Maestría en

CIENCIA DE DATOS Y MÁQUINAS DE APRENDIZAJE CON MENCIÓN EN INTELIGENCIA ARTIFICIAL

Trabajo previo a la obtención de título de Magister en Ciencia de Datos y Máquinas de Aprendizaje con mención en Inteligencia Artificial.

AUTOR/ES:

Edison Salomon Cadena Serrano

Raquel Johanna Moyano Arias

Norma Mavelin Ati Herdoiza

Félix Alejandro Ruiz Muñoz

TUTOR/ES:

Andrés Roberto Navas Castellanos

Marcela Azucena Venegas Espinosa

TEMA

Optimización del Rendimiento Operativo en una Planta de Inyección de Plásticos mediante Modelos de Regresión

Quito - Ecuador

2026

REINVENTEMOS EL FUTURO

Certificación de autoría

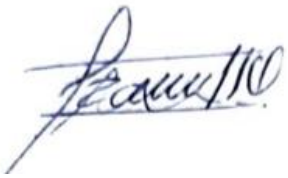
Nosotros, **Edison Salomon Cadena Serrano, Raquel Johanna Moyano Arias, Norma Mavelin Ati Herdoiza y Félix Alejandro Ruiz Muñoz,** declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido presentado anteriormente para ningún grado o calificación profesional y que se ha consultado la bibliografía detallada.

Cedemos nuestros derechos de propiedad intelectual a la Universidad Internacional del Ecuador (UIDE), para que sea publicado y divulgado en internet, según lo establecido en la Ley de Propiedad Intelectual, su reglamento y demás disposiciones legales.

Firma
Edison Salomon Cadena Serrano

Firma
Raquel Johanna Moyano Arias

Firma
Norma Mavelin Ati Herdoiza



Firma
Félix Alejandro Ruiz Muñoz

Autorización de derechos de propiedad intelectual

Nosotros, **Edison Salomon Cadena Serrano, Raquel Johanna Moyano Arias, Norma Mavelin Ati Herdoiza y Félix Alejandro Ruiz Muñoz**, en calidad de autores del trabajo de investigación titulado ***Optimización del Rendimiento Operativo en una Planta de Inyección de Plásticos mediante Modelos de Regresión***, autorizamos a la Universidad Internacional del Ecuador (UIDE) para hacer uso de todos los contenidos que nos pertenecen o de parte de los que contiene esta obra, con fines estrictamente académicos o de investigación. Los derechos que como autores nos corresponden, lo establecido en los artículos 5, 6, 8, 19 y demás pertinentes de la Ley de Propiedad Intelectual y su Reglamento en Ecuador.

D. M. Quito, (06/2026)

Firma
Edison Salomon Cadena Serrano

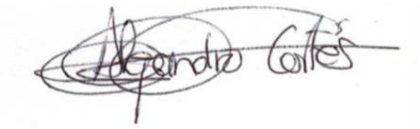
Firma
Raquel Johanna Moyano Arias

Firma
Norma Mavelin Ati Herdoiza

Firma
Félix Alejandro Ruiz Muñoz

Aprobación de dirección y coordinación del programa

Nosotros, **Director Alejandro Cortés López de EIG y Coordinadora Karla Estefanía Mora Cajas de UIDE,** declaramos que: **Edison Salomon Cadena Serrano, Raquel Johanna Moyano Arias, Norma Mavelin Ati Herdoiza y Félix Alejandro Ruiz Muñoz** son los autores exclusivos de la presente investigación y que ésta es original, auténtica y personal de ellos.



Alejandro Cortés López
Director/a de la
Maestría en Ciencia de datos y máquinas de aprendizaje con mención en Inteligencia Artificial

Karla Estefanía Mora Cajas
Coordinador/a de la
Maestría en Ciencia de datos y máquinas de aprendizaje con mención en Inteligencia Artificial

DEDICATORIA

A Ariana Anahí, cuya dulzura lleva el ritmo de mi corazón y a Ariel Alejandro, cuya fuerza me recuerda que todo es posible.

Que este trabajo les recuerde que la constancia y el amor siempre abren puertas. No se trata de llegar rápido, sino de llegar con propósito.

Norma Mavelin Ati Herdoiza

A mi padre, por su apoyo incondicional y por ser mi guía en cada paso.

A mi mujer y a mi hijo, por ser mi motor diario, mi refugio y la razón de este gran esfuerzo.

A mi bisabuela, por su sabiduría, sus bendiciones y por ser el pilar eterno de nuestra familia.

Edison Salomon Cadena Serrano

A mis padres, Rubén y Adelaida, quienes han estado junto a mí en cada etapa de mi vida, brindándome su amor, apoyo incondicional y enseñanzas que han sido fundamentales en mi camino.

A mi hermana Karina y a mis sobrinos, quienes han sido mi soporte y apoyo constante, recordándome que los sueños se alcanzan cuando se trabaja con pasión y compromiso.

Raquel Johanna Moyano Arias.

A Nataly, el amor de mi vida, por sostenerme cuando me faltaban las fuerzas y por no dejarme caer.

A mi hijo Mathías, mi mayor razón para no rendirme y el aliento que me llena cada mañana.

Félix Alejandro Ruiz Muñoz

AGRADECIMIENTOS

A mi familia, pilar fundamental en este proceso. A mis padres Cecilia y Marcelo (+), por su apoyo incondicional y por celebrar cada paso.

Norma Mavelin Ati Herdoiza

A los profesores y a la universidad, por brindarme las herramientas, el espacio y el apoyo constante durante las largas jornadas de trabajo que hicieron posible culminar con éxito este proyecto.

Edison Salomon Cadena Serrano

A mi familia, por acompañarme en cada paso; a mis padres y hermana, por sus consejos y cuidado; y a mis queridos amigos Jairo y Alexis, por estar siempre presentes en este viaje.

Félix Alejandro Ruiz Muñoz

A Dios, por todas las bendiciones en mi vida, por guiarme y enseñarme que con fe, dedicación y perseverancia todo es posible.

A mi familia, por su amor, apoyo incondicional y por ser mi fuerza y motivación en cada paso de este camino.

Raquel Johanna Moyano Arias.

A nosotros: Johanna, Mavelin, Félix, Edison, porque cada palabra de esta tesis tiene el sudor de cuatro frentes, las manos de ocho brazos y el corazón de un solo equipo. No fue fácil, pero fue nuestro y eso lo hace extraordinario. Lo logramos y lo hicimos juntos.

Mavelin, Edison, Johanna y Félix

RESUMEN

El sector industrial de inyección de plásticos requiere mecanismos eficientes para mitigar la falta de automatización en el procesamiento de sus registros de planta y optimizar la toma de decisiones operativas.

El objetivo de este trabajo fue desarrollar e implementar una aplicación web denominada "elaplasml" para el Sistema de Control de Producción de la empresa Elaplas del Ecuador S.A., orientada al registro, monitoreo y análisis predictivo de la manufactura diaria.

La metodología integró una arquitectura de datos tipo *Medallion* progresiva: capas *Bronze* (datos crudos), *Silver* (datos validados) y *Gold* (variables de ingeniería), implementando controles automáticos contra duplicidad. Se entrenaron modelos de Machine Learning (aprendizaje automático) (XGBoost) sobre esta estructura y el final fue desplegado en la plataforma Streamlit Cloud (Streamlit Inc. 2023).

El diseño del sistema se hizo con estrictos controles de acceso para dos roles: un módulo de captura ágil para los operadores y un panel estratégico para la Gerencia de Planta, compuesto por cuatro módulos (Rendimiento, Monitoreo, Histórico y Simulador IA).

Concluyendo, la web efectivamente consolida los seis años de historial operativo y permite a la gerencia realizar análisis de los escenarios condicionales ("What-If") en turnos futuros para poder anticipar la eficiencia esperada de las máquinas y la probabilidad de cumplir el plan, basando la decisión en inteligencia artificial y en patrones históricos reales.

Palabras Claves: Aplicación web, Arquitectura *Medallion*, Control de producción, *Machine Learning*, *Streamlit Cloud*.

ABSTRACT

The plastics injection molding industry requires efficient mechanisms to mitigate the lack of automation in processing its plant records and optimize operational decision-making.

The objective of this work was to develop and implement a web application called "elaplasml" for the Production Control System of Elaplas del Ecuador S.A., designed for recording, monitoring, and predictive analysis of daily manufacturing.

The methodology integrated a progressive Medallion data architecture: Bronze (raw data), Silver (validated data), and Gold (engineering variables) layers, implementing automatic controls against duplication. Machine Learning models (XGBoost) were trained on this structure, and the final version was deployed on the Streamlit Cloud platform (Streamlit Inc. 2023).

The system design incorporated strict access controls for two roles: an agile data capture module for operators and a strategic dashboard for Plant Management, comprised of four modules (Performance, Monitoring, Historical Data, and AI Simulator). In conclusion, the website effectively consolidates six years of operational history and allows management to perform conditional scenario analysis ("What-If") in future shifts to anticipate the expected efficiency of the machines and the probability of meeting the plan, basing the decision on artificial intelligence and real historical patterns.

Keywords: web application, *Medallion* architecture, production control, *machine learning*, predictive analytics

TABLA DE CONTENIDOS

LISTA DE TABLAS (Índice de tablas)

LISTA DE FIGURAS (Índice de figuras)

1. CAPÍTULO 1: INTRODUCCIÓN

1.1. Definición del proyecto

En el marco de la transformación digital promovida por la Industria 4.0, las organizaciones de manufactura integran la analítica de datos y el aprendizaje automático para optimizar sus procesos operativos. En este entorno, la Efectividad Total de los Equipos (OEE) se consolida como la métrica de preferencia para evaluar la eficiencia global en planta. No obstante, las tendencias actuales orientan el uso de este indicador hacia un enfoque predictivo.

Un hito en esta evolución es el trabajo de Dobra y Jósvai (2023), quienes demostraron que los algoritmos basados en *Gradient Boosting* poseen una precisión superior para la estimación del OEE en comparación con la estadística tradicional, reduciendo el margen de error por debajo del 1% en horizontes semanales (Dobra & Jósvai, 2022).

La literatura científica documenta que los algoritmos de aprendizaje por conjuntos (ensemble learning) son capaces de afrontar el comportamiento no lineal de las variables en planta. En este sentido, Kiangala y Wang (2021) concibieron una arquitectura en base a Random Forest y, también, a XGBoost, que otorgan resiliencia operativa a respuesta de la fluctuación de la demanda.

Y, aplicaciones robustas elaboradas por Zhang et al. (2022) a partir de la optimización de la herramienta XGBoost, y unidas a los sistemas de previsión intradía propuestos por Longard et al. (2023), vienen a ratificar la capacidad de estos modelos para poder anticipar desviaciones microoperativas y reducir costos en tiempo real.

A pesar de estos avances, se identifica un vacío significativo en la literatura contemporánea: la mayoría de los casos de estudio se centran en procesos metalúrgicos o ensamblaje automotriz,

existiendo una escasa exploración de modelos predictivos aplicados específicamente a las dinámicas del moldeo por inyección de plástico que vinculen las variables térmicas con el factor humano del turno laboral.

La presente investigación se posiciona justamente en dicho vacío de conocimiento, con el propósito de desarrollar modelos de regresión avanzados (*Random Forest* y *XGBoost*) ajustados a los requerimientos de la planta motivo de este estudio.

En ese contexto, una empresa ecuatoriana fabrica piezas plásticas por inyección, en la actualidad en dicha fábrica, las órdenes de producción se asignan a las máquinas sin un procedimiento fijo. Quien decide es el supervisor, según su experiencia, y ese criterio no está escrito en ninguna parte ni se puede pasar a otra persona. Si el supervisor se va o entra un operador nuevo, ese conocimiento se pierde.

No todas las máquinas responden igual con todos los productos, ya que hay combinaciones máquina-producto que funcionan perfectamente y combinaciones máquina-producto que se convierten, una y otra vez, en máquinas que no paran de parar, que producen poco o que dan lugar a muchos defectos. A pesar de que estas situaciones son las condiciones habituales del día a día en la planta, hasta el momento no se les ha dedicado un análisis exhaustivo para determinar la generación de directrices que ayuden a tomar decisiones optimizadas.

El centro del problema reside en que no existe un sistema analítico basado en datos que permita predecir el rendimiento de una asignación de forma previa a la jornada laboral. En consecuencia, el enfoque de la investigación se centra en calcular:

- La probabilidad de que se produzcan eventos disruptivos (paradas no planificadas) para una combinación máquina-producto-operador-turno.

- Los valores del rendimiento esperado (eficiencia, productividad en piezas/hora, tiempo cíclico) en función de las condiciones de entrada.

- La comparación objetiva para determinar las combinaciones alternativas que permitan localizar la asignación con mayor probabilidad de éxito.

La organización tiene un histórico de registros desagregados por turno en el que se recogen variables como: equipo utilizado, producto fabricado, operador, tiempos planificados frente a los tiempos reales, interrupciones operativas, volúmenes de producción y control de calidad (buenas y malas piezas).

Sin embargo, el mismo análisis de estos datos ha limitado en un enfoque reactivo y retrospectivo. La investigación aquí planteada propone redirigir el uso de este activo de la información en un enfoque predictivo y proactivo del mismo, convirtiéndolo en un mecanismo de soporte para la optimización en la asignación de recursos al inicio de cada jornada operativa.

1.2. Justificación e importancia del trabajo de investigación

El proceso de inyección varía por el desgaste de la máquina, el cambio de turno y la experiencia del operador, y eso hace que la planificación tradicional se quede corta. Algoritmos como *Random Forest* o *XGBoost* son de utilidad porque captan relaciones no lineales entre las variables que no se ven a simple vista.

Predecir bien la eficiencia (%) y la productividad baja el riesgo de sobreestimar lo que la planta puede dar. Eso se traduce en menos tiempo de inactividad, mejor uso de la mano de obra por turno y plazos de entrega más realistas.

1.3.Alcance

El proyecto se desarrollará exclusivamente con datos históricos de una empresa ecuatoriana de fabricación de piezas plásticas por inyección, con registros a nivel de turno de producción.

El alcance comprende las siguientes dimensiones:

1.3.1. Datos y variables

Se analizarán los registros históricos de turnos de producción que incluyen las siguientes variables de entrada y salida:

Tabla 1

Tipos de Variables

Categoría	Variables incluidas
Recurso productivo	Máquina, capacidad nominal, tipo de máquina
Producto	Código de pieza, tipo de plástico, ciclo de producción estimado
Operador y turno	Identificador de operador, turno (mañana/tarde/noche)
Planificación	Horas planificadas, cantidad planificada, materia prima asignada
Resultado operativo	Horas de parada, cantidad real producida, piezas buenas y defectuosas
Indicadores derivados	Eficiencia (%), productividad (piezas/hora), tiempo de ciclo real, tasa de defectos (%)

1.3.2. Modelos a desarrollar

Modelos de regresión para estimar la eficiencia (%) y la productividad (piezas/hora) esperadas en una combinación máquina-producto-operador-turno. Se prueban Regresión Lineal Múltiple, Random Forest Regressor y XGBoost Regressor.

1.3.3. Salida del sistema

El resultado es un apoyo para decidir cómo asignar recursos: dadas varias opciones de asignación para un turno. El sistema evaluará para cada alternativa una métrica del pronóstico integrada, de forma tal que se evalúen tres importantes dimensiones: la probabilidad de interrupciones no planificadas, la estimación de la eficiencia y la productividad esperadas del proceso. Con esos datos, el supervisor contrasta opciones y, a partir de la historia de la planta, selecciona aquella que, por lo que dictan sus experiencias anteriores, debería ser la que mejor rinda.

1.3.4. Limitaciones del alcance

- No se realiza optimización combinatoria automática de cronogramas; el objetivo es poner de manifiesto la decisión del supervisor, no sustituirla.
- Los modelos se entrenan y se validan sólo con datos de la empresa a estudiar; no se quiere generalizar a otras empresas y/o sectores.
- No se realiza el desarrollo de una interfaz de usuario final; los resultados se entregan en formato de informe analítico.
- Las restricciones de secuencia de las operaciones entre máquinas del Job-Shop Scheduling clásico no se consideran.

1.4. Objetivos

1.4.1. Objetivo general

Desarrollar un sistema de soporte a la decisión fundamentado en modelos predictivos para proyectar la eficiencia operativa y la productividad en una planta de inyección de plásticos, con el fin de optimizar la planificación y asignación de recursos por turno.

1.4.2. Objetivos específicos

- Diagnosticar y preprocesar el conjunto de datos históricos de los equipos periféricos y de inyección, mediante técnicas de imputación, normalización e ingeniería de características para la estructuración de las variables críticas del proceso.

- Entrenar y comparar modelos de regresión basados en aprendizaje supervisado (*Random Forest* y *XGBoost*) para la estimación de la eficiencia y productividad, evaluando su rendimiento mediante métricas estadísticas de error (R^2, RMSE, MAE) para la selección del algoritmo óptimo.

- Integrar los modelos predictivos seleccionados en una interfaz funcional, validando su pertinencia operativa y capacidad de respuesta mediante la simulación de escenarios basados en datos históricos de producción.

2. CAPÍTULO 2: REVISIÓN DE LITERATURA

2.1.Estado del arte

En las pequeñas y medianas empresas manufactureras de América Latina, la gestión de la información ha avanzado de forma despareja y más lento que en los grandes corporativos. Durante años, las plantas de transformación de plásticos han funcionado con la experiencia no escrita de los supervisores, hojas de cálculo sueltas y registros llevados a mano. Esto alcanza para anotar lo básico pero repercute en la creación de versiones duplicadas, en la falta de trazabilidad e inevitablemente provoca la acumulación de errores manuales, con la consiguiente dificultad a la hora de hacer cualquier intento serio de analítica o de decidir con base en datos.

En los últimos cinco años el aprendizaje automático en manufactura dejó de ser una ilusión para llegar a conformar una práctica respaldada en la literatura. Existen por lo tanto modelos predictivos que se utilizan para predecir la demanda, para ajustar inventarios y para predecir fallas en líneas de producción en serie. En el moldeo por inyección de plásticos se comprueba que el resto de los autores citados alcanzar a demostrar que los parámetros del proceso (temperatura, presión, tiempo de ciclo) y su relación con la máquina y con el producto tendrían relaciones no lineales que no son capturadas del todo con los métodos estadísticos más clásicos, de ahí que surja el interés por los algoritmos basados en árboles que modelan las relaciones sin necesidad de imponerle una forma funcional fija.

En lo que atañe a la eficiencia operativa, diferentes trabajos han empleado, por ejemplo, Random Forest y XGBoost para estimar indicadores de rendimiento como la Efectividad Total de los Equipos (OEE) y la eficiencia de la planta. Coinciden en que para los modelos de ensamble llegan más lejos que la regresión lineal clásica pero con variables continuas y categóricas

combinadas o cuando el comportamiento de la planta depende de factores que interactúan. Visto que su ventaja no es solo la de predecir mejor: también permiten conocer el peso de cada variable y por qué no decidir.

La literatura también reitera cuánto influye el factor humano y la variabilidad de la máquina en el rendimiento técnico. El turno, la experiencia del operador y el desgaste de la máquina introducen una variabilidad que la planificación clásica suele pasar por alto. Unos pocos autores han encontrado que introducir esas variables como predictores mejora mucho la estimación de la eficiencia y la productividad, porque recoge las diferencias del rendimiento si no se atribuyen al azar. Esto es lo que contrasta y respalda la decisión que se adopta aquí de usar la combinación máquina-producto-operador-turno como unidad de análisis.

Pese a ese progreso externo, en el sector plástico ecuatoriano escasean casi por completo los estudios sobre cómo implementar modelos predictivos verdaderos. La mayoría de empresas de la producción todavía opera con registros desconectados y decide sólo en función de la experiencia, y ahí se presenta un hueco muy interesante para la investigación aplicada que respete las buenas prácticas internacionales, pero que se adapte a los límites de presupuesto y de operación del medio local. Esta investigación va en esa dirección: toma como caso de estudio a una empresa ecuatoriana de piezas plásticas por inyección, y, a partir de sus registros históricos, genera una herramienta para apoyar la asignación de recursos al inicio de cada turno.

2.2.Marco teórico

Para este proyecto se recurre a técnicas, métodos y conceptos vistos a lo largo de la maestría, que sirven de base para implementar modelos predictivos de regresión en Python.

2.2.1. El aprendizaje supervisado

El componente predictivo de este proyecto se sustenta en el aprendizaje supervisado, que se diferencia del no supervisado, puesto que este último se caracteriza por la búsqueda de estructuras ocultas en datos sin etiquetar, mientras que el aprendizaje supervisado tiene como base el uso de ejemplos donde se supone que se conoce la entrada y la salida esperada. Con los pares de ejemplos, el modelo que se utiliza ajusta los parámetros de un modelo de modo que sea capaz de reproducir lo mejor posible la relación existente entre las variables predictoras y su respuesta, finalmente emplea en observaciones nuevas que no conoce por haber formado parte de su entrenamiento.

En el sentido más formal del término, se considera que hay una distribución de probabilidad sobre un espacio de entrada X por un espacio de salida Y, del que únicamente se ha observado una muestra finita de pares de datos. La finalidad, por lo tanto, consiste en encontrar una función f, en un espacio de hipótesis ya dado, que se aproxime razonablemente a la relación existentes entre las variables independientes y aquél del dependiente. Para ello se minimiza una función de pérdida, la que mide el error entre lo que predice el modelo y lo que se observa. Esta elección tiene su importancia: el error cuadrático medio castiga más las grandes desviaciones, mientras que el error absoluto aguanta muy bien los outlayers.

El aprendizaje supervisado cuenta con dos grandes tipos de tarea en función de la variable de interés. Si es categórica, hablamos de clasificación; si continua, de regresión. Este artículo se

inscribe en el segundo caso, ya que son variables las que deseamos estimar, la eficiencia (en porcentaje) y la productividad (en piezas por hora), son continuas. Por este motivo, los modelos de los aptados siguientes son del tipo de regresión.

2.2.2. Metodología CRISP-DM

Se recurre el método CRISP-DM (CRoss-industry Standard Process for Data Mining) como hilo conductor y organizador del trabajo, que está estructurado en 6 fases que se repiten: comprensión del negocio, comprensión de los datos, preparación de los datos, modelado, evaluación e implementación. Lo iterativo es fundamental; cada vez que salta a la vista una limitación que quiebra una de las hipótesis iniciales, el equipo se retrotrae en una fase anterior, en lugar de continuar con una secuencia estrictamente lineal.

Aquí, la comprensión del negocio es el diagnóstico inicial hecho con la empresa de inyección; la comprensión y preparación de los datos son la limpieza y el desarrollo funcional de los registros históricos por turno; el modelado usa los algoritmos de regresión elegidos; la evaluación se hace con las métricas que se describen más adelante; y la implementación es la interfaz para quienes planifican. Que cada fase metodológica calce con un componente técnico es a propósito: busca que la forma de trabajar y lo que se construye vayan de la mano.

2.2.3. Ingeniería de características

La ingeniería de características reúne las transformaciones que se aplican a los datos de entrada para dejar más a la vista la información que le sirve al modelo. Buena parte del éxito de un proyecto supervisado se juega aquí, muchas veces más que en qué algoritmo se elija. En la planta de inyección, esto incluye crear variables derivadas a partir de los registros crudos (tiempo

de ciclo real, tasa de defectos, eficiencia por turno) que no vienen dadas y hay que calcular con las cantidades producidas, las horas ejecutadas y las piezas defectuosas.

Las variables categóricas piden un trato aparte, porque aquí hay identificadores de máquina, operador, producto y turno. Entre las técnicas más comunes están la codificación one-hot, que arma una columna binaria por categoría; la codificación ordinal, útil cuando hay un orden natural; y los enfoques basados en el objetivo, que cambian cada categoría por una estadística condicional. Vale notar que algunos modelos de árboles, como Random Forest y XGBoost, manejan estas variables con más soltura que la regresión lineal, lo cual juega a favor en este problema.

Queda el escalado de las variables numéricas. Los modelos basados en gradientes son sensibles a la escala de las entradas; los de árboles, en cambio, no se inmutan ante transformaciones monótonas. Lo más usado es la estandarización, que centra cada variable en cero y la lleva a varianza unitaria, y la normalización min-max, que aprieta los valores a un rango fijo. Estas operaciones se ajustan solo con el conjunto de entrenamiento y después se aplican a validación y prueba, para no filtrar información.

