelines RAG y streaming para adaptar.
- Evaluar mediante demo práctica en vivo y revisión del dashboard.
- Promover discusión final sobre límites de autonomía y transparencia.

Unidad IV: Sistemas Multiagente, Reinforcement Learning y Optimización Autónoma (20 horas)

Resumen ejecutivo

Unidad técnica-práctica de 20 horas orientada a diseñar, simular y desplegar arquitecturas multiagente aplicadas al marketing. Se trabajan roles de agentes (Monitor, Decisor, Executor), orquestación con frameworks como LangChain o AutoGen, técnicas de personalización dinámica mediante Multi-Armed Bandits contextuales y RL ligero, y optimización multiobjetivo con análisis de Frontera de Pareto. El objetivo es entregar dos casos de uso empresariales reproducibles por participante o equipo: un simulador de campañas con agentes optimizadores y un orquestador de journeys que demuestre mejora en KPI simulados.

Objetivos de la unidad

- Diseñar una arquitectura multiagente con roles, protocolos de comunicación y registro de agentes (Agent Registry).
- Implementar y evaluar agentes especializados (Campaign Optimizer, Budget Allocator, Personalization Agent, Trend Detector) en un entorno simulado o realista.
- Aplicar técnicas de RL y MAB contextuales para optimización dinámica de bidding, presupuesto y contenido.
- Resolver problemas multiobjetivo y presentar soluciones en la Frontera de Pareto para decisiones guiadas por trade-offs.
- Entregar artefactos reproducibles: simulador, agent configs, workflows de orquestación (n8n o LangChain), reporte técnico y demo.

Resultados de aprendizaje (medibles)

Al completar la unidad el participante será capaz de:

1. Exponer una arquitectura de sistema multiagente (diagrama y justificación) y desplegar un agent registry funcional.
2. Implementar al menos dos agentes especializados que interactúen vía message bus y registrar su desempeño en logs trazables.
3. Diseñar e implementar un experimento de Multi-Armed Bandit contextual o RL ligero que mejore un KPI simulado respecto a una política baseline.
4. Producir un análisis multiobjetivo con al menos tres soluciones en la Frontera de Pareto y justificar la elección operativa.

5. Entregar código reproducible (notebooks y scripts), workflows de orquestación, reporte técnico y screencast demo (≤ 5 min).

Duración y estructura (20 horas)

- Bloque A: Fundamentos y arquitectura multiagente (3.5 h)
- Bloque B: Agent Orchestrator y protocolos (4 h)
- Bloque C: Reinforcement Learning y Multi-Armed Bandits contextuales (5 h)
- Bloque D: Optimización multiobjetivo y Pareto (3.5 h)
- Bloque E: Casos prácticos: Dynamic Pricing, Campaign Optimization, Journey Orchestrator (4 h)

Cada bloque contiene: microclase (20 a 40 min), taller guiado (60 a 120 min), reto autónomo (20 a 60 min), checklist y evidencia requerida.

Recursos y setup (prioridad)

Software y librerías: Python 3.10+, Docker, Git, JupyterLab o Colab, n8n, LangChain o AutoGen, PettingZoo o entornos similares, Stable-Baselines3 o Ray RLlib, Gymnasium, scikit-learn, pandas, numpy, plotly o matplotlib.
Opcionales: Neo4j para mapping, Redis o Kafka como message bus, PostgreSQL para persistencia, MLflow para tracking, Pinecone o FAISS para retrieval.
Datasets y simuladores: dataset sintético de campañas (impressions, bids, clicks, conversions, cost, product, geo) o export real anonimizado; simulador de mercado provisto en plantillas.
Sandboxes: Docker + local runners; mocks de Ads APIs y métricas si no hay acceso cloud.

Bloque A — Fundamentos y arquitectura multiagente (3.5 h)

Objetivo: Comprender paradigmas multiagente y diseñar la arquitectura del sistema: agent registry, message bus, estado compartido y observabilidad.

Temas clave:

- Tipos de agentes: monitor, decisor, executor, compliance checker.
- Protocolos: comunicación asíncrona, eventos, solicitudes/respuestas, contratos JSON schema.
- Estado compartido: snapshot, locks, convergencia eventual, consistencia requerida.
- Observabilidad: logs estructurados, traceId, métricas de latencia y throughput por agente.