2.2.4. Compromiso sesgo-varianza y validación

Un resultado teórico muy citado en aprendizaje supervisado descompone el error de predicción en tres partes: sesgo, varianza y error irreducible. El sesgo viene de las suposiciones simplificadoras que el modelo impone sobre la forma real de la función. La varianza mide cuánto cambia el modelo entrenado al darle conjuntos de datos algo distintos. Y el error irreducible es el ruido propio del problema, que ninguna técnica quita. Los modelos muy simples suelen tener sesgo

alto y varianza baja, lo que se llama subajuste; los muy flexibles caen en lo contrario, el sobreajuste.

Para detectar sobreajuste hay que evaluar el modelo con datos distintos a los del entrenamiento. Lo habitual es partir los datos en al menos tres bloques: uno de entrenamiento, donde se ajustan los parámetros; uno de validación, para elegir hiperparámetros y comparar variantes; y uno de prueba, guardado solo para la evaluación final. Dejar que se filtre información del conjunto de prueba durante el modelado arruina las conclusiones sin que se note.

Cuando hay pocos datos, la validación cruzada k-fold rinde más: parte la muestra en k bloques y va rotando cuál se usa para validar. Aquí encaja bien, porque se comparan tres algoritmos y hace falta una estimación confiable del desempeño de cada uno antes de quedarse con el mejor. Así, el procedimiento de validación que se elija pesa mucho, porque marca cuánta confianza se podrá tener al interpretar los resultados.

2.2.5. Regresión lineal simple

La regresión lineal simple, como dice el nombre, es un método lineal que predice una respuesta cuantitativa Y a partir de una única variable predictora X. Se asume la existencia de una relación aproximadamente lineal entre X e Y. Matemáticamente, se escribe la relación lineal como:

$$Y = \beta_0 + \beta_1 X \qquad (1)$$

Los coeficientes β_0 y β_1 son constantes desconocidas las cuales representan los términos de intersección y pendiente en el modelo lineal.

2.2.5.1. Estimación de los coeficientes

En la práctica se utilizan los datos para estimar los coeficientes.

Sea $(x_1, y_1), (x_2, y_2), \dots, (x_n, y_n)$ un conjunto de n pares de observaciones, cada uno de los cuales consta de una medida de X y una medida Y. La idea es estimar los coeficientes para que el modelo lineal se ajuste bien a los datos disponibles

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i, \quad para\ i = 1, \dots, n. \tag{2}$$

En otras palabras, se quiere encontrar una intersección $\hat{\beta}_0$ y una pendiente $\hat{\beta}_1$ tal que la línea resultante sea lo más cercana posible a los n punto de datos. Hay distintas maneras de medir esa cercanía, pero la más usada es el criterio de mínimos cuadrados.

Suma de cuadrados de los residuos

Sea $\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$ la predicción de Y basada en la i-ésimo valor de X. Entonces $e_i = y_i - \hat{y}_i$ representa el i-ésimo residuo. Entonces, se define la suma de cuadrados de los residuos (RSS) como

$$\begin{aligned} RSS &= e_1^2 + e_2^2 + \cdots + e_n^2 \\ &= (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + \cdots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2 \end{aligned} \tag{3}$$

El método de mínimos cuadrados elige $\hat{\beta}_0$ y $\hat{\beta}_1$ para minimizar el RSS. Por lo tanto, los minimizadores son

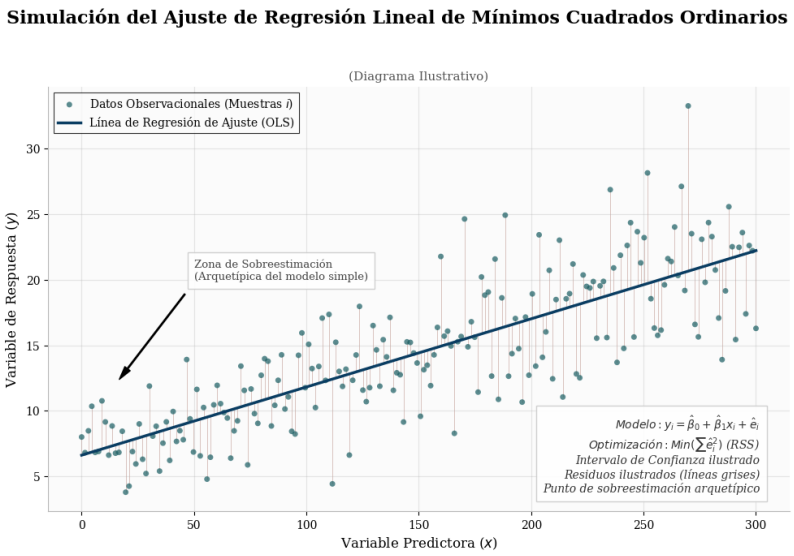
$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{4}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}. \tag{5}$$

Con $\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$ y $\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$ medias muestrales. Donde $\hat{\beta}_1$ define las estimaciones de los coeficientes de mínimos cuadrados para la regresión lineal simple.

Figura 1

Simulación del Ajuste de Regresión Lineal de Mínimos Cuadrados Ordinarios



Nota. Representación gráfica del ajuste por mínimos cuadrados Ordinarios. La recta representa el modelo estimado $y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$, mientras que los segmentos verticales muestran los residuos $e_i = y_i - \hat{y}_i$. El objetivo del modelo es minimizar la suma de los cuadrados de dichos residuos, buscando el mejor ajuste lineal para el conjunto de datos.

2.2.5.2. Evaluación de la precisión del modelo

La calidad del ajuste de una regresión lineal se evalúa típicamente utilizando dos cantidades relacionadas:

1. El error estándar residual (RSE)
2. El estadístico R^2

Residual Standard Error

Sea el modelo

$$Y = \beta_0 + \beta_1 X + \epsilon, \tag{6}$$

Que a cada modelo se le asocia un término de error ϵ. Dado a estos términos de error, incluso si conociéramos la verdadera línea de regresión, no se podría predecir Y a partir de X con precisión. El RSE es una estimación de la desviación estándar de ϵ. En términos generales, es la cantidad promedio en que la respuesta se desviará de la verdadera línea de regresión. Se calcula utilizando la formula

$$RSE = \sqrt{\frac{1}{n-2} RSS} = \sqrt{\frac{1}{n-2} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}. \tag{7}$$

Donde,

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2. \tag{8}$$

El RSE mide qué tan mal se ajusta el modelo a los datos. Si las predicciones quedan muy cerca de los valores reales $\hat{y}_i \approx y_i \; para \; i = 1, \dots, n$. Entonces, RSE será pequeño y podemos concluir que el modelo se ajusta muy bien a los datos.

Estadístico R^2

El RSE proporciona una medida absoluta de la falta de ajuste del modelo $Y = \beta_0 + \beta_1 X + \epsilon$ a los datos. Sin embargo, dado que se mide en unidades de Y, no siempre está

claro qué constituye un buen RSE. El estadístico R^2 da una medida de ajuste distinta. Como es una proporción, queda siempre entre 0 y 1 y no depende de la escala de Y.

Entonces, se utiliza la siguiente fórmula matemática

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS} \qquad (9)$$

Donde $TSS = \sum(y_i - \bar{y})^2$ es la suma de los cuadrados, y RSS como la suma total.

1. El TSS mide la varianza total de la respuesta Y y se puede considerarse como la cantidad de variabilidad a la respuesta antes de realizar la regresión.
2. El RSS mide la cantidad de variabilidad que queda sin explicar tras realizar la regresión.
3. TSS-RSS mide la cantidad de variabilidad de la respuesta que se explica al realizar la regresión.
4. R^2 mide la proporción de la variabilidad de Y que puede explicarse utilizando X.

Un valor cercano a 1 quiere decir que la regresión explica buena parte de la variabilidad de la respuesta; uno cercano a 0, que explica poco. Esto último pasa cuando el modelo lineal no es el adecuado, cuando la varianza del error es alta, o por las dos cosas a la vez.

El estadístico R^2 ofrece la ventaja interpretativa sobre RSE, ya que, a diferencia del RSE, siempre se encuentra entre 0 y 1. Además, R^2 es una medida de la relación lineal entre X y Y.

La correlación está definida como

$$Cor(X,Y) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}, \qquad (10)$$

también es una medida de la relación lineal entre X e Y. Esto sugiere que podríamos usar $r = Cor(X, Y)$ en lugar de R^2 para evaluar el ajuste del modelo lineal. (James et al., 2023)

2.2.6. Regresión lineal múltiple

La regresión lineal simple sirve, pero en la práctica casi siempre hay más de un predictor. En vez de ajustar un modelo simple por separado para cada uno, conviene extender el modelo para que incluya varios predictores a la vez. Para eso se le asigna a cada predictor su propio coeficiente de pendiente dentro de un único modelo.

En general, supongamos que tenemos predictores distintos. Entonces, el modelo de regresión lineal múltiple adopta la forma

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \qquad (11)$$

Donde,

X_j: representa el j-ésimo predictor.

β_j cuantifica la asociación entre esa variable y la respuesta.

Interpretamos β_j como el efecto promedio sobre Y de un aumento de una unidad en X_j, manteniendo fijos todos los demás predictores.

2.2.6.1. Estimación de los coeficientes de regresión

Como en el caso de la regresión lineal simple, los coeficientes de regresión $\beta_0, \ldots, \beta_p$ son desconocidos y deben estimarse. Con las estimaciones $\hat{\beta}_0, \ldots, \hat{\beta}_p$ podemos hacer predicciones usando la formula

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p. \quad (12)$$

Donde,

$\hat{y}$ es el valor predicho.

p es el número de características.

x_i es el valor de la i-ésima característica.

$\hat{\beta}_j$es el j-ésimo parámetro del modelo (incluyendo el término de sesgo $\hat{\beta}_0$ y los pesos de las características $\hat{\beta}_0, \dots, \hat{\beta}_p$.

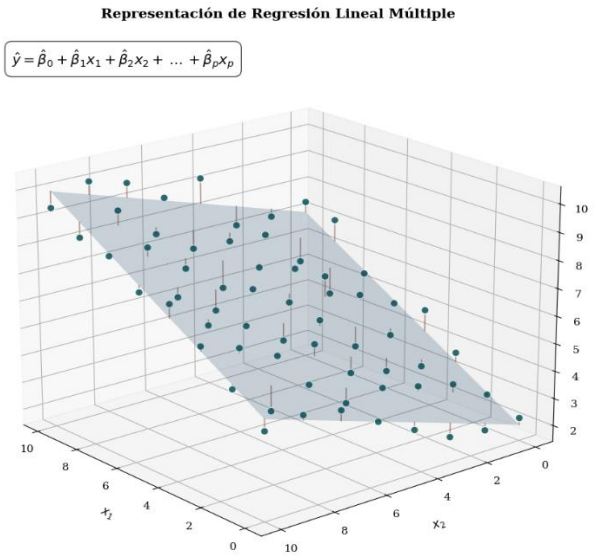
Los parámetros se estiman utilizando el mismo método de mínimos cuadrado en el contexto de regresión lineal simple. Elegimos $\beta_0, \dots, \beta_p$ para minimizar la suma de los residuos al cuadrado

$$RSS = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$
$$= \sum_{i=1}^{n} \left(y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip}\right)^2. \quad (13)$$

Los valores $\hat{\beta}_0, \dots, \hat{\beta}_p$ que minimizan la ecuación anterior son las estimaciones de los coeficientes por mínimos cuadrados múltiples. A diferencia del caso simple, aquí las fórmulas son algo más enredadas y se expresan mejor con álgebra matricial.

Figura 2

Regresión Lineal Múltiple



Nota. Se muestra el ajuste de un hiperplano de regresión $\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$ sobre un conjunto de datos en un espacio tridimensional. Las esferas representan las observaciones reales y_i, mientras que las líneas verticales denotan los residuos, definidos como la diferencia entre el valor observado y el valor predicho.

En regresión lineal, para determinar si existe una relación entre la variable de respuesta y la variable predictora, basta con comprobar si $\beta_1 = 0$. En regresión múltiple con p variables predictoras, debemos preguntarnos si todos los coeficientes de regresión son cero, es decir

$$\beta_1 = \cdots = \beta_p = 0 \qquad (14)$$

Utilizamos una prueba de hipótesis para comprobar la relación entre la variable de respuesta y las variables predictoras. Probamos la hipótesis nula,

$$H_0: \beta_1 = \cdots = \beta_p = 0 \quad (15)$$

Versus la alternativa

$$H_\alpha: al\ menos\ una\ \beta_j\ no\ es\ cero. \quad (16)$$

Esta prueba de hipótesis se realiza calculando el estadístico F, donde, como en la regresión lineal simple,

$$F = \frac{\frac{TSS - RSS}{p}}{\frac{RSS}{(n-p-1)}}, \quad (17)$$

Donde, $TSS = \sum(y_i - \bar{y})^2$ y $RSS = \sum(y_i - \hat{y}_i)^2$. Si los supuestos del modelo son correctos, se puede demostrar que

$$E\left\{\frac{RSS}{n-p-1}\right\} = \sigma^2 \quad (18)$$

Y que, siempre que H_0 sea verdadero,

$$E\left\{\frac{TSS - RSS}{p}\right\} = \sigma^2 \quad (19)$$

Por lo tanto, cuando no hay relación entre la respuesta y los predictores, se esperaría que el estadístico F tomara un valor cercano a 1. Por otro lado, si H_α es verdadera, entonces $E\left\{\frac{TSS-RSS}{p}\right\} > \sigma^2$, por lo que se espera que F sea mayor que 1.

Problemas potenciales

Cuando ajustamos un modelo de regresión lineal a un conjunto de datos particular, pueden surgir muchos problemas. Los más comunes son los siguientes:

1. No linealidad de las relaciones entre la respuesta y el predictor.
2. Correlación de los términos de error.
3. Varianza no constante de los términos de error.
4. Valores atípicos.
5. Puntos de alto apalancamiento.
6. Colinealidad.

Para poder operar de manera simultánea todo el conjunto de parámetros y características $\hat{y}$ se le puede escribir de forma vectorizada de la siguiente manera

$$\hat{y} = h_{\beta}(x) = \beta \cdot x \qquad (20)$$

Donde,

β es el vector de parámetros del modelo, contiene el sesgo y los pesos de las características.

x es el vector de características de la instancia.

$\beta \cdot x$ es el producto escalar de los vectores, resulta la suma $\hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$.

Entrenar el modelo consiste en ajustar los parámetros para que pegue lo mejor posible con el conjunto de entrenamiento. Para eso hace falta una métrica que diga qué tan bien encaja. Una de las más usadas es la Raíz del Error Cuadrático Medio (RMSE), así que se busca el valor de β que minimice el RMSE.

En la práctica es más cómodo minimizar el Error Cuadrático Medio (MSE), que da el mismo resultado: el valor que minimiza una función también minimiza su raíz cuadrada.

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2.} \qquad (21)$$

2.2.6.2. Ecuación Normal

Al entrenar un modelo se ajusta los parámetros β para minimizar una función de costo. Para obtener el valor que minimiza esta función, se utiliza la ecuación normal:

$$\hat{\beta} = (X^T X)^{-1} X^T y \qquad (22)$$

Donde,

$\hat{\beta}$ valor de los parámetros que minimiza la función de costo.

y es el vector de valores objetivos.

2.2.6.3. Gradiente Descendiente.

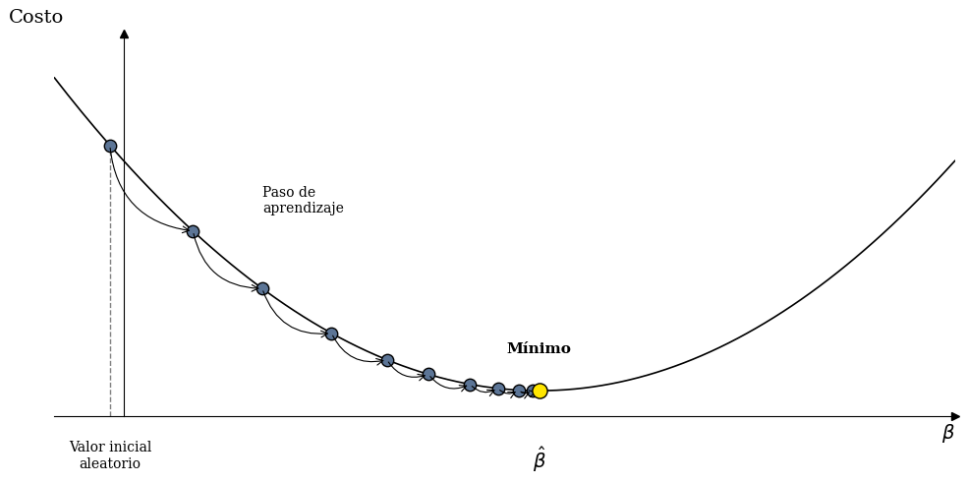
Es un algoritmo de optimización que permite encontrar soluciones óptimas. La idea fundamental es ajustar los parámetros de manera iterativa para minimizar la función de costo.

En la práctica el algoritmo empieza asignando valores aleatorios a $\hat{\beta}$. Luego, mejora los valores gradualmente, dando un pequeño paso a la vez, Cada paso intenta reducir la función de costo hasta que el algoritmo converge hacia un mínimo.

La ecuación normal da la respuesta directa, pero el Descenso de Gradiente gana cuando hay muchas características o muchos datos, porque resulta más eficiente en cómputo.

Figura 3

Descenso del Gradiente



Nota. Los parámetros parten de valores aleatorios y se ajustan para minimizar la función de costo; como el tamaño del paso es proporcional a la pendiente, los pasos se van haciendo más cortos a medida que se acerca al mínimo.

El tamaño del paso lo fija el hiperparámetro de la tasa de aprendizaje. Si es muy pequeño, el algoritmo necesita demasiadas iteraciones para converger y tarda mucho. Y si es muy alta, el algoritmo puede divergir y no llegar a una buena solución.

Como la función de costo MSE de una regresión lineal es convexa, no tiene mínimos locales, solo uno global. Por eso el descenso del gradiente se acerca tanto como se quiera a ese mínimo global.

Nota: Al usar Descenso de Gradiente, conviene que todas las características estén en una escala parecida; si no, la convergencia tarda mucho más.

2.2.7. Árboles de decisión

Vamos a analizar el proceso de construcción de un árbol de regresión.

Dividimos el espacio predictivo, es decir, el conjunto de valores posibles para $X_1, X_2, \dots, X_p$ en J regiones distintas y no superpuestas, $R_1, R_2, \dots, R_J$

Para cada observación que cae dentro de la región R_j, hacemos la misma predicción, que es simplemente la media de los valores de respuesta para las observaciones de entrenamiento en R_j.

Por ejemplo, supongamos que en el paso 1 obtenemos dos regiones, $R1$ y $R2$, y la media de respuesta en la primera región es 10 y en la segunda es 20. Entonces, para una observación dada $X = x, si\ x \in R_1$ predeciremos un valor de 10, y si $x \in R_2$ predeciremos un valor de 20.

En teoría las regiones podrían tener cualquier forma, pero se eligen rectángulos de alta dimensión, o cajas, porque simplifican el modelo y lo hacen más fácil de interpretar. La meta es hallar las cajas $R_1, \dots, R_J$ que minimicen la suma de cuadrados de los residuos (RSS), dada por

$$\sum_{j=1}^{J} \sum_{i \in R_j} \left(y_i - \hat{y}_{R_j}\right)^2. \qquad (23)$$

Dado que al evaluar todas las particiones posibles resulta computacionalmente no viable, se emplea un enfoque top-down y greedy:

- Top down: El algoritmo comienza con todas las observaciones en una única región y procede a realizar particiones sucesivas, creando ramas en cada nodo del árbol.

- Greedy: En cada nodo, el algoritmo selecciona el predictor X_j y el punto de corte s que optimizan la reducción inmediata del Error Cuadrático Residual RSS. El objetivo es encontrar el par (j, s) que minimice la ecuación

$$RSS = \sum_{i:x_i \in R_1(j,s)} (y_i - \hat{y}_{R_1})^2 + \sum_{i:x_i \in R_2(j,s)} (y_i - \hat{y}_{R_2})^2 \quad (24)$$

Donde

$\hat{y}_{R_1}$representa la media de las respuestas de las observaciones de entrenamiento contenidas en la región $R_1(j, s)$.

$\hat{y}_{R_2}$representa la media de las respuestas de las observaciones de entrenamiento contenidas en la región $R_2(j, s)$.

Con esto se elige la mejor partición en cada nodo, probando todas las combinaciones de predictores y puntos de corte. Luego el proceso se repite sobre las nuevas subregiones hasta llegar a un criterio de parada.

2.2.8. Random Forest

Los bosques aleatorios mejoran a los árboles de decisión con un truco que descorrelaciona los árboles. Se arman varios árboles a partir de muestras de entrenamiento con remuestreo, pero con un detalle: cada vez que toca dividir un nodo, se sortea un subconjunto aleatorio de m predictores como candidatos a la división del conjunto completo de p predictores. Se toma una nueva muestra de m predictores en cada división, y normalmente se elige $m \approx \sqrt{p}$, es decir, el

número de predictores considerados en cada división es aproximadamente igual a la raíz cuadrada del número total de predictores.

Dicho de otro modo, en cada división el algoritmo ni siquiera ve la mayoría de los predictores. Imaginemos que hay un predictor muy fuerte y otros moderados: si se usaran todos, casi todos los árboles pondrían ese predictor fuerte en la división de arriba. Así, todos los árboles del bagging (Bootstrap Aggregating) terminarían pareciéndose mucho entre sí.

Y árboles parecidos dan predicciones muy correlacionadas. El problema es que promediar cantidades muy correlacionadas no baja la varianza tanto como promediar cantidades poco correlacionadas, así que el remuestreo por sí solo no reduce gran cosa la varianza de un árbol en este caso.

Los bosques aleatorios superan este problema al obligar a cada división a considerar solo un subconjunto de predictores. Por lo tanto, en promedio $(p - m)p$ de las divisiones ni miran el predictor más fuerte, lo que da chance a los demás de pesar más. Esto descorrelaciona los árboles, reduce la varianza del promedio y hace el resultado más confiable.

La principal diferencia entre el bagging y los bosques aleatorios radica en la elección del tamaño del subconjunto de predictores m. Por ejemplo, si se construye un bosque aleatorio con $m = p$, esto equivale simplemente a bagging.

2.2.9. Método de ensamble de Tipo Boosting

La idea central del Boosting es entrenar predictores uno tras otro, donde cada uno corrige al anterior. Enseguida se ven AdaBoost y Gradient Boosting.

2.2.9.1. Algoritmo Adaptive Boosting (AdaBoost)

AdaBoost entrena un clasificador (por lo general un árbol de decisión) sobre todo el conjunto de datos, dando al inicio el mismo peso a cada observación. Tras medir cómo le fue al modelo base, marca los casos mal clasificados o con mayor desajuste y les sube el peso de forma exponencial.

El siguiente predictor se entrena con esos pesos ya actualizados, lo que lo obliga a concentrarse en los errores que dejó la fase anterior.

Esto se repite un número fijo de veces. La predicción final del conjunto es una combinación lineal ponderada de todos los predictores base, y el peso de cada voto depende de qué tan preciso fue en el entrenamiento.

AdaBoost se parece al descenso del gradiente, salvo que en lugar de ajustar los parámetros de un solo predictor para minimizar la función de costo, va sumando predictores al conjunto y lo mejora poco a poco.

Una vez entrenados todos los predictores, el conjunto predice igual que en Bagging, con la diferencia de que cada predictor pesa distinto según su precisión sobre el conjunto de entrenamiento ponderado.

El peso de cada instancia $w^{(i)}$ se establece inicialmente en $1/m$. Se entrena un primer predictor y se calcula su tasa de error ponderada r_1 sobre el conjunto de entrenamiento. A continuación, se establece la tasa de error ponderada del j-ésimo predictor

$$r_j = \left(\sum_{\substack{i=1 \\ \hat{y}_j^{(i)} \neq y^{(i)}}}^{m} w^{(i)} \right) / \left(\sum_{i=1}^{m} w^{(i)} \right) \quad (25)$$

Donde,

r_j: tasa de error ponderada correspondiente al j-ésimo estimador base.

m: Número de instancias en el conjunto de entrenamiento.

$w^{(i)}$: Peso asignado a la j-ésima instancia en la iteración actual.

$\hat{y}_j^{(i)}$: Vector de predicciones vectoriales generado por el j-ésimo predictor para la i-ésima instancia.

$y^{(i)}$: Valor objetivo real correspondiente a la i-ésima instancia.

$\hat{y}_j^{(i)} \neq y^{(i)}$: Condición lógica que limita la suma del numerador solo a las observaciones donde el modelo se equivocó al clasificar o estimar

El peso α_j del predictor se calcula mediante la siguiente ecuación

$$\alpha_j = \eta \log \frac{1 - r_j}{r_j} \quad (26)$$

Donde,

η: Hiperparámetro de la tasa de aprendizaje, con valor por defecto 1. Mientras más preciso el predictor, mayor su peso. Si acierta al azar, el peso queda cerca de cero, y si falla más de lo esperado, se vuelve negativo.

Con la siguiente ecuación, AdaBoost sube el peso de las instancias mal clasificadas: las que quedaron bien mantienen su ponderación y las que tienen error de estimación crecen de forma exponencial. De ese modo, el estimador queda forzado a priorizar la reducción del error.

$$w^{(i)} \leftarrow \begin{cases} w^{(i)} & si\ \hat{y}_j^{(i)} = y^{(i)} \\ w^{(i)}\exp(\alpha_j) & si\ \hat{y}_j^{(i)} \neq y^{(i)} \end{cases} \quad (27)$$

Por último, se entrena un nuevo predictor con los pesos actualizados y se repite todo. El algoritmo para cuando se llega al número de predictores buscado o aparece un predictor perfecto.

Por lo tanto, AdaBoost calcula las predicciones de todos los predictores y las pondera utilizando los pesos de los predictores α_j.

Las predicciones de AdaBoost se definen de la siguiente manera

$$\hat{y}(x) = \underset{k}{\operatorname{argmax}} \sum_{\substack{j=1 \\ \hat{y}_j(x)=k}}^{N} \alpha_j \quad (28)$$

Donde Nes el número de predictoras.

Nota: Si AdaBoost sobreajusta el conjunto de entrenamiento, conviene bajar el número de estimadores o regularizar más el estimador base.

2.2.9.2. Aumento del Gradiente (Gradient Boosting)

Este algoritmo también va sumando predictores al conjunto, cada uno corrigiendo al anterior. La diferencia con AdaBoost es que no reajusta los pesos de las instancias en cada vuelta, sino que ajusta el nuevo predictor a los errores residuales que dejó el predictor previo.

2.2.10. Métricas de evaluación de modelos

Para validar la capacidad de predicción de los modelos implementados se realiza el análisis de sus residuos. El medir el error un proceso clave para verificar la fiabilidad de las predicciones y poder optimizar el algoritmo (Chai & Draxler, 2014). Por lo tanto, buscando un balance que permita examinar tanto la magnitud absoluta de las desviaciones como la sensibilidad a valores atípicos, se seleccionan tres indicadores esenciales: MAE, RMSE y MAPE.

La formulación matemática y el sentido analítico de cada métrica se estructuran de la siguiente manera:

MAE (Error Absoluto Medio)

Para analizar la precisión sin penalizaciones exponenciales, se recurre al Error Absoluto Medio, el cual determina la magnitud de los residuos de forma independiente a su signo. Willmott y Matsuura (2005) lo describen esencialmente como la distancia geométrica promedio entre las observaciones reales y los pronósticos. Debido a que evalúa las diferencias en su escala original y lineal. Esta métrica proporciona una lectura estable y menos sensible a anomalías extremas en la muestra.

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}| \quad (29)$$

Donde,

n: Representa el número total de observaciones o tamaño de la muestra evaluada.

y_i: es el valor real u observado de la variable dependiente para la observación i.

$\hat{y}$: es el valor predicho o estimado por el modelo para la observación i.

RMSE (Raíz del Error Cuadrático Medio)

La Raiz del Error Cuadrático Medio evalúa la desviación promedio de las predicciones respecto a los valores reales. Esta métrica eleva los errores al cuadrado antes de promediarlos y, finalmente extrae la raíz cuadrada del resultado (Chai & Draxler, 2014). Esto implica que penalice de forma mas severa los errores de gran magnitud. Por tanto, es una métrica ideal cuando las desviaciones grandes son particularmente costosas en el contexto de la investigación.

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y})^2} \quad (30)$$

Donde los componentes n, y_i y $\hat{y}$ mantienen el concepto definido previamente.

MAPE (Error Porcentual Absoluto Medio)

El error porcentual Absoluto Medio evalúa la precisión que tiene el modelo expresando el error como un porcentaje relativo a los valores reales observados (Hyndman & Koehler, 2006). Esto permite comprender el tamaño del error en términos proporcionales, facilitando la comparación directa del rendimiento entre modelos aplicados a variables con diferentes escalas.

$$MAPE = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}}{y_i}\right| \quad (31)$$

La combinación de estas tres métricas nos asegura robustez en la evaluación desempeño de los modelos predictivos.

3. CAPÍTULO 3: DESARROLLO

3.1.Desarrollo del trabajo

3.1.1. Enfoque general del desarrollo

Para predecir la eficiencia y la productividad de la planta se trabajó con un enfoque aplicado y experimental, *"montado sobre un pipeline (flujo de procesamiento) que convierte los registros sueltos de producción en una base estructurada, consistente y lista para entrenar modelos de machine learning" (Sculley et al., 2015).* La idea es que el proceso sea reproducible: que las transformaciones queden documentadas y que se prevengan los sesgos que de otro modo arrastrarían los modelos y ensuciarían los resultados.

La materia prima son datos históricos reales de una empresa ecuatoriana de inyección de plásticos, que cubren de 2020 a abril de 2026. Llegaron desglosados en diversos archivos de Excel, y además presentaban diferencias entre años: distintos nombres de columna, distinta cantidad de campos, e incluso dos tipos de planilla incompatibles. Por ello, antes pensar en modelar, hubo que estandarizar, limpiar y reorganizar todo para dejarlo en una estructura fija que se mantenga en el tiempo. Desde el punto de vista metodológico el problema es de aprendizaje supervisado: las variables objetivo son la eficiencia del turno y la productividad en piezas por hora, ambas continuas, calculadas a partir de las horas, las cantidades producidas y las piezas buenas y defectuosas de cada turno. El desarrollo avanza por etapas encadenadas, en las que cada una alimenta a la siguiente: primero la transformación del formato crudo, luego la construcción del conjunto limpio, y finalmente el entrenamiento y la evaluación de los modelos, cuyos resultados se reportan en el Capítulo 4.

3.1.2. Arquitectura de datos por capas

Estructura general. El flujo de datos se organizó con una *arquitectura por capas tipo Medallion* (Databricks 2021), que separa la información en niveles de refinamiento crecientes. Cada capa se materializa en archivos planos y la lógica del flujo está respaldada en diagramas de arquitectura.

Función de cada capa. "En la capa Bronze entran los registros crudos por turno, tal como salen de los archivos de la empresa, sin transformar. En la capa Silver esos mismos registros pasan por limpieza, normalización y unificación de esquema" (Databricks, 2021), y quedan listos para el análisis.

Ventaja operativa. La separación conlleva una ventaja práctica, y es que, si aparece un error en la limpieza, la limpieza se corregiría en Silver sin tocar el dato original de Bronze, de esta manera se garantiza la trazabilidad del extremo al extremo.

La Figura 4 figura la organización por capas que va del dato crudo al conjunto a modelar.

Figura 4

Arquitectura de datos por capas (Medallion)

Transformación y estandarización inicial de los datos

La primera fase fue la que llevaba a cabo el formato original para pasar a ser un formato que pudiera ser analizado con el software de análisis de datos. Los datos estaban en hojas de cálculo con los encabezados de las distintas columnas en distintas posiciones; existían nombres de columnas que en cada uno de los años podían ser diferentes, además de dos esquemas diferentes según la época: uno antiguo, el que se había usado en los registros más antiguos, era el que tenía los campos: fecha, registro, código de operador, máquina, código de producto, cantidad producida, cantidad planificada, turno y horas; el segundo, más reciente, contaba con mes, día, máquina, número de turno, número de lote, código SAP, descripción de producto, operadores, supervisor, horas planificadas, horas de parada, motivo de la parada y datos de kilogramos de material y ciclos que en el formato anterior no se recogían.

A fin de salvar dicha diferencia de formato se aplicó un procedimiento consistente en determinar la fila de encabezado correcta para cada archivo y tolerar las diferencias entre planillas, de forma que un archivo fuese procesado aunque su estructura no coincidiese con el archivo correspondiente a otro año. Una vez contenida la estructura válida se normalizaron los nombres de columna, dado que una misma variable aparecía escrita de distintas formas, determinándose un conjunto de claves posibles y un nombre estándar asociado a cada uno de los campos y, finalmente, los nombres originales fueron automáticamente mapeados a ese estándar, de forma que máquina, turno, lote, código de producto, operador y supervisor pasaron a tener un único nombre, el nombre en snake_case (es decir, el nombre en minúsculas, separado por guiones bajos) a lo largo de todo el conjunto de datos sin importar de qué año vinieran.

En esta fase también se eliminaron columnas que únicamente suponían ruido para el modelado, tales como las que representaban los nombres de los operadores y de los supervisores, manteniendo sus IDS. A continuación, se normalizó el campo de mes, que contenía cadenas de texto en mayúsculas con errores ortográficos (por ejemplo, se corrigió un mes erróneo antes de hacer el caso que lo mapeaba a número).

Se construyó una fecha con el día y el año para ordenar los registros con el tiempo. Sobre los campos numéricos se aplicaron rutinas de limpieza para manejar valores faltantes, símbolos no numéricos y registros incompletos, y se verificó la coherencia entre los totales declarados y los calculados a partir de las piezas buenas y defectuosas, descartando los registros que no cuadraban.

Como resultado de esta etapa se obtuvo la capa *Silver*: una base unificada de 27.253 registros a nivel de turno, con esquema único y trazabilidad de cada fila a su archivo y formato de origen. Esta base es el punto de partida de todo lo que viene después, porque garantiza que los datos de entrada cumplan los requisitos de calidad, consistencia y estructura que un entrenamiento confiable necesita.

3.1.3. Los dos formatos de origen y su unificación

Diferencias entre formatos.

Características del formato antiguo. Los años más antiguos guardan la cantidad producida en una sola columna, sin separar piezas buenas de defectuosas, traen la fecha en un campo propio y anotan el turno como mañana o tarde.

Características del formato reciente. Los años recientes, en cambio, separan piezas buenas y malas, agregan purga y pesos en kilogramos, registran ciclo planificado y ciclo real, parten la fecha en mes y día, y nombran el turno como primero, segundo o tercero.

Procedimiento de unificación.

Criterio de integración. Esa diferencia no es un detalle cosmético: cambia qué variables existen en cada tramo de la historia. Por eso *"la unificación tuvo que mapear ambos esquemas hacia uno solo" (Rahm & Do, 2000)*, decidiendo para cada campo de dónde sale en cada formato y qué hacer cuando un formato simplemente no lo tiene.

Tratamiento de valores ausentes. Donde el formato antiguo no distingue piezas buenas de defectuosas, esa información queda como faltante y se trata como tal; no se inventa un valor. El resultado es un esquema común que respeta lo que cada época sí registró, sin forzar una equivalencia que no existe.

Correcciones manuales aplicadas.

Errores estructurales detectados. Hubo detalles de captura que debieron sanearse manualmente. Un año traía la cabecera corrida una fila, con columnas sin nombre, y hubo que indicarle al lector dónde empezaba la tabla. Otro archivo arrastraba una segunda hoja con fechas sueltas que no correspondían a datos de producción y se descartó.

Errores de formato en el campo de fecha. En los años recientes el mes venía escrito junto al año en un mismo campo, con errores como un año imposible o espacios de más, que debieron corregirse antes de poder ordenar los registros cronológicamente.

La Tabla 2 resume las diferencias entre los dos formatos de origen.

Tabla 2

Comparación de los dos formatos de origen

Aspecto	Formato antiguo (2020-2023)	Formato reciente (2024-2026)

Cantidades	Solo cantidad producida	Piezas buenas y defectuosas separadas
Materiales	No registra kg	Kg de material, piezas y purga
Fecha	Campo de fecha único	Mes y día por separado
Turno	Mañana / tarde	Primero / segundo / tercero
Ciclos	No registra	Ciclo planificado y real

3.1.4. La definición de eficiencia: una decisión, no un dato heredado

El punto más delicado de toda la preparación fue la variable objetivo. Los archivos ya traían una columna de porcentaje, pero al revisarla se vio que no significaba lo mismo en todos los años. En los registros antiguos ese porcentaje se acerca a la cantidad producida sobre la planificada, es decir, una eficiencia de cantidad. En los recientes corresponde al ciclo planificado sobre el ciclo real, o sea, una eficiencia de velocidad. Son dos cosas distintas: una mide cuánto se produjo frente a lo previsto, la otra qué tan rápido giró la máquina frente a su cadencia teórica.

Si se hiciera una mezcla entre ambas sin más, el modelo estaría aprendiendo a partir de una variable cuyo significado cambia a mitad de historia, lo cual contamina cualquier conclusión. Encima de todo, la fiabilidad del % de origen fue bastante mala incluso dentro de cada época (el % de origen fue, en varios años, de poca utilidad, ya que su correlación con la fórmula esperada era baja, lo que indica que o bien se había calculado con criterios inconsistentes, o bien se había subido manualmente, también con errores). La solución, entonces, fue la de no confiar en esa columna y volver a construir la eficiencia de base, a partir de una única definición, que sí fuera computable en los seis años: el de las piezas buenas sobre la cantidad planificada. Esta definición es clara, se puede computar en todo el periodo y, además, deja explícito el criterio del negocio.

Conviene ser transparente con esto: la elección de eficiencia es una decisión metodológica, no un dato que venga dado. *“Se la dejó parametrizable, de modo que si el equipo de la planta*

prefiere la definición basada en ciclo, o una que combine cantidad y calidad, el pipeline pueda recalcularla sin rehacer todo" (Sculley et al., 2015). Esta es la clase de supuesto que conviene validar con quienes operan la planta, porque de él depende qué está prediciendo el modelo.

La Tabla 3 muestra cómo el porcentaje heredado cambiaba de significado según la época, lo que motivó reconstruir la eficiencia con una definición única.

Tabla 3

Definición del porcentaje de eficiencia según el formato

Período	Significado del % de origen	Tipo de eficiencia
2020-2023	Cantidad producida / planificada	De cantidad
2024-2026	Ciclo planificado / ciclo real	De velocidad
Adoptada	Piezas buenas / planificadas	Única y homogénea

3.1.5. Variables derivadas y sus fórmulas

"A partir de los campos crudos se construyeron las variables que el análisis y los modelos realmente usan" (Kuhn & Johnson, 2019). Cada una tiene una definición de negocio detrás, y vale la pena dejarla por escrito para que cualquiera pueda reproducir el cálculo.

Tabla 4

Variables derivadas: definición, fórmula e interpretación

Variable	Fórmula	Interpretación
Eficiencia	Piezas buenas / cantidad planificada	Qué parte de lo previsto se cumplió con producto aprovechable
Tasa de rechazo	Piezas defectuosas / total producido	Cuánto de lo producido no sirve
Cumplimiento del plan	Bandera: piezas buenas ≥ planificadas	Sí o no cumplió el turno su objetivo
Horas de producción	Horas planificadas − horas de parada	Tiempo efectivo de producción
Productividad	Piezas buenas / horas de producción	Piezas por hora efectiva
Ciclo real	(Horas producción × 3.600) / cantidad producida	Segundos por pieza

Nota. Las fórmulas se aplican de manera uniforme en todo el período 2020–2026. Cuando un campo necesario está ausente, la variable derivada queda como faltante en lugar de estimarse con datos que no corresponden. Elaboración propia.

Estas fórmulas se aplican de manera uniforme en todo el período, lo que evita el problema que tenían las columnas heredadas. Cuando un campo necesario falta (por ejemplo, las piezas buenas en el formato antiguo), la variable derivada queda como faltante en vez de calcularse con un dato que no corresponde, y esa decisión se registra para que el modelado la tenga en cuenta.

3.1.6. Depuración de errores de parsing

Durante la construcción del conjunto aparecieron errores que, de no haberse detectado, habrían arruinado los resultados sin dar la cara. De todos ellos, el más grave se presentó en el caso de emparejamiento de columnas: en el caso de buscar la cantidad planificada por su nombre el procedimiento tomaba la columna de horas planificadas por error, porque era la primera en la planilla. Eso hacía que la eficiencia arroje valores absurdos de miles por ciento, hasta que se ató la búsqueda con el nombre correcto. Un error análogo confundía el código de operador con el código producto, son errores silenciosos: programa no se queja, solo se limitaba a producir números equivocados, y solo se detectaban al revisarlos cuando hacían sentido.

También hubo que normalizar el turno que estaba escrito de formas diferentes según el formato y depurar la productividad que mostraba números negativos cuando las horas paradas eran mayores a las horas planificadas. Cada uno de estos arreglos se fue probando contra los datos reales, verificando por ejemplo que un turno conocido diera el tiempo de ciclo esperado, antes de

darlo por bueno. La lección de esta etapa es que la calidad del dato no se asume: se comprueba registro a registro, porque un solo emparejamiento mal hecho contamina todo el conjunto.

La Tabla 5 resume los errores detectados y su corrección.

Tabla 5

Errores de procesamiento detectados y corregidos

Error	Efecto	Corrección
Columna de horas confundida con cantidad planificada	Eficiencias de miles por ciento	Atar la búsqueda al nombre correcto
Código de operador cruzado con código de producto	Identificadores erróneos	Selección por nombre exacto
Turno escrito de formas distintas	Categorías inconsistentes	Normalización a un único formato
Horas de parada mayores a las planificadas	Productividad negativa	Cálculo solo con horas válidas

3.1.7. Consolidación del conjunto de entrenamiento (capa *Gold*)

Una vez corregidos los errores y unificadas las fórmulas, se consolidó un conjunto final orientado al entrenamiento, que se puede pensar como una capa *Gold* por encima de Silver: ya no es solo dato limpio, sino dato listo para alimentar a los modelos. Esta capa quedó con los seis años representados de forma equilibrada y con los catálogos saneados, del orden de varias decenas de máquinas, un par de cientos de productos y un centenar largo de operadores. Se filtraron además los registros con valores imposibles, como un turno con un código que no existe, que delataban errores de carga.

El conjunto se guardó en un formato columnar eficiente, pensado para que los *notebooks* (cuadernos de ejecución interactiva) lo lean rápido y siempre igual. A fin de que todo el proceso pudiera ser reproducible se fijó una semilla aleatoria común, para que al correr el pipeline, cualquiera obtenga los mismos cortes de datos, los mismos resultados. La reproducibilidad no es

un capricho: es lo que permite que el trabajo sea revisable, auditable, defendible, como realmente puede exigir una tesis.

3.1.8. Descripción de las variables finales del conjunto de datos

El conjunto final está organizado a nivel de turno: cada fila es un turno de producción con su fecha, su máquina, su producto, su operador y los indicadores de desempeño asociados. Las 51 columnas se agrupan en ocho bloques, descritos en la Tabla 6.

Tabla 6

Descripción de los grupos de variables del conjunto de datos final

Grupo	Variables que incluye	Observaciones
Trazabilidad	Identificador del registro, referencia al registro original en *Bronze*, formato y archivo de origen, marca de tiempo de carga, número de lote	Permiten auditar de dónde salió cada dato
Tiempo	Fecha del turno, año, mes, día, día de la semana, indicador de fin de semana, semana del año, trimestre	Capturan la estacionalidad y los patrones cíclicos de la planta
Catálogos	Turno, máquina, código y descripción del producto, identificadores de operador principal y secundario, indicador de segundo operador, supervisor	Variables categóricas que describen el contexto del turno
Tiempos y paradas	Horas planificadas, horas de parada, horas de producción, indicadores de parada y turno extendido, categoría del motivo de parada	—
Cantidades y ciclos	Cantidad planificada, piezas buenas, piezas defectuosas, total producido, ciclo planificado, ciclo real (segundos), desviación del ciclo real frente al planificado	Incluyen banderas de imputación cuando el ciclo fue estimado
Materiales (kg)	Kilogramos de piezas buenas, materia prima total y purga	Solo disponibles en el formato reciente; ausentes en el 61% de los registros históricos
Métricas derivadas	Eficiencia del turno, cumplimiento del plan, tasa de rechazo, productividad en piezas por hora	Son las variables objetivo que los modelos deben predecir
Calidad del dato	Puntaje de calidad por registro, bandera de dato completo	Sirven para filtrar qué registros entran al entrenamiento

Nota. Elaboración propia a partir de silver.production_clean. El grupo de materiales condiciona qué modelos pueden usar esas variables dado su nivel de completitud.

3.1.9. Construcción de la base de modelado y prevención de fuga de información

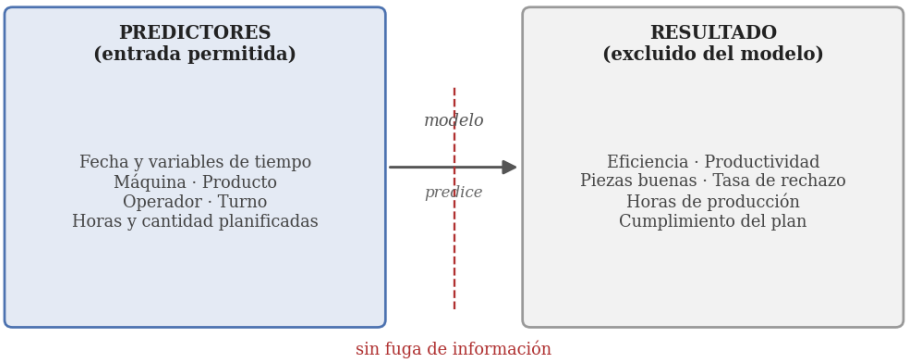
Separación entre contexto y resultado. Con la capa *Silver* lista, el siguiente paso es armar un conjunto específico para modelar. El objetivo es que los modelos se entrenen y evalúen bajo las mismas condiciones que tendrían en producción, sin acceder a información que en un escenario real todavía no existiría al momento de predecir. *"El riesgo a evitar se llama fuga de información (data leakage): ocurre cuando el modelo usa, directa o indirectamente, datos del resultado que se supone debe estimar" (Kaufman et al., 2012).* En este caso el riesgo es claro: la eficiencia se calcula con las piezas buenas, y la productividad con las horas de producción; si esas mismas columnas entraran como variables de entrada, el modelo estaría viendo la respuesta disfrazada y produciría métricas engañosamente buenas.

Selección controlada de variables de entrada. Por eso la construcción de la base de modelado se diseñó como una etapa aparte, que separa de forma explícita lo que es contexto del turno (máquina, producto, operador, turno, fecha, horas y cantidades planificadas) de lo que es resultado del turno (eficiencia, productividad, piezas buenas, rechazo). Solo lo primero puede usarse como entrada. El diccionario de la base marca de antemano qué columnas son resultados, justamente para no caer en esa trampa. Sobre esa marca se hace la selección controlada de variables, dejando fuera todo lo que dependa del desenlace del turno y conservando las variables temporales, de catálogo y de planificación que sí estarían disponibles antes de arrancar.

La Figura 5 ilustra la separación entre variables de entrada y variables de resultado que evita la fuga de información.

Figura 5

Separación de predictores y resultado para evitar la fuga de información



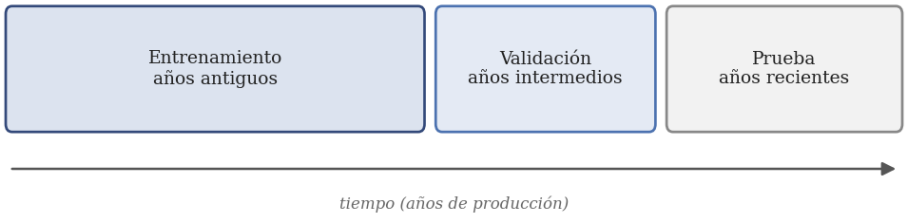
3.1.10. Partición de entrenamiento, validación y prueba

Para evaluar los modelos de forma honesta, los datos se dividen en tres bloques: entrenamiento, validación y prueba. Dado que el problema tiene una componente temporal fuerte (la planta cambia de un año a otro por desgaste, recambio de personal y mezcla de productos), *"la separación no se hace al azar sino por fecha: los turnos más antiguos van a entrenamiento, los intermedios a validación y los más recientes a prueba. Así se simula el escenario real, donde el modelo aprende del pasado y se mide contra turnos posteriores que no vio" (Hyndman & Athanasopoulos, 2018).* Como paso final se valida que cada variable tenga el tipo correcto (temporal, categórica o numérica), para evitar inconsistencias durante el entrenamiento.