Taller guiado A1 (90 a 120 min):

1. Diseñar diagrama de arquitectura (draw.io) con Agent Registry, Message Bus, Feature Store y Orchestrator.
2. Implementar un Agent Registry simple (servicio REST) y exponer endpoints de registro y listado.

Reto autónomo A2 (30 a 45 min):

- Registrar 3 agentes simulados y comprobar descubrimiento mediante GET /agents.

Checklist A:

- Diagramas exportados.
- Agent Registry corriendo localmente y endpoints probados.
- Log básico con traceId implementado.

Snippet (Agent Registry - Flask pseudocode)

```
from flask import Flask, request, jsonify
app = Flask(__name__)
registry = {}
    @app.route('/register', methods=['POST'])
def register():
data = request.json
registry[data['agent_id']] = data
return jsonify({'status':'ok'})
    @app.route('/agents')
def list_agents():
return jsonify(list(registry.values()))
```

Bloque B — Agent Orchestrator y protocolos (4 h)

Objetivo: Construir un orquestador que defina flujos visuales o en JSON, gestione condiciones y fallback, y registre ejecuciones con rollback y compensaciones.

Temas clave:

- Visual workflow: nodos = agentes, edges = triggers/conditions.
- Resolución de conflictos y priorización (p. ej. ROAS vs Engagement).
- Rollback y compensating actions para transacciones largas.
- Integración low-code para edición de workflows por usuarios de negocio.

Taller guiado B1 (120 min):

1. Construir workflow en n8n o como JSON: DataCollectorAgent -> CampaignOptimizerAgent -> ExecutorAgent.
2. Implementar condiciones: si predicted_ROAS < threshold -> marcar para revisión humana; else -> ejecutar.
3. Test de rollback: simular fallo en ExecutorAgent y ejecutar compensating action.

Reto autónomo B2 (30 a 45 min):

- Definir regla de resolución multiobjetivo: priorizar ROAS cuando budget_util > 80%, priorizar Engagement si < 80%.

Checklist B:

- Workflow importable en n8n o JSON válido.
- Tests que simulan fallo y demuestran rollback.
- Documentación de la regla de resolución.

Snippet (Flow JSON conceptual)

```
{
"nodes": [
{"id":"collector","type":"agent","config":{}},
{"id":"optimizer","type":"agent","config":{"policy":"bandit"}},
{"id":"executor","type":"agent","config":{}}
],
"edges":[{"from":"collector","to":"optimizer"},{"from":"optimizer","to":"executor"}]
}
```

Bloque C — Reinforcement Learning y Multi-Armed Bandits contextuales (5 h)

Objetivo: Implementar y evaluar MABs contextuales y RL ligero para tareas de bidding, allocation y content selection.

Temas clave:

- MAB vs RL: criterios de elección. Contextual bandits para personalización; RL para políticas secuenciales.
- Algoritmos: Epsilon-greedy, UCB, Thompson Sampling, LinUCB; para RL: DQN, PPO, A2C ligeros.
- Reward design y safety: shaping, clipping, constraints.

- Evaluación: cumulative reward, uplift vs baseline, regret, evaluación offline con replay buffer.

Taller guiado C1 (120 min) - Contextual Bandit:

1. Desplegar simulador que dados (context, action) devuelve reward simulado.
2. Implementar LinUCB y Thompson Sampling con features contextuales.
3. Evaluación offline: IPS y Doubly Robust estimators.

Taller guiado C2 (120 min) - RL ligero:

1. Configurar entorno con Stable-Baselines3 o wrapper simple.
2. Entrenar política con episodios cortos y evaluar estabilidad.
3. Incluir safety checks que respeten constraints de presupuesto.

Reto autónomo C3 (30 a 45 min):

- Ejecutar una política entrenada vs baseline random y reportar cumulative reward y uplift.

Checklist C:

- Implementación de bandit en notebook.
- Training logs y evaluation plots.
- Script de evaluación offline para logged data.

Snippet (LinUCB sketch)

```
A = {a: np.eye(d) for a in actions}
b = {a: np.zeros(d) for a in actions}
for t, (x, reward) in enumerate(data):
p = {}
for a in actions:
theta = np.linalg.inv(A[a]).dot(b[a])
p[a] = theta.dot(x) + alpha * np.sqrt(x.T.dot(np.linalg.inv(A[a])).dot(x))
chosen = argmax(p)
A[chosen] += np.outer(x,x)
b[chosen] += reward * x
```

Bloque D — Optimización multiobjetivo y Frontera de Pareto (3.5 h)

Objetivo: Formular y resolver problemas multiobjetivo y presentar soluciones eficientes en la Frontera de Pareto.