La Figura 6 muestra la partición temporal del conjunto en entrenamiento, validación y prueba.

Figura 6

Partición temporal del conjunto de datos



3.1.11. Almacenamiento de la base

La base consolidada se guarda como conjunto estructurado y queda disponible para el equipo de trabajo, lo que permite que el entrenamiento se ejecute en distintos ambientes a partir de la misma fuente. La persistencia se apoya en Supabase, un servicio gestionado sobre PostgreSQL, que aloja las capas del flujo: una capa *Bronze* (production_raw) con los registros crudos tal como los ingresa el operador, y una capa *Silver* (production_clean) con los registros ya depurados y las variables derivadas. Apoyarse en una base relacional gestionada da portabilidad y permite que la captura, la limpieza y el consumo de los datos compartan una única fuente de verdad.

3.1.12. Implementación del sistema de captura y visualización

Sistema y control de acceso. Sobre esa base de datos se construyó una aplicación web con *Streamlit (Streamlit Inc., 2023),* desplegada en la nube y en operación continua. La aplicación ha implementado control de acceso basado en roles, con un perfil de operador y un perfil de gerencia, para que cada usuario realice únicamente las funciones que le corresponden. Esta separación entre captura y análisis proporciona al sistema un cierto orden, ya que el operador captura la información y la gerencia se ocupa de consultarla, sin que se puedan dar interferencias entre roles.

Perfil operador — captura de datos. El perfil del operador tiene un formulario para subir la producción de cada turno. Éste tiene validaciones de campos y un control que evita que esté duplicado el registro y también escribe directamente en la capa Bronze de la base de datos. Las variables que se capturan son exactamente las necesarias por parte del proyecto, que son: máquina, producto, operador, turno, horarios, cantidades planificadas y producidas, paradas y motivos. De este modo, el dato entra estructurado desde el origen, evitando así el trabajo de limpieza posterior.

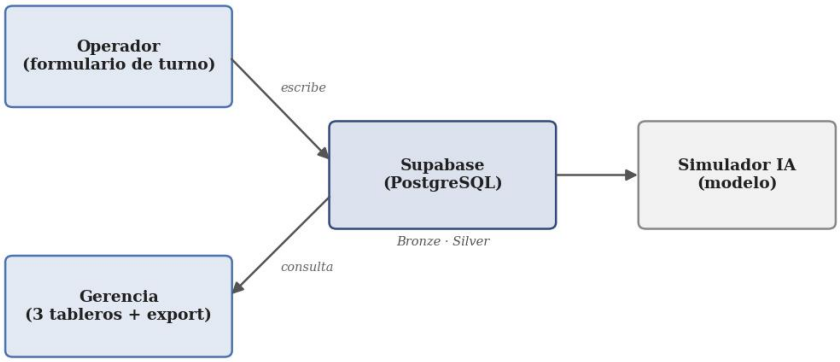
Perfil gerencia — visualización y análisis. En la interface se ofrece el perfil de gerencia que incluye la visualización de tres tableros a partir de la lectura de la Silver Layer, estos son el tablero de rendimiento, el de monitoreo y el histórico, todos ellos permiten la exportación a Excel para informes. Con esto quien planifica tiene una oportunidad de revisar el comportamiento de la planta según el máquina, el turno y el tiempo, sin la necesidad de acceder a la base de datos.

Simulador de predicción. La aplicación contempla además una pestaña de simulador, pensada para integrar el componente predictivo: una vez entrenado el modelo, el supervisor podrá seleccionar una combinación de máquina, producto, operador y turno, y obtendrá la eficiencia y la productividad esperadas. Esta pestaña cierra el recorrido del sistema, desde la captura del dato hasta el apoyo a la decisión, y constituye el punto de integración con el modelado descrito en el siguiente capítulo.

La Figura 3.4 resume el flujo del sistema desde la captura del operador hasta el simulador.

Figura 7

Arquitectura del sistema de captura y visualización

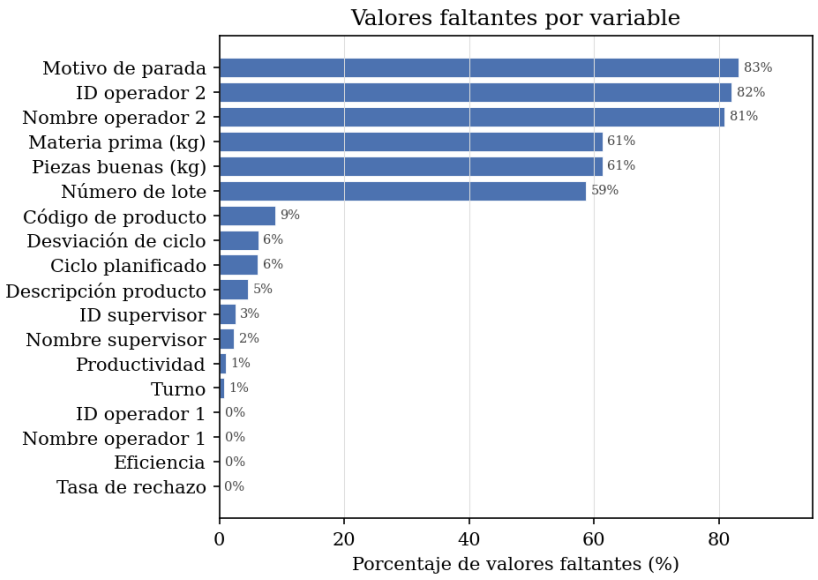


3.1.13. Análisis exploratorio de datos (EDA, por sus siglas en inglés) y control de calidad

Antes de modelar se hizo un análisis exploratorio para entender la base, detectar patrones y revisar la calidad. *"Se inspeccionaron las dimensiones del conjunto, los tipos por columna y las estadísticas descriptivas, y se construyó un perfil por variable con cantidad de nulos y de valores únicos" (Tukey, 1977)*. Eso permitió ubicar las variables categóricas de alta cardinalidad (sobre todo el motivo de parada, con miles de descripciones distintas) y medir qué tan completo está cada campo. También se confirmó la cobertura temporal del conjunto, de 2020 a 2026, repartida de forma despareja entre años porque cada período aportó un número distinto de turnos.

Figura 8

Porcentaje de valores faltantes por variable



La Figura 8 y la Tabla 7 muestran el resultado de ese conteo. Se ve que el grueso de los faltantes se concentra en variables que solo existen en el formato reciente o que dependen de condiciones puntuales del turno: el motivo de parada falta en la mayoría de los registros porque solo se anota cuando hubo parada; el segundo operador falta cuando el turno trabajó con una sola persona; y los kilogramos de material y el número de lote solo aparecen desde el formato nuevo. En cambio, las variables centrales para el modelado (eficiencia, turno, máquina, horas y cantidades) están prácticamente completas, lo que es una buena señal de calidad.

Tabla 7

Valores faltantes en las principales variables

Variable	Faltantes	Porcentaje
Motivo de parada	22.684	83,2 %
ID operador 2	22.367	82,1 %
Materia prima (kg)	16.720	61,4 %

Piezas buenas (kg)	16.719	61,3 %
Número de lote	16.017	58,8 %
Código de producto	2.437	8,9 %
Ciclo planificado	1.690	6,2 %
Supervisor	697	2,6 %
Productividad	272	1,0 %
Turno	205	0,8 %
Eficiencia	20	0,1 %

Seguidamente se comprobó la consistencia lógica de las variables numéricas observando que el total producido cuadrara con la suma de piezas buenas y defectuosas, y que la eficiencia cayera dentro de un rango razonable. Aquí se dio un hallazgo que se debe tener presente para el modelado, la eficiencia estaba muy concentrada en los extremos, habiendo muchos turnos en valores muy bajos y muchos otros con valores cercanos al cumplimiento o por encima de la totalidad del plan, e incluso algún valor negativo o superior a uno que delataba registros a mirar. Esta forma de la distribución debe tenerse presente para que los modelos no acaben prediciendo siempre lo mismo.

Finalmente se dejó marcado qué registros tienen dato completo (con kilogramos, ciclos y eficiencia a la vez), que son alrededor de diez mil registros de los más de veintisiete mil. Esta distinción importa porque las variables de material solamente existen en el formato reciente, de forma que cualquier modelo que las use deberá entrenarse sobre ese subconjunto, mientras que un modelo que use solamente contexto y tiempos se puede entrenar con toda la base.

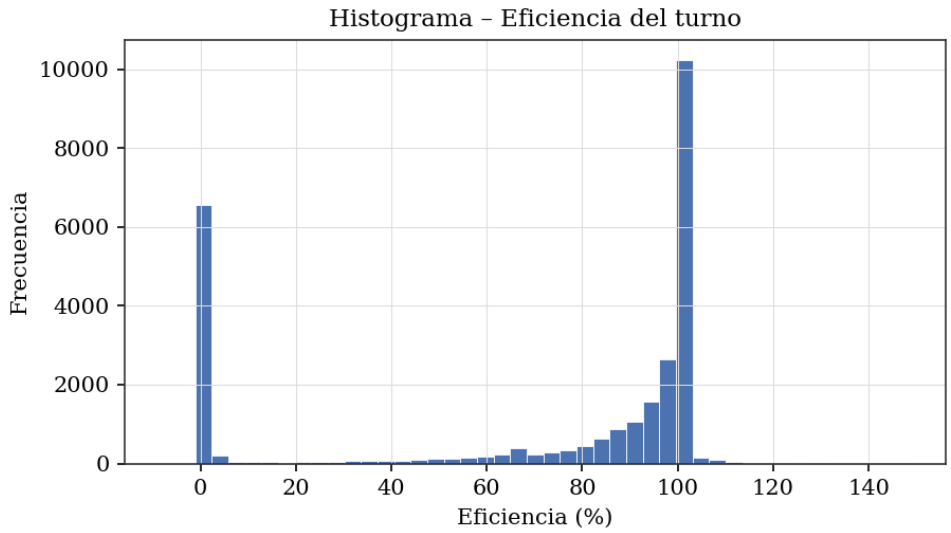
3.1.14. Distribución de la variable objetivo

Si se representa gráficamente la eficiencia se aprecian las irregulares distribuciones pues se presenta amontonada. La eficiencia se distribuye en un gran grupo de turnos con eficiencia muy baja, otro grupo de parecidas dimensiones pegado al cumplimiento completo del plan, y

relativamente pocos casos del centro de la distribución. Esta distribución bimodal dice algo del proceso real: el turno ha ido bien y se ha cumplido, o ha ido mal y ha rendido poco, ya sin demasiados grises intermedios. Para el modelado supone únicamente un aviso, dado que un modelo vago podría caer en predecir siempre una eficiencia central no muy apta para nadie. Aparecieron también algunos valores poco plausibles, en el sentido que estaban por debajo del límite de cero o por encima del cien por cien, valores de eficiencia imposibles que son claros indicios de registros a revisar antes de pasar a la fase de entrenar.

Figura 9

Histograma de la eficiencia del turno



3.1.15. Eficiencia por máquina, turno y operador

Se comparó el medio de la eficiencia en función de las grandes dimensiones del problema. Entre máquinas se observan diferencias claras, lo cual tiene sentido porque no todas las máquinas se encuentran en igualdad de condiciones ni se utilizan para ellos los mismos productos. También se aprecian diferencias entre los turnos, coherentes con que el turno determina el rendimiento a causa del cansancio, de la supervisión o de la cadencia de trabajo. Finalmente se identifica

dispersiones entre operadores, argumentando así que la experiencia de quien opera incide en el resultado. Estas comparativas no constituyen todavía un modelo, pero evidencian que las variables seleccionadas como predictoras (máquina, producto, operador, turno) sí tienen que ver con la eficiencia, suponiendo así el supuesto que fundamenta todo el proyecto.

Figura 10

Eficiencia promedio por año

Tabla 8

Resumen de registros y eficiencia por año

Año	Registros	Eficiencia media (%)	Eficiencia mediana (%)
2020	3.212	97,1	99,9
2021	6.205	61,7	89,0
2022	5.097	56,8	89,2
2023	3.984	77,5	96,1
2024	3.941	62,1	85,1
2025	3.512	72,4	98,5
2026	1.302	77,4	100,0
Total	27.253	69,4	95,4

Figura 11

Eficiencia promedio por máquina

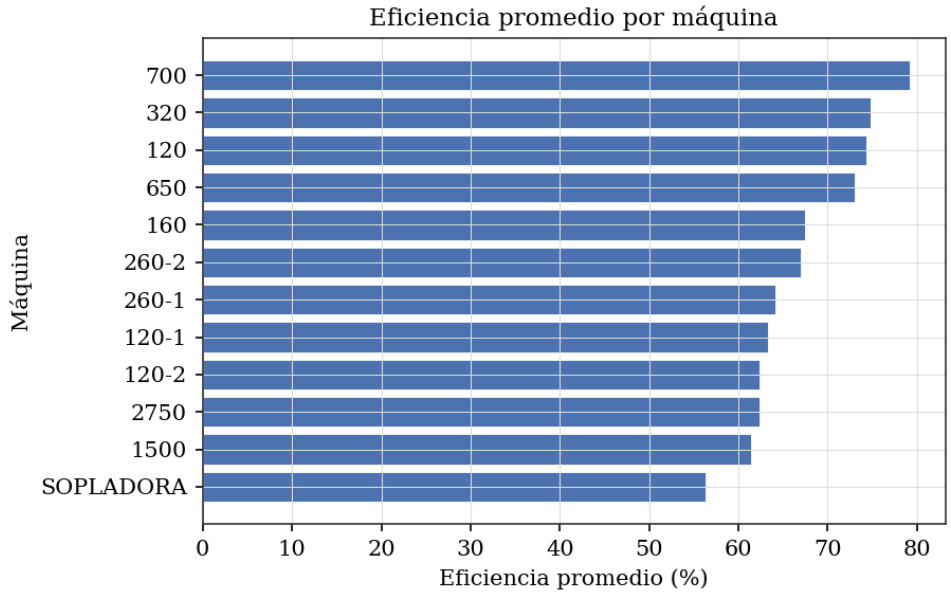
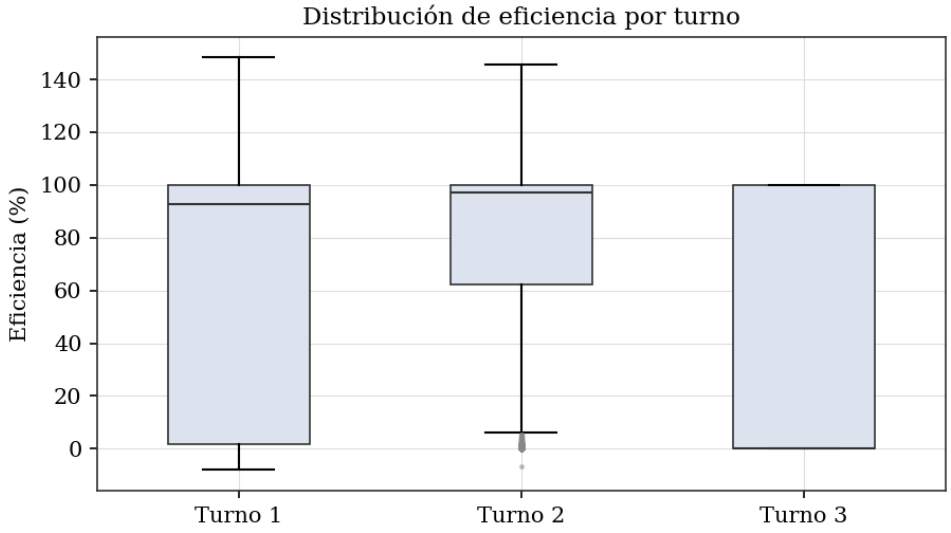


Figura 12

Distribución de la eficiencia por turno



3.1.16. Valores atípicos y registros dudosos

Detrás de los valores, imposibles de encontrar, hay asientos que son posibles, pero poco frecuentes, es decir, turnos que se caracterizan por una productividad muy alta o por tener un

número lo suficientemente alejado de lo habitual de la máquina en cuestión. Antes de tomar la decisión de borrarlos conviene reflexionar acerca de qué suponen, ya que hay algunos que son muestra de una situación real de la planta y otros son reflejo de que se ha cometido algún error de carga. *"El criterio que se adoptó fue conservador: marcar los casos extremos con una bandera en vez de eliminarlos de entrada" ((Aggarwal, 2017);* así, después se puede entrenar con los casos extremos y con aquellos que no lo son y ver cómo varían los resultados cuando se les incorpora. En el caso de los valores imposibles, directamente se eliminan. De este modo de proceder se mantiene a la base honesta y deja constancia de lo que se le ha cargado y por qué.

3.1.17. Síntesis de hallazgos para el modelado

Distribución de la variable objetivo. El análisis exploratorio evidenció que la eficiencia presenta una distribución concentrada en los extremos: se obtiene por un lado un grupo ampliamente numeroso de turnos que es de muy bajo rendimiento y, por el otro lado, otro grupo de turnos muy próximo al cumplimiento total del plan con escasísimos casos intermedios. Esta forma debe tenerse en consideración durante el modelado de modo que el modelo no converja hacia una solución intermedia que no represente ninguno de los dos grupos predominantes (Tukey, 1977).

Pertinencia de las variables de contexto. Las variables de contexto (máquina, producto, operador y turno) presentan relación estadística con el rendimiento observado, lo que da soporte al planteamiento de aprendizaje supervisado que hemos adoptado y también a la hora de justificar que debieran estar incluidas como variables de entrada dentro de los modelos.

Condicionantes del conjunto de datos. La base recoge dos formatos de origen de carácter diferente con campos de origen distintos a partir de los cuales se reconstruye una definición de

eficiencia desde cero; de este modo se hace necesario explicitar qué se produce y en qué condiciones son válidos los resultados, los cuales se retoman en el argumento correspondiente al Capítulo 4.

Tabla 9

Características generales del conjunto de datos final

Característica	Valor
Registros totales (turnos)	27.253
Período cubierto	2020, 2026
Máquinas distintas	16
Productos distintos	251
Operadores distintos	183
Variables (columnas)	51
Eficiencia media global (%)	69,4
Eficiencia mediana global (%)	95,4

Figura 13

Volumen de registros por año

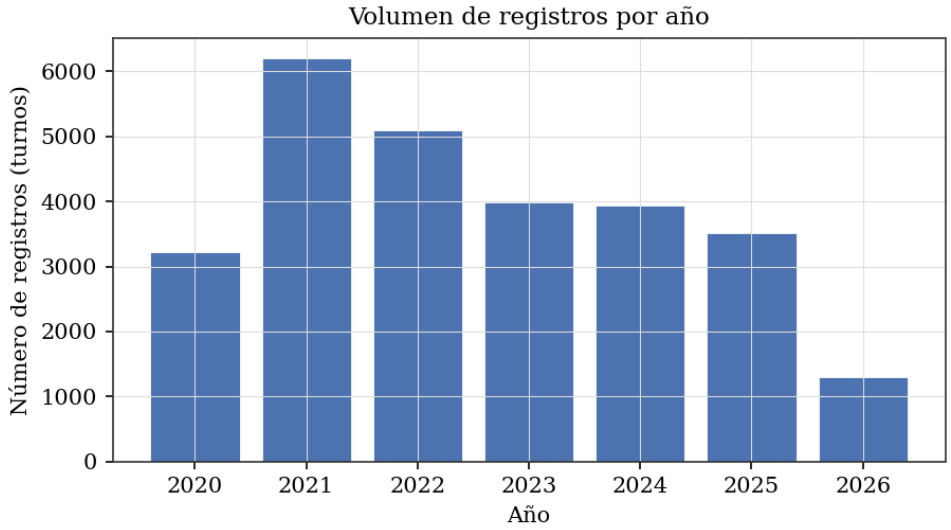
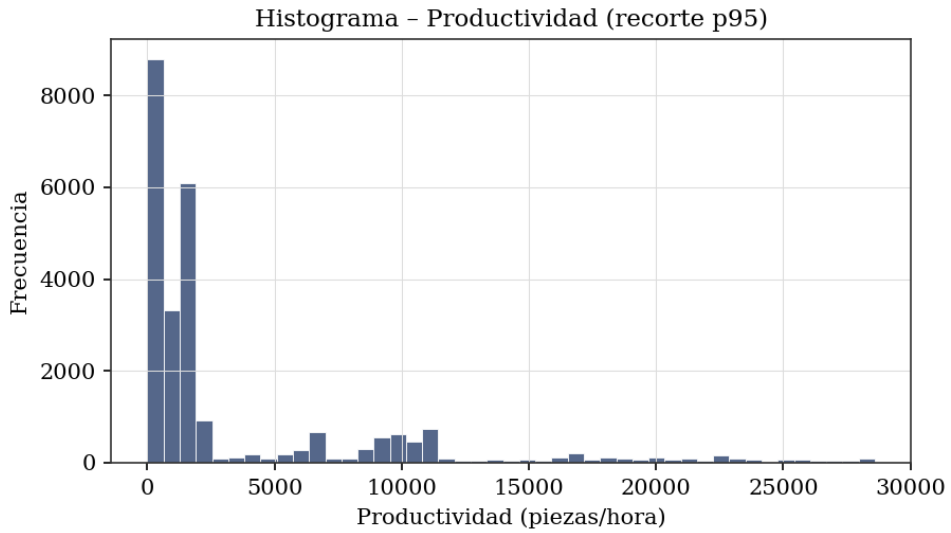


Figura 14

Histograma de la productividad (recorte al percentil 95)



3.1.18. Decisiones y supuestos por validar

Para cerrar este capítulo, es adecuado concentrar en un mismo lugar las decisiones que fueron tomadas de manera razonable pero que deberían ser confirmadas por el equipo de la planta (porque afectan a lo que el modelo estima). La primera es la definición de eficiencia (aquí se puso piezas buenas sobre planificadas, aunque hubo una alternativa que era sobre ciclo y en la que conviene quedar en cuál es la oficial). La segunda es el tratamiento del segundo operador, que en no pocas ocasiones es el que falta; aquí se lo manejó como ausencia real y no como error. La tercera es la definición del objetivo de cumplimiento, que fijamos: alcanzar lo planificado; aunque por propia planta se puede tener un objetivo de umbral. Y la cuarta es qué hacer con los turnos extremos, que de momento están marcados. Poner esto por escrito no supone una debilidad del trabajo, sino más bien lo contrario. Es lo que permite que las decisiones se discutan y finalmente se respalden, y no que queden escondidas en el código.

CAPÍTULO 4: ANÁLISIS DE RESULTADOS

4.1. Configuración experimental

El modelado parte de la capa *Silver* descrita en el Capítulo 3, contenida en el archivo silver_production_clean_rows.csv. Ese conjunto reúne 27.253 registros a nivel de turno, con 51 columnas que cubren el período 2020-2026 e integran tanto los formatos históricos como las cargas recientes del formulario de planta. Antes de entrenar fue necesario depurar el conjunto en dos sentidos. Primero se excluyeron los registros con imputación crítica (los marcados con calidad de dato baja, 3.195 filas), porque en ellos las variables de planificación provienen de rellenos por mediana y no de un dato real del turno. Después se descartaron las filas sin alguno de los predictores categóricos o sin valor de la variable objetivo. Tras esa depuración quedan 21.816 registros utilizables para el objetivo de eficiencia y 21.747 para el de productividad. La diferencia entre los 27.253 declarados en el Capítulo 3 y los algo más de 21.800 que llegan al entrenamiento no es una pérdida de información, sino el resultado de reservar para el modelo únicamente los turnos con contexto de planificación completo y trazable.

Se trabajó con dos variables objetivo. La eficiencia del turno se define como el cociente entre piezas buenas y cantidad planificada, expresada en puntos porcentuales. La productividad se define como las piezas buenas por hora efectivamente productiva. Convenía señalar lo que ya se había señalado en el Capítulo 3, en el que la eficiencia fue reconstruida bajo el prisma de la definición única, homogénea para los seis años de estudio, no utilizando la columna de porcentaje del archivo de origen, ya que combinaba criterios incompatibles entre épocas.

La selección de predictores siguió un criterio estricto de ausencia de fuga de información. Solo se admitieron variables conocidas antes de que el turno comenzara: la máquina, el producto,

el operador principal, el turno, las horas planificadas, la cantidad planificada y los componentes cíclicos del calendario (mes, día, día de la semana, indicador de fin de semana, semana del año y trimestre). Quedaron deliberadamente fuera todas las magnitudes que solo existen una vez terminado el turno, como las piezas buenas, las defectuosas, el total producido, las horas de producción, la tasa de rechazo, el ciclo real y, por supuesto, las propias variables objetivo. Incluir cualquiera de ellas habría inflado artificialmente los resultados sin ningún valor predictivo real.

La partición de los datos es temporal, no aleatoria, en coherencia con el planteamiento del Capítulo 3 y con el uso previsto del sistema, que es anticipar el desempeño de turnos futuros. Los años 2020 a 2023 forman el conjunto de entrenamiento, el año 2024 funciona como validación y los años 2025 y 2026 quedan reservados como conjunto de prueba. El preprocesamiento se ajustó siempre con los datos de entrenamiento. Para la regresión lineal las variables categóricas se codificaron con one-hot, agrupando las categorías poco frecuentes, y las numéricas se estandarizaron; los modelos de árbol usaron codificación ordinal sin escalado. La comparación se apoyó además en validación cruzada temporal de cinco particiones, de modo que ningún modelo se juzgara por un único corte. Todo el proceso usa una semilla fija para garantizar la reproducibilidad.

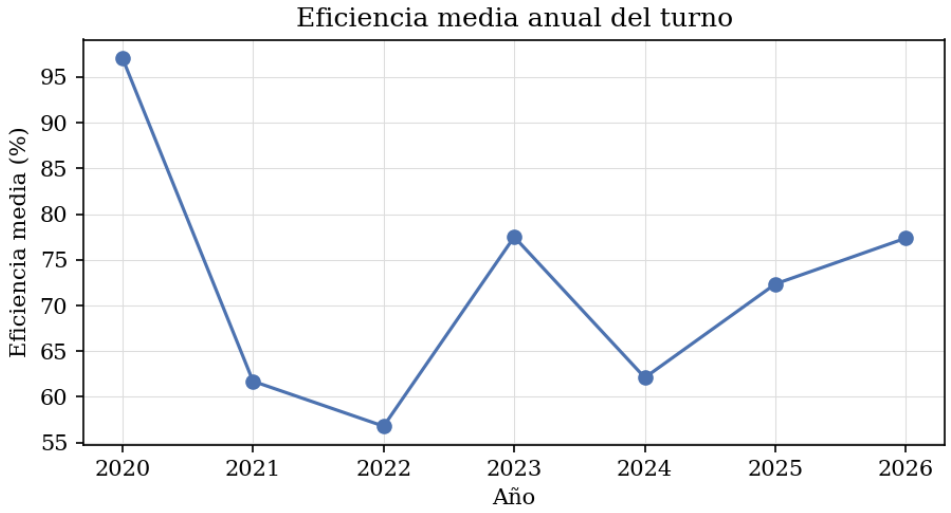
Las métricas se reportan en la escala original de cada objetivo mediante cuatro indicadores que en adelante se citan por sus siglas: el error absoluto medio o MAE, la raíz del error cuadrático medio o RMSE, el error porcentual absoluto medio o MAPE y el coeficiente de determinación R^2. El MAPE debe leerse con cautela en el caso de la eficiencia, porque una fracción considerable de turnos tiene eficiencia cercana a cero y la división por valores tan pequeños dispara el indicador. Por esa razón, para la eficiencia el análisis se apoya sobre todo en el MAE, el RMSE y el R^2.

4.2. Resultados por modelo

El primer hecho que ordena la lectura de los resultados aparece antes de cualquier modelo y se muestra en la Figura 15: la eficiencia media de la planta cambia de forma brusca de un año a otro. Pasa de rondar 97 puntos en 2020 a 62 en 2021, cae a 57 en 2022, se recupera a 77 en 2023 y vuelve a oscilar entre 62 y 77 en los años siguientes. Esta inestabilidad de fondo condiciona todo lo que viene después, porque significa que el nivel de eficiencia de un año no es un buen anuncio del nivel del año siguiente.

Figura 15

Eficiencia media anual del turno (2020-2026)



La regresión lineal múltiple no logra capturar la relación entre el contexto del turno y su eficiencia. Sobre el conjunto de prueba obtiene un MAE de 35,0 puntos y un R^2 de −0,13, es decir, predice peor que limitarse a usar la media. El comportamiento resulta esperable: el modelo lineal no es capaz de extrapolar en condiciones aceptables cuando hay una mezcla de categorías de alta cardinalidad junto con un objetivo con un comportamiento tan disperso como el que muestra la eficiencia.

El Random Forest mejora la situación en validación, alcanzando para este modelo un R^2 de 0,10 y un MAE de 32,7 puntos, pero es una mejora que no se mantiene en el conjunto de prueba, donde el R^2 vuelve de nuevo a bajar hasta a valores cercanos a cero. La Figura 16 lo muestra visualmente: la nube de puntos de real frente a predicho no escalan por la diagonal, sino que tienden a agruparse a modo de banda, hecho muy característico e indicador de que el modelo está prediciendo un valor medio independientemente del valor real. El mismo sentido tiene la Figura 17, donde para los residuos del modelo se pone de manifiesto una estructura clara que pone de manifiesto que parte de la varianza no está explicada.

Figura 16

Eficiencia: valores reales frente a predichos en prueba (Random Forest)

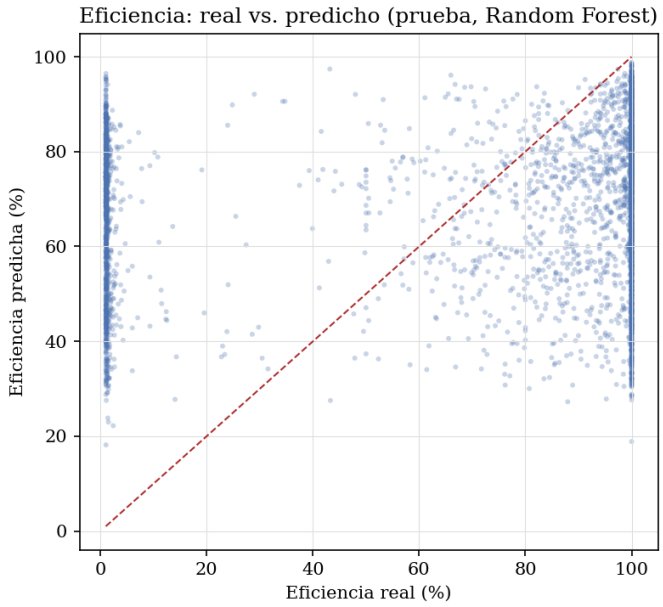
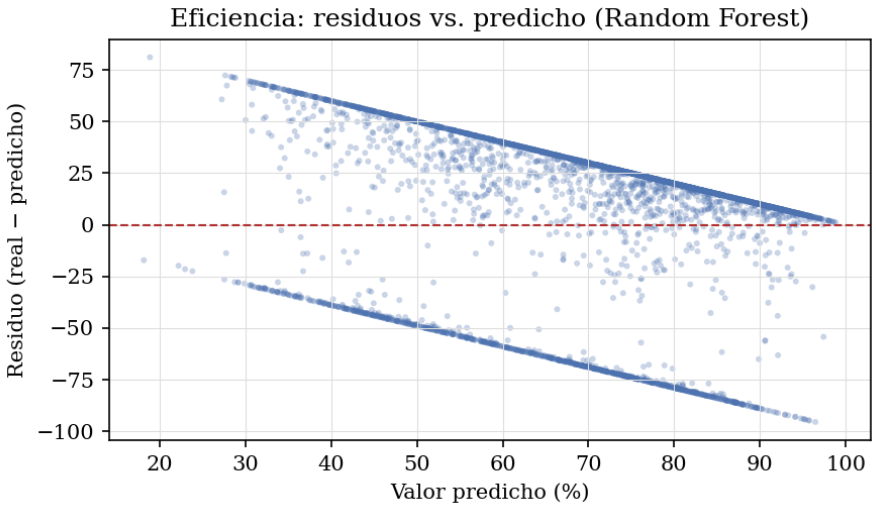


Figura 17

Eficiencia: residuos frente al valor predicho (Random Forest)



XGBoost tiene un comportamiento similar al Random Forest en validación, pero también se desmejora en la prueba. Los tres modelos, entonces, concuerdan en una misma conclusión incómoda pero honesta: con la información que se tiene antes de iniciar el turno, no se puede prever mejor la eficiencia de los años 2025 y 2026 que con un promedio. La razón no está en la elección del modelo o del algoritmo, sino en el cambio de régimen que se acredita en la Figura 15.

La situación cambia radicalmente si el objetivo que se desea modelar es la productividad y no la eficiencia. Dado que esta variable depende, básicamente, de la cadencia física de la máquina y del tipo de producto, su comportamiento en el tiempo es mucho más estable y, por tanto, más predecible. La productividad se modeló en escala logarítmica, para complementarla a partir de su importante asimetría, y luego las estimaciones se devolvieron a piezas/hora.

XGBoost resuelve este objetivo con holgura. Sobre el conjunto de prueba alcanza un R^2 de 0,71 y un MAE cercano a 197 piezas por hora. La Figura 18 muestra una nube de puntos que sigue la diagonal a lo largo de todo el rango, y los residuos de la Figura 19 se distribuyen de forma

mucho más equilibrada que en el caso de la eficiencia. La regresión lineal y el Random Forest, en cambio, no consiguen generalizar a los años futuros y entregan R^2 negativos, lo que vuelve a situar a XGBoost como el único modelo capaz de transferir lo aprendido al período de prueba.

Figura 18

Productividad: valores reales frente a predichos en prueba (XGBoost)

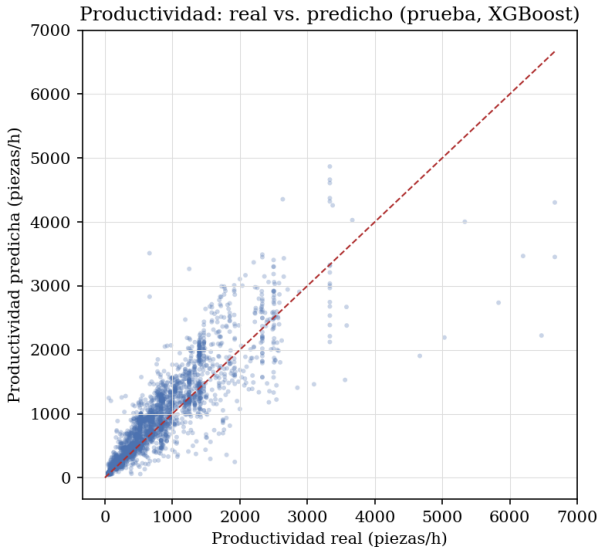
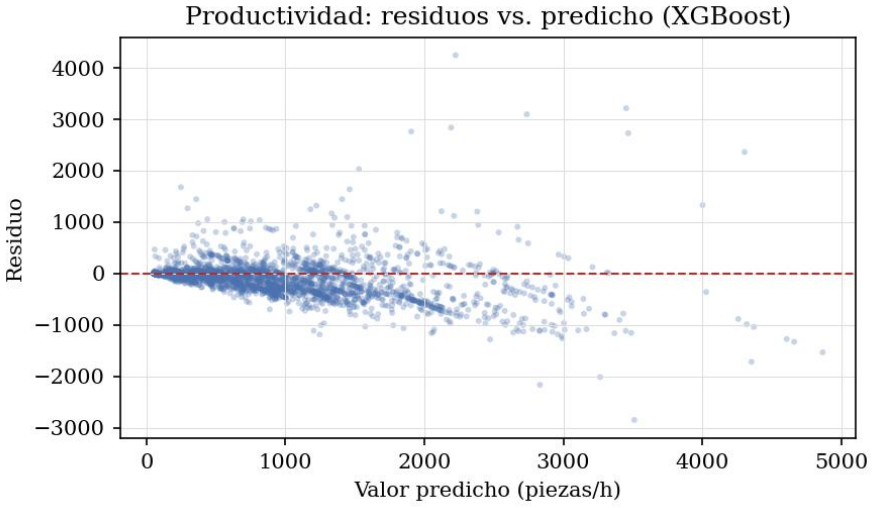


Figura 19

Productividad: residuos frente al valor predicho (XGBoost)



4.3. Comparación de modelos y selección

La Tabla 10 reúne el desempeño de los tres algoritmos sobre el objetivo de eficiencia, y la Tabla 11 hace lo propio con la productividad. La Tabla 12 detalla los hiperparámetros con los que se entrenó cada modelo.

Tabla 10

Comparación de modelos para la predicción de eficiencia (puntos porcentuales)

Modelo	MAE val.	R^2 val.	MAE prueba	RMSE prueba	R^2 prueba
Regresión Lineal Múltiple	34,7	−0,13	35,0	43,8	−0,13
Random Forest	32,7	0,10	35,2	41,8	−0,03
XGBoost	32,5	0,06	35,7	44,2	−0,15

Nota. MAE y RMSE en puntos porcentuales de eficiencia. Val. corresponde al conjunto de validación del año 2024 y prueba a los años 2025 y 2026. Elaboración propia.

Tabla 11

Comparación de modelos para la predicción de productividad (piezas por hora)

Modelo	MAE val.	R^2 val.	MAE prueba	RMSE prueba	R^2 prueba
Regresión Lineal Múltiple	245,0	−1,59	279,8	884,7	−0,81
Random Forest	577,3	−1,61	828,9	1334,9	−3,13
XGBoost	195,5	0,59	196,8	351,8	0,71

Nota. MAE y RMSE en piezas por hora. Val. corresponde al conjunto de validación del año 2024 y prueba a los años 2025 y 2026. La productividad se modeló en escala logarítmica y se reconvirtió a la escala original para el cálculo de las métricas. Elaboración propia.

Tabla 12

Hiperparámetros de cada modelo

Modelo	Hiperparámetros
Regresión Lineal Múltiple	Con intercepto; one-hot con agrupación de categorías raras y estandarización de variables numéricas
Random Forest	400 árboles, profundidad máxima 18, mínimo de 3 muestras por hoja, max_features = sqrt
XGBoost	600 árboles, tasa de aprendizaje 0,05, profundidad máxima 6, submuestreo 0,8, colsample_bytree 0,8, min_child_weight 5

Nota. Configuraciones fijadas sobre el conjunto de validación, con semilla aleatoria fija para asegurar la reproducibilidad del entrenamiento. Elaboración propia.

El modelo seleccionado para el sistema es XGBoost, y la justificación sale de las dos tablas anteriores. La Tabla 11 muestra que en productividad, que es el objetivo donde el aprendizaje automático aporta valor real, XGBoost es el único de los tres que generaliza al período de prueba, con un margen amplio sobre sus competidores. La Tabla 10, en cambio, deja ver que en eficiencia ninguno de los tres supera al promedio, de modo que ahí la elección del algoritmo es indiferente desde el punto de vista del error y conviene quedarse con el mismo modelo por coherencia y simplicidad de mantenimiento. Elegir XGBoost permite atender ambos objetivos con una sola familia de modelos, bien entendido que la predicción de productividad es la confiable y la de eficiencia se ofrece solo como referencia.

4.4. Importancia de variables e interpretación de negocio

La Figura 20 ordena las variables según su contribución al modelo de eficiencia, y la Figura 21 hace lo mismo para el de productividad. Ambas coinciden en lo esencial: las dos variables de planificación, las horas planificadas y la cantidad planificada, encabezan el ranking. Conforme al modelo de productividad, el peso conjunto es el más dominante y, a distancia, es seguido por la máquina. Dicha jerarquía tiene una lectura industrial clara. La cadencia de un turno la determina

buena parte lo que se planifica producir y la máquina asignada que fija la velocidad física del ciclo. El peso del producto, del operador y del turno está por debajo del peso conjunto, lo que se puede interpretar como que, una vez fijados el volumen objetivo y el equipo, el margen de variación explicable por la persona o por el turno es relativamente poco significativo.

Figura 20

Importancia de variables en el modelo de eficiencia (Random Forest)

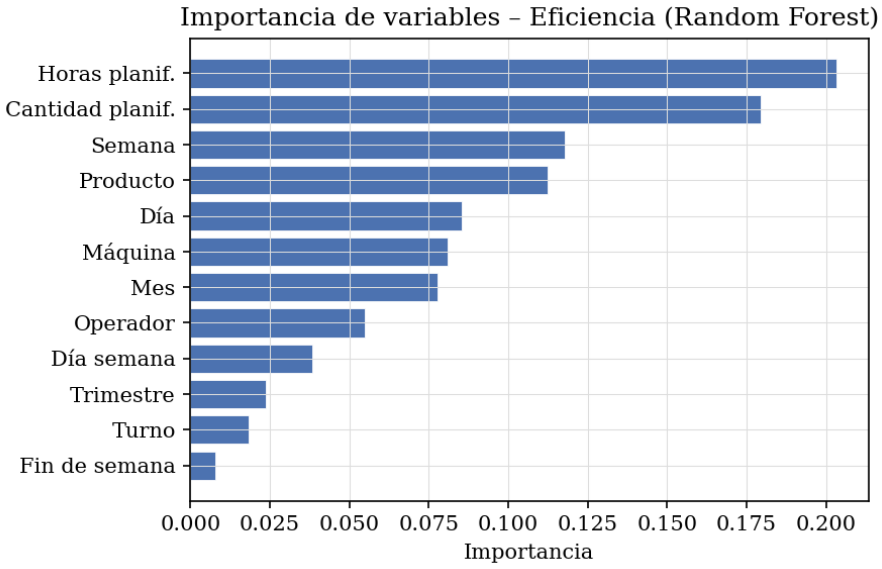
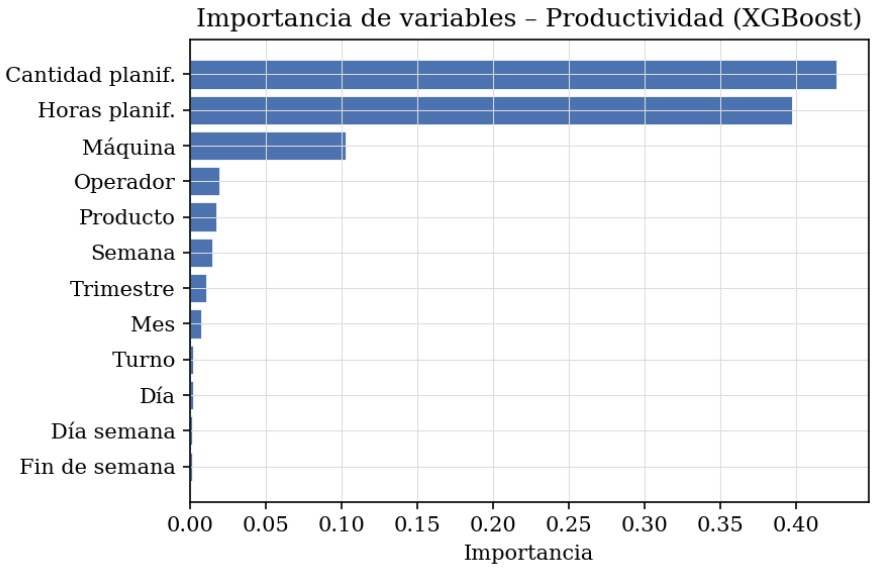


Figura 21

Importancia de variables en el modelo de productividad (XGBoost)



En la Figura 22 y en la Figura 23 se pone de manifiesto el error absoluto medio por año en el conjunto de prueba. En lo tocante a productividad el error se mantiene estrecho entre 2025 y 2026, lo que permite confirmar que el modelo es estable a través del tiempo. En lo que concierne la eficiencia el error se mantiene alto, en la línea de la dificultad anteriormente comentada.

Figura 22

Eficiencia: error absoluto medio por año en prueba

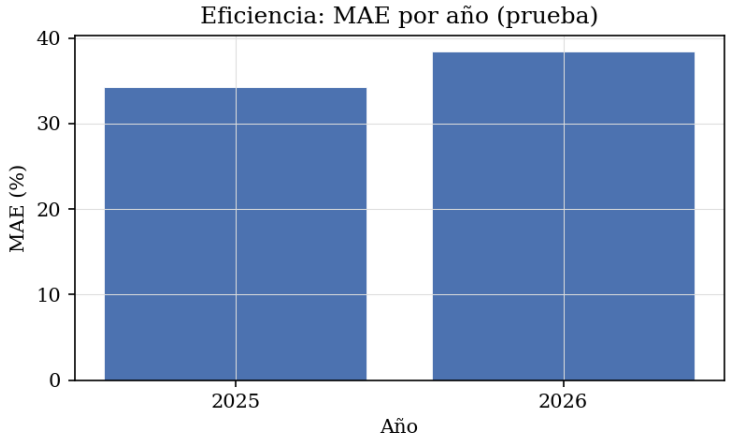
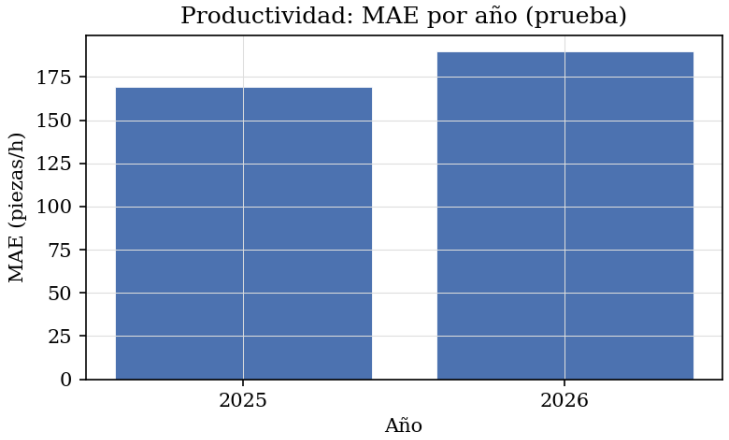


Figura 23

Productividad: error absoluto medio por año en prueba

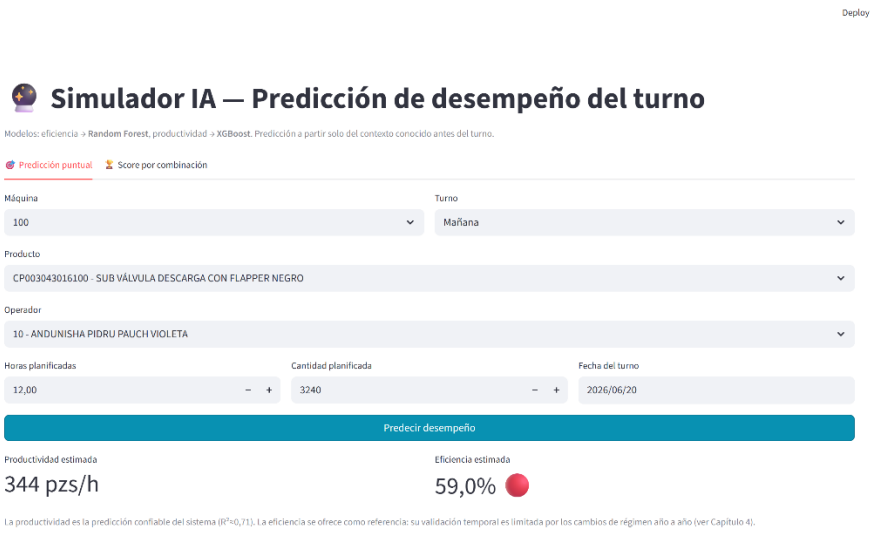


4.5. Integración en el simulador

El modelo que fue entrenado se cargó en la pestaña llamada «Simulador IA» de la app de gerencia, que en la versión anterior era un simple mecanismo de acceso. El simulador carga el modelo serializado más su preprocesador y funciona sin tener que ir a consultar la base de datos, respondiendo así de manera inmediata. La figura 24 muestra el modo de predicción puntual: el supervisor escoge una máquina, un producto, un operador y un turno, indica las horas y la cantidad planificadas, y el sistema devuelve la productividad estimada que debe considerarse como una predicción fiable más la eficiencia estimada a modo de referencia. La interfaz dice de manera explícita que la productividad es la predicción fiable y que la eficiencia debe ser considerada con precaución, lo cual hace que la herramienta no inculque una falsa sensación de seguridad.