Temas clave:

- Formulación: objectives, constraints, action space.
- Técnicas: scalarization, epsilon-constraint, NSGA-II, Bayesian multiobjective.
- Visualización: Pareto front plotting, trade-off curves, sensitivity analysis.

Taller guiado D1 (90 a 120 min):

- Definir objetivos y constraints.
- Resolver con NSGA-II o weighted sum.
- Visualizar front y seleccionar puntos operativos.

Reto autónomo D2 (30 a 45 min):

- Producir tres asignaciones candidatas desde el Pareto set y justificar según prioridades de negocio.

Checklist D:

- Script multiobjective_opt.py que outpute Pareto set.
- Plots de trade-offs y documento de justificación.

Bloque E — Casos prácticos: Dynamic Pricing, Campaign Optimization, Journey Orchestrator (4 h)

Objetivo: Integrar componentes A-D en casos reales reproducibles.

Caso 1: Dynamic Pricing (120 min)

- Componentes: Trend Detector Agent, Pricing Decisor (RL), Executor Agent, Monitor.
- Deliverables: simulador, política entrenada o heurística, demo de ajustes de precio y KPI plots.

Caso 2: Campaign Optimization (120 min)

- Componentes: Data Collector, Campaign Optimizer (MAB/RL), Budget Allocator, Executor.
- Deliverables: workflow (n8n or JSON), evaluación de política vs baseline, dashboard de KPIs.

Caso 3: Customer Journey Orchestrator (opcional)

- Orchestrator que aplica NextBestAction por usuario (contextual bandit).
- Deliverables: sample flow, simulación de journeys, métricas de lift.

Checklist E:

- Notebooks/scripts, policy artifacts, y screencast ≤ 5 min.
- README con pasos para reproducir localmente (Docker Compose).

Entregables finales (por participante o equipo)

1. ZIP con: notebooks (A_architecture.ipynb, B_orchestrator.ipynb, C_bandits_rl.ipynb, D_multiobjective.ipynb, E_cases.ipynb), scripts (agent_registry.py, simulator.py, train_policy.py, multiobjective_opt.py), y workflows (workflow_campaign.json).
2. Diagrama de arquitectura (PNG/PDF), README reproducible y requirements.txt.
3. Reporte técnico (≤ 3 páginas) con diseño, experimentos y resultados (incluye Pareto analysis).
4. Demo screencast (≤ 5 min) y checklist completado.

Rúbrica de evaluación (100 puntos)

- Arquitectura y diseño multiagente: 20 puntos
- Implementación y reproducibilidad de agentes: 20 puntos
- Calidad del experimento RL/bandit y mejora sobre baseline: 20 puntos
- Resolución multiobjetivo y análisis Pareto: 15 puntos
- Documentación, demo y entrega ejecutiva: 25 puntos

Aprobación mínima: 60 puntos.

Pruebas de aceptación (mínimas y automáticas)

1. Registry test: POST/GET a Agent Registry endpoints debe funcionar.
2. Workflow import test: workflow_campaign.json debe importarse o parsearse y validar esquema.
3. Simulation test: ejecutar simulator.py --scenario campaign debe generar CSV con rewards y policy_evaluation.json.
4. Bandit performance: la policy entrenada debe superar baseline random en cumulative reward tras N steps.
5. Pareto test: multiobjective_opt.py debe generar al menos 10 soluciones candidatas y un plot del pareto front.

Si todos los tests pasan, resultado: APROBADO.

Mapeo de requisitos funcionales clave (RF -> Unidad IV)

- RF-016 a RF-035: Agent Registry, message bus, state manager, agent health (Bloques A y B)
- RF-021 a RF-030: Agentes especializados (Data Collector, Trend Detector, Campaign Optimizer, etc.) (Bloques B y E)

- RF-031 a RF-035: Orquestador y workflow visual (Bloque B)
- RF-081 a RF-085: RL para bidding, content y budget (Bloque C y E)
- RF-061 a RF-075: Perfilado y decisioning engine (Bloque C y E)
- RF-086 a RF-095: Monitoring, KPIs y model management transversales.