Figura 24

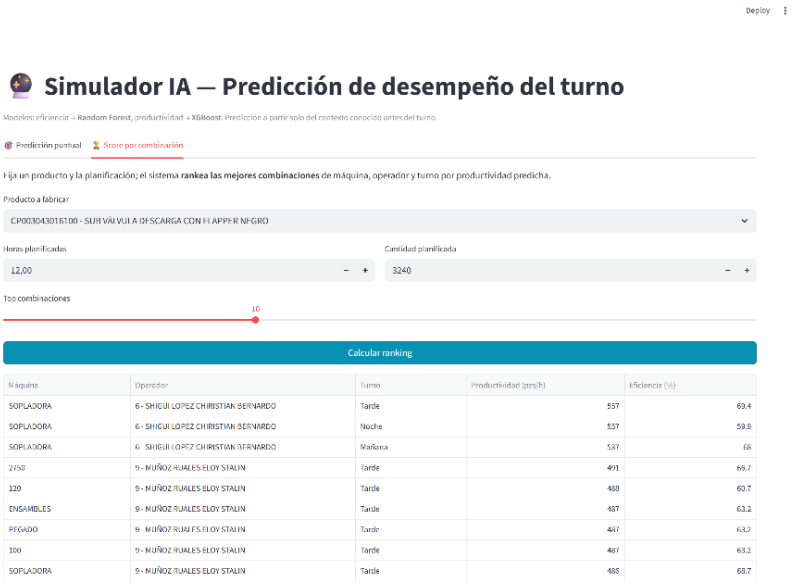
Simulador IA: predicción puntual de desempeño para una combinación máquina-producto-operador-turno



La Figure 25 ilustra la segunda función del simulador, que está pensada como una ayuda directa a la decisión en sí. Por medio de un producto y una planificación, el sistema evalúa un gran número de combinaciones de máquina, operador y turno y las jerarquiza por productividad predicha. Esta jerarquización transcribe el modelo en una recomendación concreta; ante una orden de fabricación, el supervisor sabe qué asignaciones podrían dar mejor cadencia y tendrá una partícula más de información para incluir en la programación de la planta. Es aquí donde el componente predictivo se encuentra con el objetivo de optimización que presenta el trabajo.

Figura 25

Simulador IA: ranking de combinaciones por productividad predicha (score por combinación)



4.6. Limitaciones y honestidad de los resultados

Los resultados deberían ser verídicos; la predicción de la eficiencia de los turnos desde el contexto anterior no se sostiene en una validación temporal: ninguno de los tres modelos logra superar el promedio de los años 2025 y 2026. Y no se trata de un error de implementación. Cuando se explora el mismo conjunto a partir de una partición aleatoria, el Random Forest logra alcanzar un R^2 cercano a 0,48, lo que señala que existe una señal transversal entre el contexto del turno y la eficiencia del mismo. Lo que no se sostiene es una estabilidad temporal de esa señal: la Figura 15 hace ver como el nivel medio de eficiencia varía fuertemente según el año de modo que un modelo entrenado en el pasado parte con una clara desventaja estructural para anticipar lo que ha de venir. Esta diferencia entre aleatoriedad y temporalidad es, a su vez, un resultado interesante, ya que pone de manifiesto que evaluar de una manera más realista es más costoso y dice la verdad.

La productividad, al estar anclada en la física del ciclo, sí se sostiene en el tiempo y por eso su modelo es el que se lleva al uso real. Conviene recordar, de todos modos, que el R^2 de 0,71 deja

una parte de la variación sin explicar, atribuible a factores que no figuran en los datos previos al turno, como las paradas no programadas, la variabilidad del material o el estado puntual de la máquina.

Deben considerarse tres limitaciones adicionales. La primera de ellas es la heterogeneidad de origen de los datos, que recogen un formato distinto a lo largo de los años y comportan imputaciones que, aun siendo controladas, son fuentes de ruido. La segunda es la elevada cardinalidad de productos y operadores, ya que los modelos aprenderán con pocos ejemplos por categoría en los casos menos comunes. La tercera limitación es que el sistema describe correlaciones históricas, no causas: es el ranking del simulador el que introducirá asignaciones asociadas en el pasado a un buen desempeño, no el que por sí mismo 'haga' que el desempeño mejore con un cambio de condiciones de planta.

4.7. Decisiones y supuestos

Las siguientes decisiones se tomaron de forma razonable ante puntos ambiguos y deben validarse con el equipo del proyecto.

• Eficiencia. Se utilizó piezas buenas sobre cantidad planificada (homogénea para los seis años), sin la columna de porcentaje del origen. Consideramos que es la única definición comparable a lo largo de todo el periodo.

• Exclusión del año como predictor. A pesar de que la consigna nos lo permitía, se había excluido el año como predictor porque, llegado a la partición del tiempo, los años de prueba no están en el entrenamiento y forzar al modelo a extrapolar fuera de rango. La estacionalidad tiene que ver con las cíclicas (mes, semana, trimestre), que sí se repiten del año.

• Variables disponibles no utilizadas. El diccionario establece el supervisor y el ciclo planificado, como información disponible antes del turno y potencialmente predictora. Pero se decidieron excluir respetando la lista de predictores pactada en Capítulo 3; pueden incorporarse si el equipo lo aprueba.

• Filtro de calidad. Se eliminaron 3.195 registros con imputación crítica. El conjunto de entrenamiento usable quedó en 21.816 para eficiencia y 21.747 para productividad.

• Operador secundario. Únicamente fue modelado el operador principal. El segundo operador estuvo ausente en la mayoría de turnos y contribuiría a obtener más ruido que señal.

• Umbral en el cumplimiento. La capa Silver marcaba el cumplimiento del plan cuando las buenas eran iguales o superiores a las planificadas (100 %), mientras que los tableros de gerencia reflejaban visualmente una meta del 95 %. Es conveniente unificar ambos criterios.

• Tratamiento de la productividad. Fue modelado en escala logarítmica y se recortaron los extremos inferior y superior bajo los percentiles 1 y 99 del entrenamiento, para evitar que un puñado de registros anómalos distorsionara el ajuste.

• Reproducibilidad. Todo el flujo usa una semilla aleatoria fija.

4.8. De la línea base a producción: por qué mejora la eficiencia

Hasta aquí el análisis solamente utilizaba el contexto que se contiene antes del turno, prescindiendo de la historia reciente de cada máquina, que también se registra. Y esto no fue una decisión casual, sino que es una opción deliberada: medir hasta qué punto se puede predecir con lo mínimo. Con esta base, la productividad bien, la eficiencia no. Pero el trabajo no quedó aquí, sino que en una segunda fase, con el sistema en producción, se añadieron variables de historia de

la máquina y la eficiencia pasó a predecirse bien. Aclarar qué cambió y qué no cambiaría viene a ser conveniente.

Primero, el modelo es el mismo; en relación a XGBoost en ambas fases. Lo que cambió son las variables; en la segunda fase se añadieron las variables de rezagos e históricos por máquina, todas calculables en los turnos previos como, por ejemplo, la eficiencia del turno anterior (efficiency_lag_1) o el promedio de los últimos 30 turnos (efficiency_roll_30). Con esas variables, el R^2 de la eficiencia en prueba pasa de ser un número tendiendo a cero (-0,15) a un 0,816. La mejora no viene por innovar con algoritmos distintos sino gracias a la información.

La razón es muy sencilla. La eficiencia tiene una gran tendencia. La manera en que ha trabajado una máquina en sus turnos recientes cambia bastante la manera en que trabajará en el siguiente; la línea base no tenía ese dato, tan sólo contemplaba el plan de tareas y el calendario, y con eso no llega a ser suficiente porque el nivel medio de eficiencia varía de un año para otro. Proporcionando al modelo la historia reciente, se obtiene la señal que en la primera fase no salía.

La Tabla 13 presenta la comparación de las dos fases según sus cifras reales. La productividad no se encuentra de nuevo modelizada para la producción, por lo que su cifra fiable es la que indica la línea base.

Tabla 13

Comparación de las dos fases del componente de aprendizaje automático

Fase	Objetivo	Features	Modelo	R^2 en prueba
Fase 1 (línea base, sin rezagos)	Productividad	Contexto pre-turno y calendario cíclico	XGBoost (XGBRegressor)	0,71

Fase 1 (línea base, sin rezagos)	Eficiencia	Contexto pre-turno y calendario cíclico	XGBoost (XGBRegressor)	-0,15
Fase 2 (producción, con rezagos)	Eficiencia (M1)	Contexto pre-turno más rezagos e históricos por máquina	XGBoost (XGBRegressor)	0,816
Fase 2 (producción, con rezagos)	Cumplimiento ≥ 95% (M2)	Contexto pre-turno más rezagos e históricos por máquina	XGBoost (XGBClassifier)	No aplica (clasificación)

Nota. R^2 calculado sobre el conjunto de prueba de cada fase. M1 y M2 corresponden a los modelos descritos en el Capítulo 5. Para M2, al ser un clasificador, el desempeño se evalúa con métricas de clasificación en dicho capítulo. Elaboración propia.

Faltan cosas por decir sin maquillaje. El salto de eficiencia que registra la Tabla 13 no es magia: depende sobre todo de efficiency_lag_1. Si esa variable falta, por ejemplo, en una máquina nueva o sin historial, el modelo pierde fuerza. Además, dos variables del modelo de producción hay que revisarlas. had_overtime se calcula como horas de producción mayores que las planificadas, y las horas de producción se saben al terminar el turno, así que podría estar colando información posterior. data_quality_score es una marca de calidad del dato, no algo del proceso, y podría estar separando por año o por formato de archivo. Lo correcto es justificarlas o volver a entrenar sin ellas y confirmar cuánto del 0,816 se mantiene.

El desarrollo minucioso de esta segunda fase, de las variables nuevas, de los modelos M1 y M2, del pipeline de datos y del simulador lo podemos encontrar en el capítulo de Arquitectura Técnica e Implementación; en este punto, simplemente basta dejar el puente: la línea base representa la referencia de lo que se puede predecir del contexto, y la ingeniería de variables temporales es lo que hace que dicha referencia alcance utilidad. La productividad no deja de mantener su cifra de referencia como R^2 de la línea base con 0,71.

CAPÍTULO 5: ARQUITECTURA TÉCNICA E IMPLEMENTACIÓN DEL SISTEMA

5.1. Descripción general del sistema

El sistema elaplasml combina un pipeline de datos estructurado en tres capas (Bronze, Silver, Gold) en línea con la Arquitectura Medallion (Databricks, 2021), una capa de modelos de Machine Learning construidos sobre XGBoost (Chen y Guestrin, 2016) y una aplicación web interactiva implementada sobre Streamlit Cloud cuyo principal objetivo es la transformación de registros históricos de producción guardados en archivos Excel heterogéneos en capacidad predictiva ejecutable por la gerencia de la planta.

El desarrollo de esta solución tecnológica se alinea con dos necesidades operativas fundamentales que la organización había identificado. En primer lugar, la empresa gestionaba el registro y análisis de la producción en forma manual (hojas de cálculo - Excel) lo que era un método altamente propenso a los errores humanos, a inconsistencia y a las demoras en la producción. Esta "debilidad" quedó salvada al digitalizar y estandarizar la toma de datos mediante un formulario validado en el desarrollo de la aplicación Web. En segundo lugar, el enorme volumen de datos históricos que la organización había compilado desde el 2020 hasta abril del 2026 no había sido explotado adecuadamente; y no había, por tanto, ningún plan de explotación del análisis de datos. La arquitectura implementada se apropia de este capital histórico no aprovechado y lo convierte en un motor de la inteligencia artificial (Simulador IA) por lo que permite a la gerencia pasar de un modelo gerencial reactivo a uno más predictivo que mejora la toma de decisiones.

El sistema opera en dos modalidades:

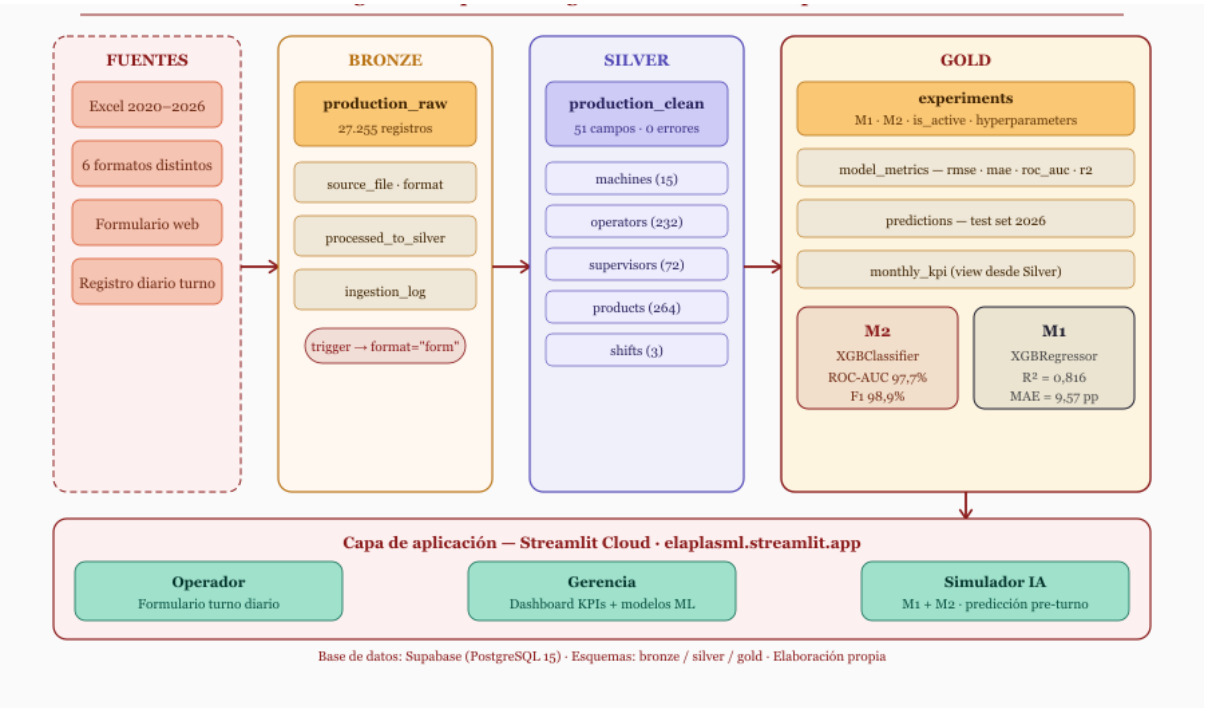
1. **Carga histórica:** mediante notebooks de Jupyter que procesan archivos Excel de períodos anteriores.

2. **Registro diario:** para ello, se sirve de un formulario web, que junto al operador va completando al término de cada turno, y que automáticamente desencadena el procesamiento hacia la capa Silver mediante un trigger de base de datos.

   Las dos modalidades comparten exactamente las mismas reglas de transformación y de validación, por lo que garantiza la coherencia entre los datos independientes del origen.

Figura 26

Visión general del sistema elaplasml. Elaboración propia



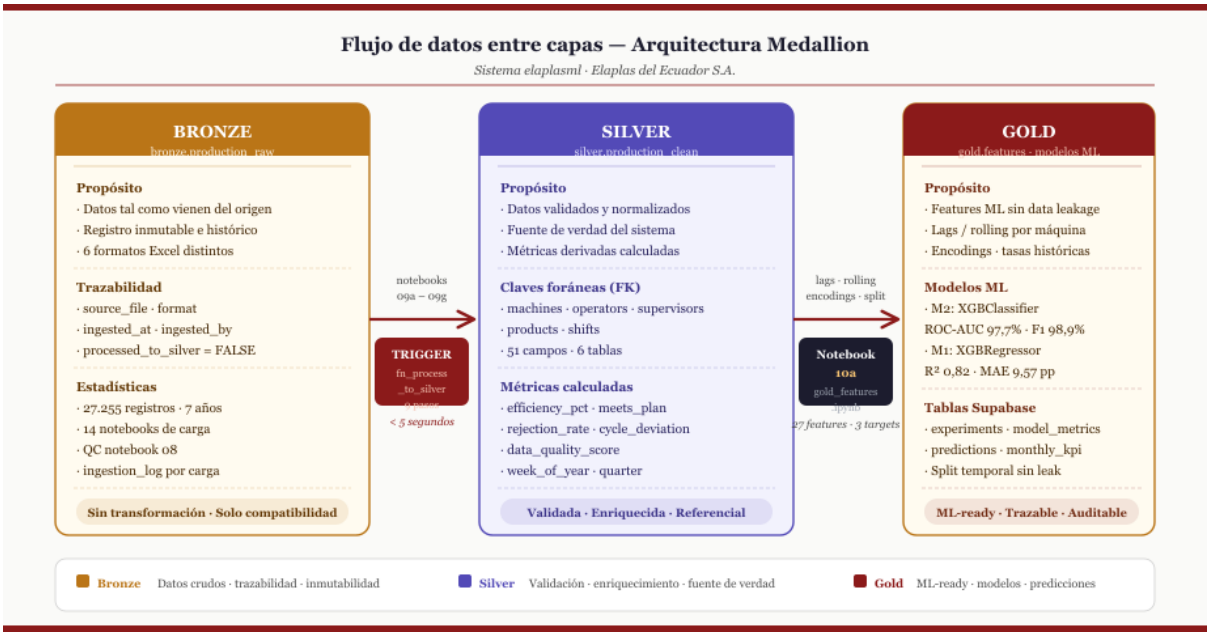
5.2. Arquitectura Medallion aplicada

La Arquitectura del tipo Medallion estructura el procesamiento de datos en una serie de capas de calidad progresiva, donde cada capa tiene las responsabilidades bien delimitadas (Chambers & Zaharia, 2018). Dicha separación de responsabilidades permite reprocesar cualquier capa sin el riesgo de afectar las demás, permite auditar el linaje de datos de forma sencilla y permite que

introducir nuevas fuentes o período de datos sea un proceso sencillo. El flujo de datos completo entre capas se recoge a continuación.

Figura 27

Flujo de datos entre capas de la Arquitectura Medallion. Elaboración propia.



5.2.1. Capa *Bronze*, ingesta y trazabilidad

La capa *Bronze* almacena los registros en su forma más cercana al origen, con las mínimas transformaciones necesarias para la compatibilidad con el esquema de base de datos. Su función es actuar como registro inmutable de la verdad histórica: una vez insertado un registro en Bronze, nunca se modifica ni elimina.

Se desarrollaron 14 notebooks de carga (01 a 07b) para procesar cada año y formato de archivo. Cada notebook implementa una función de carga idempotente que verifica la existencia previa de registros con el mismo formato antes de insertar, evitando duplicados. La capa *Bronze* contiene 27.255 registros históricos más los registros del formulario diario.

Tabla 14

Formatos de archivo procesados en la capa Bronze

Formato	Período	Notebooks	Características
Formato A	2020	01	Sin código SAP, operadores por nombre
Formato B/C	2021	02, 02b, 02c	Datos particionados en tres bloques
Formato D	2022	03, 03b	Incorpora código SAP parcial
Formato E	2023	04, 04b	Primera versión con good_pieces_kg
Formato F	2024-2026	05, 06, 07, 07b	Formato completo con todos los campos
Formulario web	Diario (2026+)	Trigger	Inserción en tiempo real vía API

Nota. Elaboración propia.

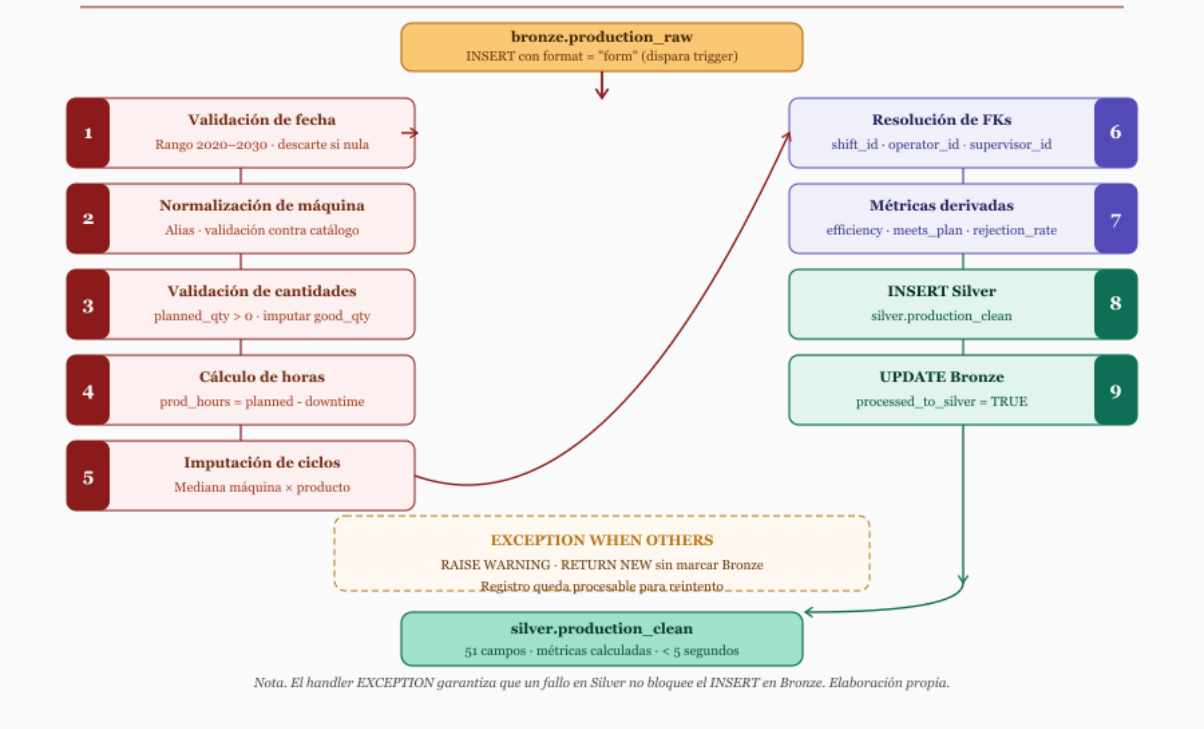
Cada registro en *Bronze* incluye los campos de trazabilidad source_file (nombre del archivo origen), format (versión del formato), ingested_at (marca temporal de carga) e ingested_by (origen del registro). El campo processed_to_silver indica el estado de procesamiento hacia la capa siguiente y permite reprocesar registros que fallaron sin afectar los ya procesados exitosamente.

5.2.2. Capa Silver, calidad y enriquecimiento

La capa *Silver* aplica un proceso de transformación determinista en nueve pasos, ejecutado mediante la función de base de datos fn_process_to_silver. Esta función actúa como trigger automático ante cada inserción con format = "form" (formulario diario) y como proceso batch para las cargas históricas. La automatización garantiza que todos los registros sigan exactamente las mismas reglas de calidad, independientemente de su origen.

Figura 28

Pasos del trigger fn_process_to_silver. Elaboración propia.



Nota. El handler EXCEPTION garantiza que un fallo en Silver no bloquee el INSERT en Bronze. Elaboración propia.

La capa *Silver* cuenta con seis tablas: **production_clean** (tabla de hechos con 51 campos) y cinco tablas de catálogo (machines, operators, supervisors, products, shifts) vinculadas mediante claves foráneas que garantizan la integridad referencial. Los catálogos fueron construidos a partir del análisis de los datos históricos y contienen 15 máquinas, 232 operadores, 72 supervisores, 264 productos y 3 turnos.

Gestión de datos maestros y actualización de Catálogos

La actualización de los catálogos en la capa *Silver* opera bajo un esquema de inserción dinámica. Las funciones helper silver.resolve_operator_id(), silver.resolve_supervisor_id() y silver.resolve_shift_id() verifican la existencia del registro en el catálogo antes de realizar la inserción y si el código no se encuentra, crean automáticamente el nuevo registro. Esto garantiza

que la incorporación de un nuevo operador, supervisor o máquina a la planta no requiera intervención de un administrador de base de datos: el primer formulario que registre ese actor crea su entrada en el catálogo correspondiente y todos los registros posteriores lo vinculan mediante clave foránea.