Riesgos y mitigaciones (específicos)

- Simulación no representativa: proveer varios escenarios y parámetros; comparar contra datos históricos si existen.
- Exploding action space: discretizar acciones o usar políticas jerárquicas.
- Seguridad de políticas: aplicar constraints hard y shadow testing.
- Complejidad pedagógica: ofrecer plantillas, step-by-step y dividir en tracks si hay grupos mixtos.

Recomendaciones pedagógicas

- Prework obligatorio: Python básico, probabilidad y conceptos de RL (3 a 4 horas).
- Pair programming: equipos mixtos (marketing + analytics).
- Sesión de debugging en vivo (1.5 a 2 h) para integración n8n + simulator + message bus.
- Checkpoint intermedio: presentar diagrama de arquitectura y proof-of-concept mínimo.

Unidad V: MLOps, Ética (XAI) y Despliegue a Escala (10 horas)

Resumen ejecutivo

Unidad práctica de 10 horas centrada en operacionalizar modelos y agentes con controles de gobernanza y ética. Cubre feature stores, registro y tracking con MLflow (o alternativa gestionada), despliegue en la nube (por ejemplo Vertex AI o SageMaker, y alternativas local/low-cost), monitorización continua (drift y data quality), y explicabilidad práctica con SHAP/LIME para reportes a stakeholders y cumplimiento. Incluye ejercicios hands-on, plantillas de CI/CD y artefactos de auditoría.

Objetivos de la unidad

- Implementar un pipeline de MLOps reproducible que incluya tracking de experimentos, model registry y deployment a staging (Vertex AI/SageMaker o Docker/Kubernetes).
- Instrumentar monitorización operativa y de modelo: latency, error rate, confidence distribution y detector de drift.
- Aplicar técnicas de XAI (SHAP) para generar explicaciones locales y globales reproducibles y entregables para compliance.
- Diseñar y ejecutar una política de detección de sesgos y fairness tests para modelos de segmentación o scoring.
- Entregar artefactos de auditoría: model card, datasheet, logs redacted, playbook de incidentes y demo operativo.

Resultados de aprendizaje (medibles)

Al finalizar la unidad, el participante podrá:

1. Presentar un pipeline reproducible (repositorio + CI) que registre experimentos en MLflow (o mock) y despliegue al menos una versión de modelo en staging.
2. Configurar monitorización básica (Prometheus/Grafana o logs + CSV) y un detector de drift que dispare una alerta simulada.
3. Generar explicaciones SHAP para un modelo tabular y producir un resumen ejecutivo comprensible para negocio y reguladores.
4. Ejecutar una suite de fairness checks (disparate impact, equal opportunity) y documentar resultados y mitigaciones.
5. Entregar: model_card.md, audit_pack.zip (logs redacted, scripts), ci.yml y screencast de demostración (≤ 5 min).

Duración y estructura (10 horas)

- Bloque A: Feature Store y Tracking (2.5 h)
- Bloque B: Despliegue en la nube y CI/CD (2.5 h)
- Bloque C: Monitoring, Drift y Data Quality (2 h)
- Bloque D: Explainable AI (SHAP) y fairness tests (2 h)
- Bloque E: Entrega final, model card y playbook (1 h)

Cada bloque incluye: micro-lección (20 a 30 min), taller guiado práctico y checklist de evidencia.

Recursos y setup (prioridad)

- Python 3.9+, MLflow (o MLflow server mock), Docker, GitHub Actions, SDKs de cloud (gcloud/azure/aws) o alternativas locales.
- Prometheus + Grafana o alternativas simples: CSV logger + Google Sheets.
- SHAP, scikit-learn, pandas, numpy.
- Dataset tabular anonimizado o sintético compatible con unidades previas.
- Si no hay cloud, usar Docker Compose para orquestar MLflow local, servicio de inferencia y un simple endpoint.

Bloque A — Feature Store y Tracking (2.5 h)

Objetivo: Conectar features reproducibles con el ciclo de vida del modelo y controlar experimentos mediante MLflow.

Contenido clave:

- Conceptos de Feature Store: catálogo, metadatos, materialización offline/online y API de consumo.
- MLflow: tracking experiments, artifacts, model registry. Alternativa: DVC + artifact store.
- Buenas prácticas: tagging de runs, guardar dataset hash, seeds y reproducibilidad.

Taller guiado A1 (90 min):

1. Script features/compute_features.py que materializa features en parquet y genera metadata features_vX.json con hash.
2. Levantar MLflow server en Docker (local) y registrar experimento u5_experiment.
3. En train.py, loggear params, metrics, artifacts (model.pkl) y registrar el modelo en el registry.