Para los productos con código SAP, el trigger verifica la existencia del código en silver.products antes de asignar la FK. Si el producto no existe en el catálogo (nuevo producto sin código SAP asignado aún), product_code se almacena como NULL en Silver, conservando la descripción textual del producto en el campo product_description para trazabilidad.

Esta estrategia permite que la planta opere sin interrupciones incluso cuando se introducen nuevos productos que aún no han sido dados de alta en el sistema SAP corporativo.

Figura 29

Diagrama entidad-relación del esquema de datos elaplasml. Elaboración propia

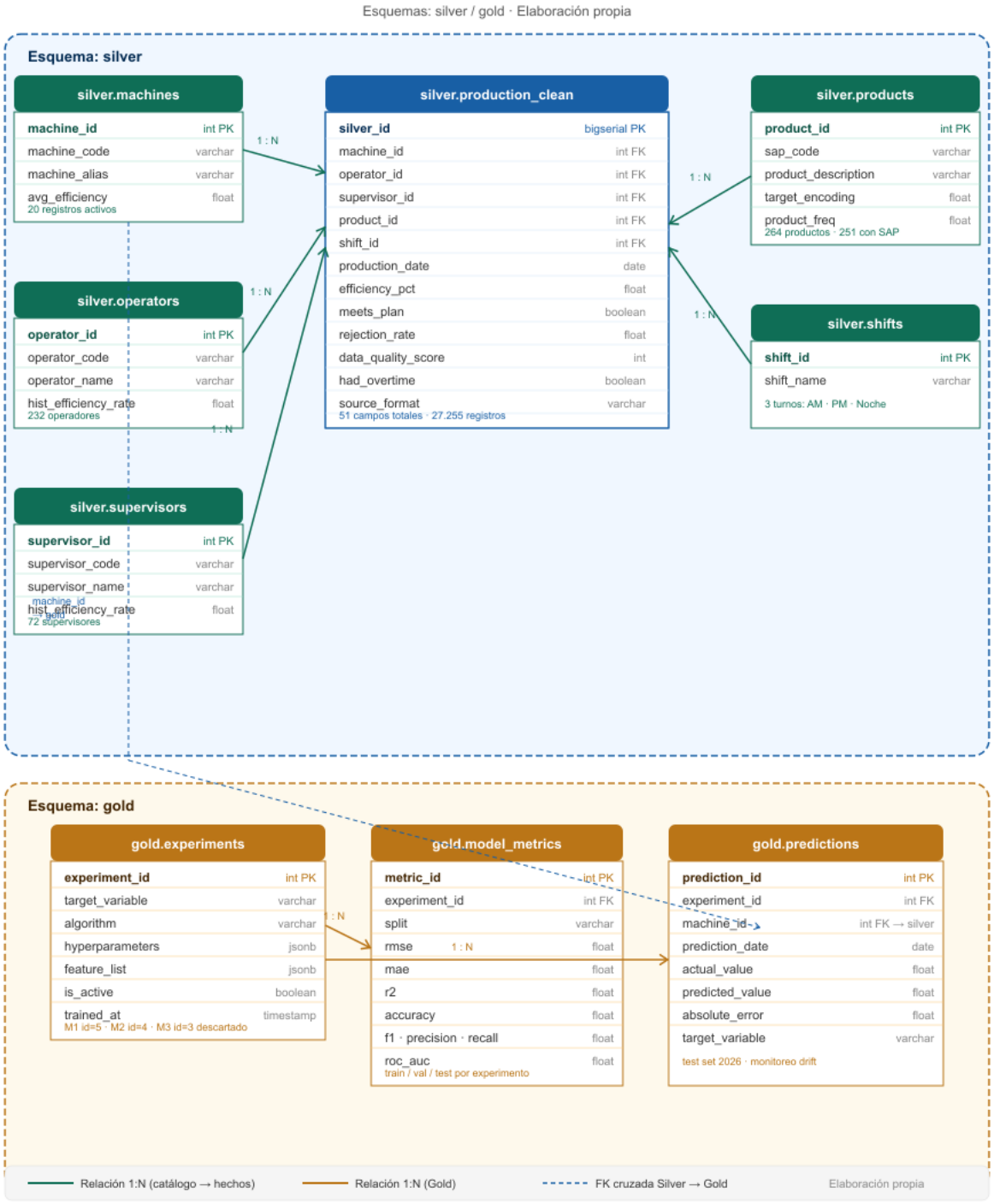


Tabla 15

Estadísticas de la capa Silver al cierre de esta versión

Indicador	Valor
Total de registros	27.255
Período de cobertura	2020, Junio 2026 (7 años)
Máquinas activas en catálogo	15
Operadores en catálogo	232
Productos con código SAP	251 de 264
Registros Formato F completo	10.531 (38,6%)
data_quality_score = 0 (completo)	10.523 (38,6%)
data_quality_score = 1 (imputación menor)	13.535 (49,7%)
data_quality_score = 2 (imputación crítica)	3.195 (11,7%)
Eficiencia promedio histórica	69,4%
Tasa de cumplimiento de plan (meets_plan)	46,4%
Errores de inserción en Silver	0

Nota. Elaboración propia a partir de silver.production_clean.

5.2.3. Capa Gold, features y modelos ML

La capa *Gold* contiene los datos preparados para el consumo de los modelos de Machine Learning, persistidos en formato Apache Parquet para lectura columnar eficiente, y los metadatos de los experimentos de modelado almacenados en Supabase. Esta capa no replica datos de *Silver* sino que los transforma en representaciones específicas para el aprendizaje automático.

Tabla 16

Tablas de la capa Gold en Supabase

Tabla / Vista	Tipo	Contenido
gold.experiments	Tabla	Metadatos de cada experimento: algoritmo, target, features, hiperparámetros, fechas, is_active
gold.model_metrics	Tabla	Métricas por split (train/val/test): RMSE, MAE, R^2, accuracy, F1, precision, recall, ROC-AUC
gold.predictions	Tabla	Predicciones del test set 2026: valor real, valor predicho, error absoluto, fecha, máquina
gold.monthly_kpi	Vista	KPIs mensuales por máquina calculados desde Silver: eficiencia, cumplimiento, paradas

Nota. Elaboración propia.

5.3. Pipeline de machine learning

El pipeline de ML transforma los datos de *Silver* en modelos predictivos desplegados en producción. Se estructura en cuatro etapas: ingeniería de features, partición temporal, entrenamiento y evaluación y registro de experimentos. Cada etapa está implementada en notebooks de Jupyter independientes, lo que permite re-ejecutar cualquier paso sin afectar los demás.

5.3.1. Ingeniería de features, capa Gold

La construcción de features sigue el principio fundamental de ausencia de filtración temporal (*data leakage*): cada observación solo accede a información disponible antes del momento que representa. Este principio, formalizado por Kaufman et al. (2012), es crítico en series temporales industriales donde la autocorrelación puede llevar a estimaciones de rendimiento artificialmente optimistas si no se respeta.

5.3.1.1. Features de lag y rolling

Calculadas por máquina, ordenadas cronológicamente, con desplazamiento mínimo de un turno (shift=1):

- efficiency_lag_1: eficiencia del turno inmediatamente anterior en la misma máquina (cobertura: 99,9%).
- efficiency_lag_7: eficiencia del séptimo turno anterior en la misma máquina.
- efficiency_roll_30: promedio de eficiencia de los 30 turnos previos (mínimo 5 observaciones).
- meets_plan_lag_1: indicador de cumplimiento del turno anterior (cobertura: 99,9%).
- meets_plan_roll_30: tasa de cumplimiento de los 30 turnos previos.
- real_cycle_roll_30: ciclo real promedio de los 30 turnos previos.
- cycle_deviation_roll_30: desviación de ciclo promedio de los 30 turnos previos.

5.3.1.2. Tasas históricas de actores

Calculadas exclusivamente sobre el conjunto de entrenamiento (≤ 2024) y aplicadas como tabla de referencia (lookup) al resto de conjuntos, evitando filtración del target (Micci-Barreca, 2001):

- operator_hist_rate: eficiencia promedio histórica del operador en el período de entrenamiento.
- supervisor_hist_rate: eficiencia promedio histórica del supervisor en el período de entrenamiento.

- machine_avg_efficiency: eficiencia promedio histórica de la máquina en el período de entrenamiento.
- product_target_enc: eficiencia promedio histórica asociada al producto (target encoding sobre train).

5.3.1.3. Encodings y variables temporales

- machine_enc: LabelEncoder ajustado sobre train, aplicado como lookup. Valor -1 para máquinas no vistas.
- product_freq: frecuencia relativa del producto en el conjunto de entrenamiento.
- Variables temporales: month, week_of_year, quarter, weekday, is_weekend, extraídas de la fecha.

5.3.1.4. Partición temporal

La partición respeta estrictamente la dimensión temporal. No existe solapamiento entre conjuntos y todas las features históricas se calculan únicamente con datos del conjunto de entrenamiento:

Tabla 17

Partición temporal del conjunto de datos

Conjunto	Período	Registros	Uso
Entrenamiento	2020, 2024	22.439 (82,3%)	Ajuste de parámetros, encodings y tasas históricas
Validación	2025	3.512 (12,9%)	Early stopping e hiperparámetros
Prueba	2026	1.302 (4,8%)	Evaluación final sin sesgo de optimización

Nota. Elaboración propia.

5.3.2. Justificación algorítmica, ¿por qué XGBoost?

Se eligió el modelo Extreme Gradient Boosting (XGBoost) tras realizar una fase de experimentación de comparación con los modelos base, los cuales en este caso son la Regresión Lineal Múltiple y el Random Forest, haciendo uso de la validación cruzada temporal (TimeSeriesSplit) y con evaluación conjunta hold-out. La elección de este modelo se encuentra debidamente justificada en cuatro razones particulares relacionadas con la técnica y el dominio manufacturero:

- Relaciones no lineales: la eficiencia lograda en un turno no depende linealmente de las horas de las mismas planificadas ni del ciclo real. XGBoost modela interacciones complejas entre features sin requerir transformaciones explícitas.
- Robustez ante valores atípicos: los tiempos de ciclo y las cantidades producidas presentan outliers frecuentes en los registros de planta. El entrenamiento por Gradient Boosting es mucho más robusto que la Regresión Lineal para las observaciones extremas.Manejo nativo de valores nulos: XGBoost aprende automáticamente la dirección óptima para datos

faltantes en cada nodo de árbol, lo que es crítico dado que ciclos, pesos y horas presentan valores nulos que no siempre pueden imputarse con certeza.

- Regularización integrada: los parámetros lambda y alpha de XGBoost controlan la complejidad del modelo, reduciendo el riesgo de sobreajuste en un dataset con alta dispersión (σ eficiencia = 41,7 pp).

> *Se eligió el algoritmo XGBoost, tras haber llevado a cabo una comparación experimental frente a la combinación de Regresión Lineal Múltiple y Random Forest usando la técnica de validación cruzada temporal.*
>
> *Su superioridad en validación y eficacia para la utilización de rezagos, obteniendo un R^2 de 0,834; y por ser el único algoritmo que generaliza en cuanto a la productividad, obteniendo un R^2 de 0,71 en el test frente a valores negativos de la Regresión Lineal y el Random Forest justifican dicha elección..*

5.3.3. Modelo M2, Clasificación de cumplimiento de plan

El modelo M2 predice si un turno cumplirá el plan de producción (meets_plan = efficiency_pct ≥ 95 % e implementa XGBClassifier con early stopping evaluado sobre la validación (mediante AUC como métrica); para el desbalanceo de clases (46,4 % positivos / 53,6 % negativos) se emplea scale_pos_weight = 1,15 sin aplicar técnicas de oversampling.

Tabla 18

Hiperparámetros del Modelo M2, XGBClassifier

Parámetro	Valor	Justificación
n_estimators	300 (efectivos: 93)	Early stopping sobre AUC validación
max_depth	6	Balance complejidad/generalización

learning_rate	0.05	Aprendizaje gradual, mayor estabilidad
subsample	0.8	Regularización por muestreo de filas
colsample_bytree	0.8	Regularización por muestreo de columnas
scale_pos_weight	1.15 (neg/pos)	Corrección de desbalance de clases
eval_metric	"auc"	Métrica de evaluación en validación
early_stopping_rounds	20	Detiene si AUC no mejora en 20 rondas
random_state	42	Reproducibilidad del experimento

Tabla 19

Métricas del Modelo M2, XGBClassifier (meets_plan)

Métrica	Entrenamiento	Validación	Prueba (2026)
Accuracy	99,1%	98,2%	97,9%
F1-Score	99,3%	98,8%	98,9%
Precision	99,2%	98,7%	98,8%
Recall	99,4%	98,9%	99,0%
ROC-AUC	99,8%	98,1%	97,7%
Best iteration	,	,	93 de 300

Nota. pp = puntos porcentuales. Elaboración propia.

La interpretación de la importancia dada a los features indica que meets_plan_lag_1 ocupa un 53,1% de la importancia total, por lo que parece confirmarse que la inercia en la operación, es decir, si el anterior turno cumplió con el plan, es el más significativo de todos los features para explicar el cumplimiento futuro. Las siguientes features en importancia son planned_hours (7,8%),

data_quality_score (5,7%), cycle_deviation (4,7%) y meets_plan_roll_30 (3,2%). El modelo se almacena en models/m2_meets_plan.joblib (455 KB) y se registra en gold.experiments (experiment_id activo).

5.3.4. Modelo M1, Regresión de eficiencia operativa

El modelo M1 estima el valor en continuo de efficiency_pct. La gran dispersión del target (media: 69,4%, σ: 41,7 pp, rango: 0-150%) constituye la mayor dificultad del modelo. Las predicciones finales no superan el rango [0; 1,5] mediante el uso posterior de np.clip a la inferencia, de acuerdo con el cap que ya se había aplicado en el trigger Silver.

Tabla 20

Hiperparámetros del Modelo M1, XGBRegressor

Parámetro	Valor	Justificación
n_estimators	300 (efectivos: 144)	Early stopping sobre RMSE validación
max_depth	6	Balance complejidad/generalización
learning_rate	0.05	Aprendizaje gradual, mayor estabilidad
subsample	0.8	Regularización por muestreo de filas
colsample_bytree	0.8	Regularización por muestreo de columnas
eval_metric	"rmse"	Métrica de evaluación en validación
early_stopping_rounds	20	Detiene si RMSE no mejora en 20 rondas
postprocess	np.clip(pred, 0, 1.5)	Limita predicciones al rango físico válido
random_state	42	Reproducibilidad del experimento

Tabla 21

Métricas del Modelo M1, XGBRegressor (efficiency_pct)

Métrica	Entrenamiento	Validación	Prueba (2026)
RMSE	9,82 pp	16,31 pp	17,75 pp
MAE	5,14 pp	9,01 pp	9,57 pp
R^2	0,944	0,834	0,816
Best iteration	,	,	144 de 300

Nota. pp = puntos porcentuales de eficiencia. MAPE no se reporta por presencia de valores cercanos a cero. Elaboración propia.

Las features más importantes en M1 son efficiency_lag_1 (21,5%) y data_quality_score (21,2%), seguidas de efficiency_roll_30 (8,1%) y had_overtime (7,6%). El modelo se almacena en models/m1_efficiency_pct.joblib (740 KB) y se registra en gold.experiments (experiment_id activo).

5.3.5. Modelo M3, tasa de rechazo (descartado)

Se evaluó un modelo de predicción de rejection_rate. El análisis de la distribución del target demostró que solo 10 del total de 10.531 registros con datos completos (has_full_data = TRUE) presentan tasa de rechazo superior a cero (0,1%). El modelo desarrollado alcanza un resultado de R^2 = -0,003 para el conjunto de prueba, lo que confirma su ausencia de habilidad predictiva. La decisión está reflejada en gold.experiments (experiment_id=3, is_active=FALSE), con una nota explicativa. El modelo se podrá reactivar cuando se disponga de un mínimo 500 eventos positivos de rechazo registrados.

5.3.6. Gestión de experimentos y versionamiento de modelos

La tabla gold.experiments actúa como un Model Registry transaccional. El mecanismo que utiliza un patrón de activación única: al compilar un nuevo modelo, el pipeline de entrenamiento automáticamente desactiva todas las versiones anteriores del mismo target_variable (UPDATE gold.experiments SET is_active = FALSE WHERE target_variable = "X") e inserta un nuevo registro con is_active = TRUE. Este mecanismo garantiza las tres propiedades operativas clave:

- El dashboard gerencial siempre lee tan sólo el experimento más reciente y activo, sin necesidad de modificar el código fuente de la aplicación.
- Cualquier nuevo reentrenamiento se hace de forma atómica: el modelo anterior se desactiva en la misma transacción en que se registra el nuevo.
- El historial de experimentos completo queda almacenado en la tabla is_active = FALSE, lo que permite auditar la evolución del rendimiento del modelo a lo largo del tiempo.

```
-- Patrón de activación única al reentrenar
UPDATE gold.experiments
   SET is_active = FALSE
 WHERE target_variable = 'meets_plan';   -- desactiva versión
anterior

INSERT INTO gold.experiments (..., is_active)
VALUES (..., TRUE);                              -- registra nueva
versión activa
```

La presente arquitectura aborda la problemática del Model Registry sin requerir herramientas externas (MLflow, W&B) y haciendo uso de la capacidad transaccional de PostgreSQL (Supabase) como backend de versionado.

5.3.7. Supabase como feature store ligero

En arquitecturas modernas de ML, un Feature Store centraliza el almacenamiento y el servicio de features históricas en tiempo real durante la inferencia; esto resuelve el problema de la asimetría entre entrenamiento e inferencia (Training-Serving Skew) (Kleppmann, 2017). En elaplasml, Supabase y la capa Silver ejercen esa función de forma muy ligera sin infraestructura adicional.

Cuando el Simulador IA requiere predecir un turno, consulta directamente la capa *Silver* mediante SQL para extraer los rezagos (efficiency_lag_1, meets_plan_lag_1) y promedios móviles (efficiency_roll_30, real_cycle_roll_30) exactos de esa máquina sobre sus registros históricos reales. Esto garantiza que las features de inferencia se calculan con exactamente los mismos datos y la misma lógica que las features usadas durante el entrenamiento, eliminando el Training-Serving Skew.

> Training-Serving Skew: error arquitectónico donde las features calculadas en producción (inferencia) difieren de las calculadas durante el entrenamiento, degradando el rendimiento real del modelo respecto al medido en evaluación.
>
> En elaplasml se resuelve porque *Silver* es la única fuente de verdad para ambas etapas: entrenamiento (10a) e inferencia (Simulador IA).

5.3.8. Arquitectura de inferencia, model serving

Para la inferencia en tiempo real del Simulador IA no se utilizó un servidor de API REST separado (como FastAPI o TorchServe). Esta decisión arquitectónica se tomó por dos razones: reducir la latencia de red entre la aplicación y el servidor de modelos y eliminar el costo de infraestructura adicional en el nivel gratuito del proyecto.

En su lugar, los artefactos serializados (.joblib) se embeben directamente en el contenedor de la aplicación *Streamlit*. Se utiliza una estrategia de caché de recursos mediante el decorador @st.cache_resource de *Streamlit*, que garantiza que los modelos M1 y M2 se carguen en la memoria RAM del servidor una única vez al iniciar el contenedor, y permanezcan en memoria para todas las solicitudes subsiguientes sin necesidad de recarga. Esto permite tiempos de predicción inferiores a 100 milisegundos por turno.

```
# Carga en memoria al inicio, un solo objeto para toda la sesión
@st.cache_resource
def load_models():
      m2 = joblib.load("models/m2_meets_plan.joblib")      # 455 KB
      m1 = joblib.load("models/m1_efficiency_pct.joblib") # 740 KB
      return m2, m1

# Inferencia < 100 ms por turno, sin llamadas a red
pred_cumple  = m2.predict(X_sim)       # → 0 o 1
prob_cumple  = m2.predict_proba(X_sim)[1]  # → probabilidad
pred_eficiencia = np.clip(m1.predict(X_sim), 0, 1.5) * 100
```

El artefacto .joblib contiene el estado completo del modelo XGBoost (árboles, hiperparámetros, estadísticas de entrenamiento y configuración de early stopping), lo que garantiza reproducibilidad exacta entre el entorno de entrenamiento y el entorno de producción.

5.3.9. Monitoreo de degradación y estrategia de re-entrenamiento

Los modelos de Machine Learning se degradan con el tiempo cuando el comportamiento de la planta cambia: incorporación de nuevas máquinas, cambio de materiales, nuevos operadores o variaciones estacionales no presentes en los datos de entrenamiento. Este fenómeno, conocido como Concept Drift, puede provocar que un modelo con alto rendimiento histórico produzca predicciones cada vez menos precisas sobre datos recientes sin que ningún mecanismo automático lo detecte.

A través de la tabla gold.predictions, que almacena el error absoluto para cada predicción, el sistema posibilita el seguimiento continuo. La política de mantenimiento angulosamente establece la re-evaluación (re-running) del modelo si:

1. El error absoluto medio mensual de M1 excede un umbral de tolerancia del 12% o
2. La precisión de M2 en los turnos del último mes cae por debajo del 90%.

De este modo, la consulta de seguimiento podría ejecutarse directamente sobre Supabase:

```
-- Monitoreo mensual de degradación
SELECT DATE_TRUNC('month', date) AS mes,
       AVG(absolute_error) AS mae_mensual
FROM   gold.predictions
WHERE  target_variable = 'efficiency_pct'
GROUP BY 1 ORDER BY 1 DESC;  -- alerta si mae_mensual > 12.0
```

Cuando se logran los umbrales establecidos, se procede a ejecutar el pipeline de reentrenamiento end to end (notebooks 10a a 10c) incorporando los nuevos registros acumulados en Silver como nuevo conjunto de datos de entrenamiento. El patrón de activación única que se ha comentado en la parte 5.3.6 garantiza que el nuevo modelo sustituye al anterior de forma transaccional, sin la necesidad de momento de inactividad del sistema. Se recomienda la frecuencia de revisitar el modelo como mínimo anual, a la vez que se cierra el año productivo de la planta.

Figura 30

Ciclo de vida MLOps

Nota. Representa el ciclo de vida MLOps implementado en elaplasml, desde la inferencia continua hasta el reentrenamiento por umbrales. Elaboración propia.

5.4. Arquitectura de la aplicación web

La aplicación web se lleva a cabo en Streamlit Cloud, que está conectado con el repositorio GitHub del proyecto a través del uso de un despliegue continuo automático: cada mutación que se realiza a la rama main provoca un nuevo despliegue sin tener que intervenir manualmente la

acción. La aplicación carga los datos de Supabase en tiempo real y carga los modelos joblib en la memoria en el arranque de la sesión.

5.4.1. Vistas y roles

Tabla 22

Vistas de la aplicación por rol de usuario

Rol	Vista	Función principal
Operador	Formulario de turno	Registra máquina, producto, cantidades, horas y paradas al finalizar cada turno. El sistema actualiza *Silver* en < 5 segundos.
Gerente	Rendimiento	KPIs operativos filtrados + métricas de modelos ML (ROC-AUC, F1, MAE, R^2) + gráficos de producción y operadores.
Gerente	Monitoreo	Matriz de cumplimiento diario: qué máquinas y turnos han reportado (✅/❌). Actualización en tiempo real.
Gerente	Histórico	Resumen anual de producción, eficiencia y paradas con gráficos comparativos de 7 años.
Gerente	Simulador IA	Predice eficiencia y cumplimiento de un turno antes de que ocurra. Ejecuta M1 + M2 con historial real de Silver.

Nota. Elaboración propia.

5.4.2. Seguridad y control de accesos

El sistema implementa control de acceso basado en roles (RBAC) a nivel de aplicación *Streamlit*, sin depender de Row Level Security (RLS) de PostgreSQL dado el modelo de plan gratuito de Supabase utilizado. El mecanismo funciona de la siguiente manera:

- **Autenticación:** el usuario ingresa credenciales (usuario y contraseña) en el formulario de login. Las credenciales se almacenan en el archivo secrets.toml de *Streamlit*, que se

configura como variable de entorno en *Streamlit* Cloud y nunca se expone en el repositorio GitHub.