Checklist A:

- features_v1.parquet y features_v1.json generados.
- MLflow server corriendo y al menos un run registrado.
- Modelo guardado en registry o artifacts folder.

Bloque B — Despliegue en la nube y CI/CD (2.5 h)

Objetivo: Automatizar build/test y despliegue en staging usando pipeline de CI.

Contenido clave:

- CI: GitHub Actions pipeline que ejecuta tests, lint, build image, push y deploy.
- Conceptos de Vertex AI / SageMaker: empaquetado, endpoints, autoscaling. Alternativa: Docker image + K8s deployment.
- Estrategias de despliegue: canary, blue/green y rollback.

Taller guiado B1 (90 min):

1. .github/workflows/ci.yml que ejecuta tests y construye imagen docker.
2. Script deploy_staging.sh que sube imagen a Docker Hub y aplica kubectl o comandos equivalentes de cloud.
3. Simular canary y rollback manual.

Checklist B:

- CI pasa en PR y build disponible como artifact.
- Script deploy_staging.sh probado en entorno mock o local.
- Documentación de rollback/canary.

Bloque C — Monitoring, Drift y Data Quality (2 h)

Objetivo: Instrumentar métricas de producción y detección de drift con alerta básica.

Contenido clave:

- Métricas operativas: request_count, latency p95, error_rate.
- Métricas de modelo: avg_confidence, accuracy, distributional changes.
- Detección de drift: PSI, KL o KS tests; umbral de alerta (ej. PSI > 0.2).
- Data quality checks: missing rate, cardinality, duplicates.

Taller guiado C1 (60 min):

1. Endpoint /metrics: exponer métricas simuladas con Prometheus client o exportar CSV diario.
2. detect_drift.py que compara histogramas baseline vs recent y notifica (print o webhook).

Checklist C:

- /metrics implementado o CSV de métricas presentes.
- detect_drift.py ejecutable y ejemplo de alerta.
- Script de data quality checks incluido.

Bloque D — Explainable AI (SHAP) y Fairness (2 h)

Objetivo: Generar explicaciones locales y globales con SHAP y ejecutar tests de fairness para identificar y mitigar sesgos.

Contenido clave:

- SHAP: explicaciones locales (force_plot) y globales (summary_plot); almacenar como artifacts.
- Métricas de fairness: disparate impact, equal opportunity difference, demographic parity.
- Estrategias de mitigación: reweighting, threshold adjustment, técnicas adversariales.

Taller guiado D1 (90 min):

1. Cargar modelo (sklearn/xgboost), calcular SHAP values y exportar shap_summary.png.
2. Ejecutar fairness checks por grupo protegido y generar fairness_report.json con recomendaciones.

Checklist D:

- shap_summary.png generado y añadido a artifacts.
- fairness_report.json con métricas y recomendaciones.
- Plan de mitigación documentado.

Snippet SHAP (pegable, simplificado)

```
import shap
explainer = shap.TreeExplainer(model)
shap_values = explainer.shap_values(X_sample)
shap.summary_plot(shap_values, X_sample)
```

Bloque E — Entrega final, model card y playbook (1 h)

Objetivo: Empaquetar evidencia, construir model_card.md y audit_pack.zip, y preparar la demo final.

Entregables mínimos:

1. model_card.md que incluya propósito, dataset summary, metrics, limitations y recommended mitigations.

2. audit_pack.zip: logs redacted, fairness_report.json, shap_summary.png, mlflow_run_<id>.json (metadata).
3. ci.yml en repo, deploy_staging.sh, detect_drift.py y README reproducible.
4. Screencast demo ≤ 5 min y PDF ejecutivo ≤ 2 pages.

Checklist E:

- model_card.md finalizado.
- audit_pack.zip generado con artifacts redacted.
- Screencast y PDF subidos.

Pruebas de aceptación (mínimas)

1. MLflow run: al menos un run con params, metrics y artifacts registrados.
2. Deployment: endpoint de staging responde 200 en /health y acepta /predict (mock permitido).
3. SHAP & fairness: shap_summary.png y fairness_report.json presentes en artifacts.
4. Drift detector: detect_drift.py detecta cambio en dataset sintético y emite alerta.

Si todos los puntos cumplen, resultado: APROBADO.

Rúbrica de evaluación (100 puntos)

- Pipeline reproducible + CI/CD + deployment: 30 puntos
- Monitorización y drift detection: 20 puntos
- Explainability y fairness tests: 20 puntos
- Model card + audit pack + evidencia: 20 puntos
- Demo y presentación ejecutiva: 10 puntos

Aprobación: ≥ 70/100.

Riesgos y mitigaciones

- No acceso a cloud: usar MLflow local y Docker Compose; documentar diferencias con Vertex.
- Datos sensibles: exigir redaction y proveer scripts de enmascarado.
- Limitaciones de tiempo: priorizar evidencia mínima viable: un run MLflow + SHAP + detector de drift.
- Falta de experiencia en CI/CD: proporcionar plantilla de GitHub Actions y scripts comentados.

Recomendaciones pedagógicas finales

- Pedir prework sobre MLflow y SHAP.
- Preparar sandbox con MLflow server y Docker Compose para prácticas.
- Dedicar 1 hora a revisión de artefactos antes de la entrega final.
- Recomendar plantilla de model_card.md y ejemplos de audit_pack para homogeneizar entregables.

7 Conclusiones

El desarrollo integral del diplomado permitió consolidar una visión operativa, analítica y estratégica de los sistemas autónomos aplicados al marketing moderno. A partir del planteamiento inicial la necesidad de habilitar procesos escalables, trazables y basados en datos propios los resultados obtenidos confirmaron que la combinación de arquitecturas de datos robustas, agentes inteligentes y metodologías de MLOps constituye un camino viable para la automatización avanzada en entornos reales.

La exploración progresiva de los componentes técnicos fundamentales demostró que los participantes lograron construir soluciones funcionales en cada etapa del ciclo: desde la ingesta y unificación de datos en arquitecturas tipo CDP, pasando por modelos predictivos y causalidad aplicada, hasta llegar al diseño de agentes autónomos en tiempo real y la orquestación multiagente. Las hipótesis planteadas —principalmente la viabilidad de sistemas autónomos para mejorar eficiencia, toma de decisiones y personalización— se validaron a través de los prototipos desarrollados, evidenciando mejoras consistentes en métricas simuladas y casos prácticos.

Asimismo, los resultados confirmaron que el uso de enfoques causales, modelos bayesianos y grafos incrementa significativamente la capacidad de interpretar fenómenos de mercado y optimizar recursos en escenarios complejos. La integración de Reinforcement Learning y Multi-Armed Bandits mostró ser especialmente útil para la asignación dinámica de presupuesto y la personalización, validando su potencial en contextos empresariales donde se requieren decisiones iterativas y adaptativas.

Finalmente, las prácticas de MLOps y las evaluaciones éticas permitieron cerrar el ciclo con una perspectiva crítica y responsable. La incorporación de mecanismos de monitoreo, explicabilidad, control de sesgos y auditoría operativa demostró que no basta con desplegar modelos; es indispensable garantizar que su comportamiento sea transparente, seguro y alineado con los principios regulatorios y organizacionales.

En conjunto, los aprendizajes obtenidos evidencian que es posible diseñar, implementar y monitorear sistemas inteligentes de marketing capaces de operar con autonomía y rigor técnico. El diplomado no sólo ofrece un mapa conceptual actualizado sobre el marketing impulsado por IA, sino un conjunto de herramientas prácticas que fortalecen la capacidad de las organizaciones para innovar, escalar y gobernar tecnologías avanzadas de manera responsable.

Referencias

Pearl, J. (2009). *Causality: Models, Reasoning, and Inference*. Cambridge University Press.

Rubin, D. B. (1974). *Estimating causal effects of treatments in randomized and nonrandomized studies*.

Sutton, R. S., & Barto, A. G. (2018). *Reinforcement Learning: An Introduction* (2nd ed.). MIT Press.

Ribeiro, Singh, S., & Guestrin, C. (2016). "Why should I trust you?" *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (LIME).

Lundberg, S. M., & Lee, S.-I. (2017). "A unified approach to interpreting model predictions." *Advances in Neural Information Processing Systems* (SHAP).

Sculley, D., et al. (2015). "Hidden technical debt in machine learning systems." *Communications of the ACM*.

Provost, F., & Fawcett, T. (2013). *Data Science for Business: What you need to know about data mining and data-analytic thinking*. O'Reilly Media.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The Elements of Statistical Learning*. Springer.