- **Sesión:** tras la autenticación exitosa, el rol del usuario se persiste en st.session_state["user_role"]. *Streamlit* mantiene esta sesión de forma aislada por usuario mediante WebSocket.
- **Enrutamiento:** el módulo principal (app.py) evalúa el rol en session_state antes de renderizar cualquier vista. Un operador solo accede a show_operator_view(); un gerente a show_management_view(). El código de la vista contraria nunca se ejecuta ni se expone.
- **Cierre de sesión:** la función logout() limpia el session_state completamente, destruyendo el contexto de sesión y redirigiendo al formulario de login.

```
# Enrutamiento por rol, app.py
if st.session_state["user_role"] == "operador":
       show_operator_view()   # solo formulario de turno
elif st.session_state["user_role"] == "gerente":
    show_management_view() # dashboard + simulador IA
```

5.4.2.1. Privacidad de datos y copias de seguridad

La base de datos almacena nombres de operadores y supervisores, datos personales básicos sujetos a políticas de privacidad. En el contexto del sistema elaplasml, estos datos se utilizan únicamente para la resolución de claves foráneas y el cálculo de tasas históricas de rendimiento, sin exposición directa de nombres individuales en el dashboard público. Los identificadores mostrados al gerente corresponden a códigos de empleado, no a nombres completos.

La base de datos en Supabase cuenta con copias de seguridad automáticas gestionadas por la plataforma, que protegen el histórico de producción ante corrupciones o pérdidas accidentales.

El acceso directo a PostgreSQL está restringido mediante políticas de red de Supabase, permitiendo únicamente el tráfico autenticado proveniente de *Streamlit* Cloud mediante las credenciales almacenadas en secrets.toml. Ningún dato de producción se almacena en el repositorio GitHub ni en el entorno local de desarrollo, garantizando la separación entre código y datos sensibles.

5.4.3. Manejo de errores y tolerancia a fallos

El sistema implementa múltiples capas de resiliencia para garantizar la integridad de los datos ante errores del usuario o fallos de conectividad:

5.4.3.1. Validaciones en el cliente

- El formulario del operador valida que la cantidad de unidades buenas no supere la planificada antes de habilitar el botón de envío.
- Los botones de acción se desactivan inmediatamente al hacer clic, previniendo envíos duplicados (double submit). Esta lógica se implementa mediante st.session_state["form_saving"] = True seguido de st.rerun(), que oculta los botones antes de ejecutar el INSERT.
- Todos los campos obligatorios se verifican localmente antes de la llamada a la API de Supabase.

5.4.3.2. Atomicidad del trigger Silver

La transformación *Bronze* → *Silver* ocurre de forma atómica dentro del trigger de base de datos fn_process_to_silver. Si cualquier paso del proceso de nueve etapas falla (violación de FK, valor fuera de rango, error de cálculo), el handler EXCEPTION WHEN OTHERS captura el error y ejecuta RAISE WARNING + RETURN NEW. Esto garantiza dos propiedades:

- El INSERT en *Bronze* siempre es exitoso: el registro queda almacenado con processed_to_silver = FALSE, disponible para reprocesamiento manual posterior.
- El fallo en *Silver* no bloquea la operación del formulario: el operador recibe confirmación de grabado en *Bronze* y el sistema marca el registro para revisión.

5.4.3.3. Resiliencia de la conexión a Supabase

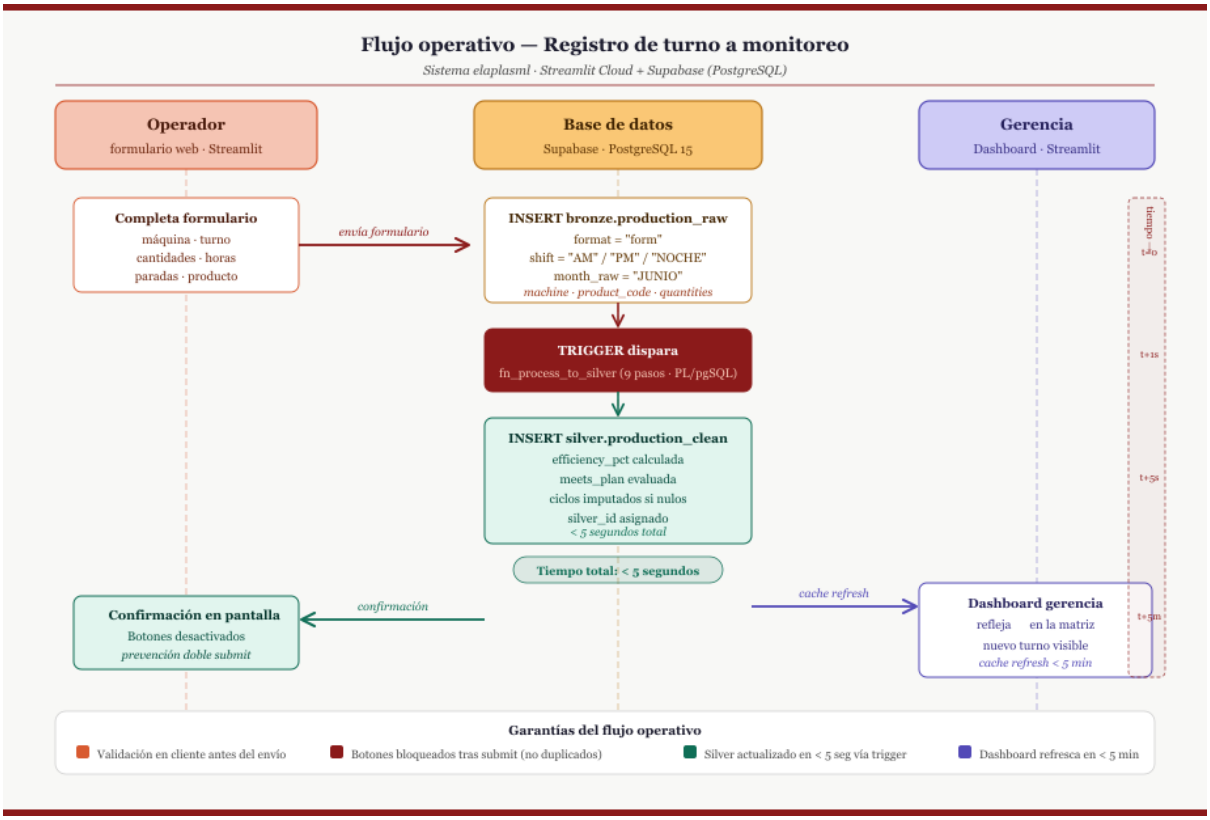
- La capa de datos usa el cliente oficial supabase-py con reintentos implícitos en el SDK. Los datos del dashboard se cachean con @st.cache_data(ttl=300), lo que permite que la aplicación sirva datos previos durante una caída temporal de conectividad de hasta 5 minutos sin impacto visible para el usuario.
- El plan gratuito de Supabase pausa el proyecto tras 7 días de inactividad. Se mantiene activo mediante un cron-job.org configurado para hacer ping a la aplicación cada 3 días.

5.4.4. Flujo operativo diario

El flujo operativo garantiza que un turno registrado por el operador esté disponible en el dashboard gerencial en menos de cinco segundos, sin intervención manual adicional:

Figura 31

Flujo operativo diario de registro a monitoreo. Elaboración propia



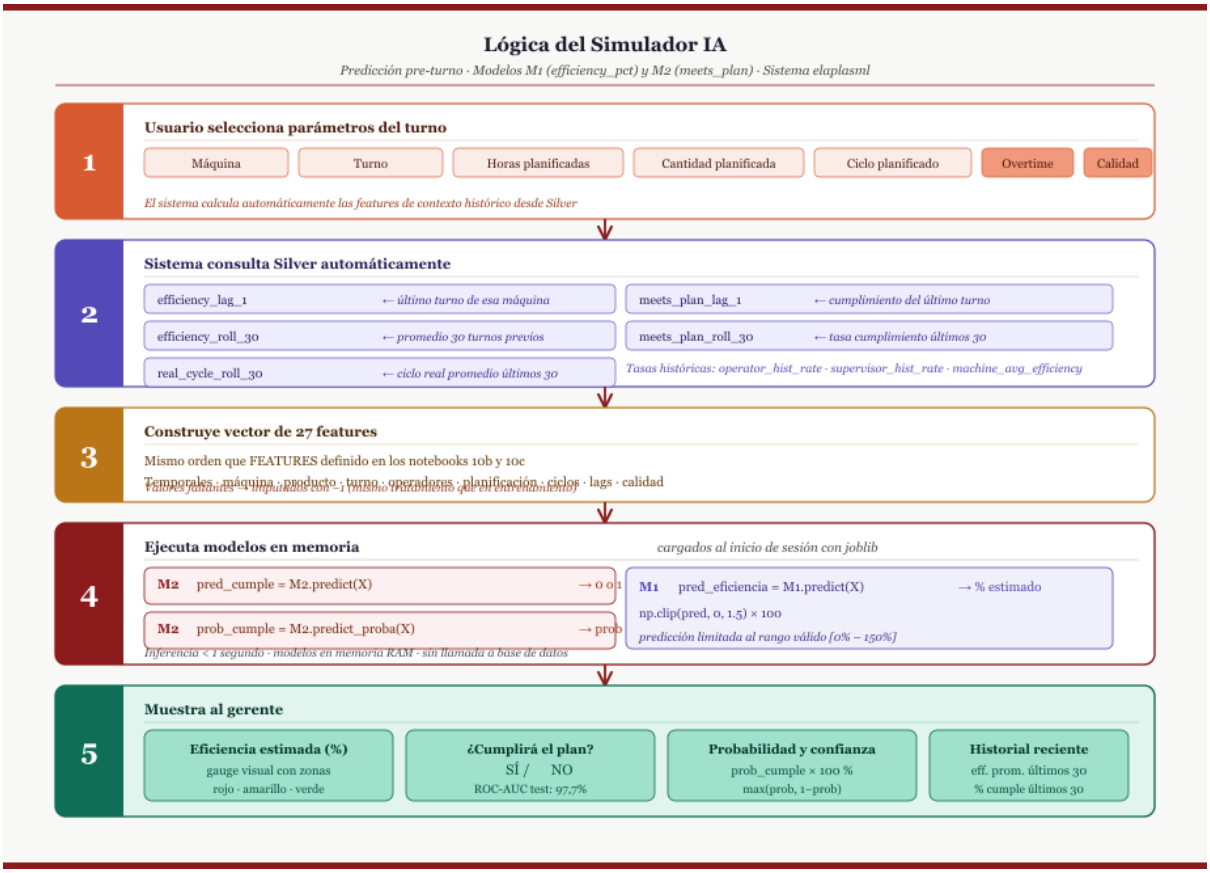
5.4.5. Simulador IA

El Simulador IA constituye el componente de mayor valor operativo del sistema. Permite a la gerencia anticipar el comportamiento de un turno antes de que ocurra, combinando los inputs del usuario con el historial real de la máquina extraído de *Silver* para construir el vector de features que alimenta los modelos M1 y M2.

LÓGICA DEL SIMULADOR IA

Figura 32

Lógica interna del Simulador IA. Elaboración propia.



5.5. Stack tecnológico

El sistema fue construido íntegramente sobre herramientas de código abierto y plataformas con modelo freemium, priorizando la integración nativa entre componentes, la comunidad activa de soporte y la capacidad de despliegue en producción con costo de infraestructura cero.

Tabla 23

Stack tecnológico del sistema elaplasml

Categoría	Herramienta	Versión	Rol en el sistema
Base de datos	Supabase / PostgreSQL	15.x	Almacenamiento *Bronze*, *Silver* y *Gold*. Triggers PL/pgSQL, funciones, esquemas múltiples.
Lenguaje	Python	3.14	Pipeline de datos, notebooks de carga y entrenamiento de modelos.
Machine Learning	XGBoost	2.0	Modelos M1 (XGBRegressor) y M2 (XGBClassifier) (Chen & Guestrin, 2016).
Machine Learning	scikit-learn	1.3	Preprocesamiento, LabelEncoder y métricas de evaluación (Pedregosa et al., 2011).
Datos	pandas / numpy	2.0 / 1.24	Transformaciones en la capa *Gold*: lags, rolling y encodings.
Serialización	joblib	1.3	Persistencia de modelos entrenados en formato binario para carga en producción.
Dashboard	*Streamlit*	1.32	Framework de despliegue de la aplicación web interactiva (Streamlit Inc., 2023).
Visualización	Plotly	5.18	Gráficos interactivos: barras, líneas, gauge de eficiencia y matrices.
Almacenamiento columnar	Apache Parquet/pyarrow	13.0	Formato de la capa *Gold* (gold_features.parquet). Lectura columnar optimizada.
Control de versiones	Git / GitHub	,	Repositorio privado con CD automático en *Streamlit Cloud* en cada push a main.
Notebooks	Jupyter	,	14 notebooks de carga *Bronze* + 4 notebooks *Gold* (features + 3 modelos).

Nota. CD = despliegue continuo. Elaboración propia.

5.5.1. Estrategia de despliegue continuo (CI/CD)

El sistema implementa un pipeline de despliegue continuo simplificado que conecta GitHub con *Streamlit Cloud* sin infraestructura de CI/CD adicional (Jenkins, GitHub Actions, Docker). La estrategia se basa en los siguientes principios:

- **Separación código/datos:** el repositorio GitHub versiona exclusivamente el código fuente de la aplicación, los notebooks y los artefactos de modelos (.joblib). Los datos de producción residen en Supabase y nunca se incluyen en el repositorio.
- **Modelos como artefactos estáticos:** los archivos .joblib (m1_efficiency_pct.joblib: 740 KB, m2_meets_plan.joblib: 455 KB) se versionan junto con el código en la carpeta models/. Esto garantiza que el entorno de producción en *Streamlit Cloud* sea inmutable y exactamente replicable: el modelo que se ejecuta en producción es bit a bit idéntico al que se evaluó durante el entrenamiento.
- **Despliegue automático:** *Streamlit Cloud* monitorea la rama main del repositorio mediante webhook. Cada push desencadena automáticamente: (1) reconstrucción del entorno Python con requirements.txt, (2) carga de secrets.toml desde las variables de entorno configuradas en el panel de *Streamlit Cloud*, y (3) reinicio del servidor de la aplicación. El tiempo de redespliegue es inferior a 2 minutos.
- **Rollback:** revertir a una versión anterior se realiza mediante git revert o git reset sobre la rama main, seguido de un push que desencadena el redespliegue automático al estado anterior.

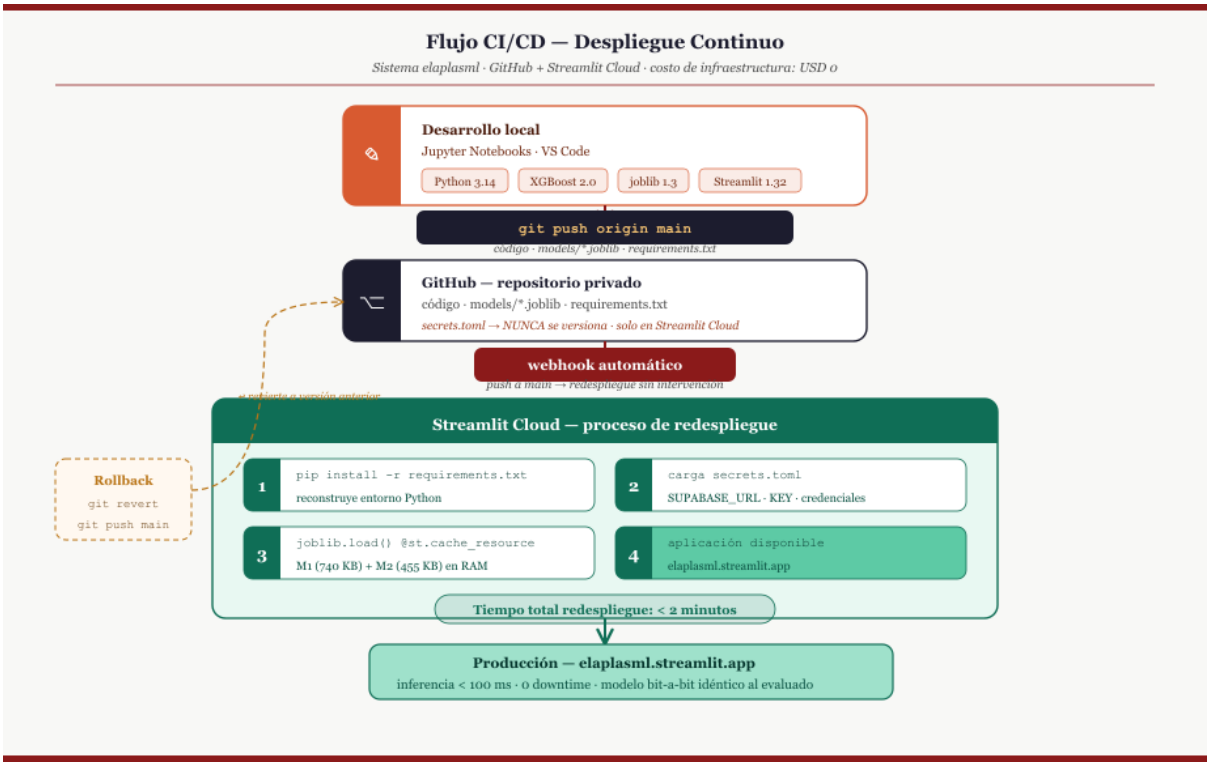
5.5.2. Análisis de límites y escalabilidad

Dado que la arquitectura se apoya en capas gratuitas, existen límites técnicos conocidos que deben considerarse ante un crecimiento sostenido del volumen de datos. Supabase restringe la base de datos a 500 MB en su plan gratuito, y *Streamlit Cloud* limita la memoria RAM del contenedor a 1 GB. Actualmente, siete años de operación (27.255 registros, 51 campos en Silver) ocupan una fracción mínima de estos límites gracias al diseño relacional normalizado: *Silver* pesa aproximadamente 18 MB y los modelos joblib suman 1,2 MB, muy por debajo de los umbrales establecidos.

Si el volumen de datos llegara a exceder estos límites en el futuro, la arquitectura está completamente desacoplada para facilitar la migración. El código de la aplicación, los notebooks y los modelos son independientes de la infraestructura de base de datos: migrar a una instancia de pago de PostgreSQL, desplegar el contenedor de *Streamlit* en AWS EC2 o Google Cloud Run o escalar a un plan superior de Supabase se realizaría sin necesidad de reescribir el código base, modificando únicamente las variables de entorno de conexión (SUPABASE_URL y SUPABASE_KEY en secrets.toml).

Figura 33

Flujo CI/CD Despliegue Continuo. Elaboración propia.



CAPÍTULO 6: RESULTADOS Y VALIDACIÓN

6.1. Rendimiento de los modelos

Ambos modelos demuestran un rendimiento sólido y consistente sobre el conjunto de prueba (año 2026, datos no vistos durante el entrenamiento). La mínima diferencia entre los conjuntos de validación y prueba confirma que la estrategia de partición temporal fue correctamente implementada y que no existe sobreajuste.

El modelo M2 (clasificación) exhibe comportamiento excepcionalmente estable, con diferencias inferiores a un punto porcentual entre validación y prueba en todas las métricas. Este resultado es atribuible al alto peso predictivo de meets_plan_lag_1 (53,1%), una feature que captura la inercia operacional con cobertura de 99,9% en el conjunto *Gold*.

El modelo M1 (regresión) presenta una brecha más amplia entre entrenamiento (R^2=0,944) y prueba (R^2=0,816), resultado de la alta dispersión del target (σ=41,7 pp). Sin embargo, un R^2=0,82 sobre datos de producción real de un año no incluido en el entrenamiento es consistente con lo reportado en la literatura para problemas similares de eficiencia productiva (Wuest et al., 2016). La comparación con estudios recientes ayuda a dimensionar las cifras. Dobra y Jósvai (2023) reportan errores inferiores al 1% al estimar el OEE en líneas de ensamblaje, pero lo hacen sobre horizontes semanales agregados y con procesos de cadencia estable; el R^2 de 0,816 obtenido aquí corresponde a un problema más fino, la predicción turno a turno en un proceso con cambio de régimen anual, donde la agregación no amortigua la variabilidad. La coincidencia en el fondo está en el método: tanto en este trabajo como en el de Kiangala y Wang (2021), los ensambles de árboles con potenciación superan las alternativas clásicas cuando la performance depende de interacciones entre máquina, producto y contexto, que es lo que precisamente nos hemos

encontrado en las Tablas 10 y 11. De forma práctica, un MAE de 9,57 puntos de eficiencia significa que la predicción puntual se desvía menos de diez puntos del valor real del turno, un margen suficiente para que el simulador pueda ordenar asignaciones alternativas, que es el uso que la planta le da el simulador, aunque no para reemplazar el control estadístico del proceso.

Los anteriores indicadores técnicos obtienen su verdadero valor cuando se expresan de acuerdo a la derivada de la decisión del gerente. Un F1-Score de 98,9% en M2 significa que de cada 100 turnos que el sistema pronostica como cumplidores del plan, menos de dos se corresponderán con falsos positivos: el gerente de planta podrá fijar las fechas de entrega logística con un respaldo en la información estadística, que reduce la incertidumbre en la operación diaria que históricamente estaba presente y dependía de la toma de decisiones aleatoria del supervisor. De la misma manera, un MAE de 9,57 puntos porcentuales en M1 sobre los datos del periodo del año 2026, que se encuentra fuera del periodo de estudio, significa que la eficiencia real de un turno se aparta de lo que el Simulador IA había previsto en cifras raramente superiores a los 10 puntos en porcentajes: con suficiente reconocimiento para que la gerencia detecte con suficiente anticipación los turnos en vías de una baja eficiencia y redistribuya recursos de forma anticipada a su ocurrencia.

En definitiva, ambos modelos hacen un desplazamiento sobre la toma de decisiones de la reacción a los hechos consumados hacia la anticipación fundamentada, que es precisamente el objetivo central de un sistema elaplasml.

6.2. Validación del flujo end-to-end

El sistema fue validado mediante una prueba de integración completa que cubre todos los componentes de la arquitectura en secuencia:

1. El formulario del operador crea un registro en la tabla Bronze, situando shift = "AM", month_raw = "JUNIO" y machine = "160" (código de catálogo).
2. El trigger fn_process_to_silver procesa el registro en menos de cinco segundos. Se ejecutan los nueve pasos de transformación.
3. El registro aparece en silver.production_clean con silver_id generada, efficiency_pct calculada, meets_plan evaluada y ciclos imputados donde corresponde.
4. La matriz de monitoreo del dashboard gerencial recoge la aparición del nuevo turno con indicador en la celda máquina 160 / turno MAÑANA sobre la fecha actual.
5. El Simulador IA genera predicciones coherentes con el historial de la máquina: sí, más confianza para máquinas con parámetros recientes, menos confianza para máquinas con pocos registros y/o antiguos.

6.3. Comportamiento del Simulador IA

Tabla 24

Resultados del Simulador IA en dos escenarios de validación

Parámetro	Escenario A, Máquina activa	Escenario B, Máquina sin datos recientes
Máquina	160	100
Último registro en Silver	29/04/2026	23/04/2021
Eficiencia promedio (30 últ.)	76,9%	87,1%
Cumple plan (30 últ.)	100,0%	26,7%

Eficiencia predicha (M1)	92,5%	50,7%
¿Cumple plan? (M2)	SÍ	NO
Probabilidad cumplimiento	95,6%	45,5%
Confianza del modelo	95,6% (alta)	54,5% (baja)
Interpretación	Historial reciente sólido → predicción optimista con alta certeza.	Sin datos recientes, bajo historial → predicción conservadora con baja certeza.

Nota. El comportamiento diferenciado entre escenarios confirma que el modelo captura correctamente la señal de inercia operacional. Elaboración propia.

6.4. Implementación y uso del sistema

El sistema se usa desde una aplicación web alojada en *Streamlit Cloud*, sin instalar nada más que un navegador actualizado. Tiene dos roles. El operador registra la producción de cada turno y el gerente de planta consulta los indicadores y usa el Simulador IA. El acceso es con usuario y contraseña, y como no hay recuperación automática, el restablecimiento lo hace el administrador.

Vista del operador

El operador llena un formulario de tres secciones que se completan en orden. Primero los datos del turno (fecha, turno, máquina y operador), luego el reporte del producto (horas y cantidades planificadas y reales, piezas buenas y defectuosas) y, si la hubo, una parada con su motivo. Antes de guardar, el sistema valida los campos, muestra una tarjeta de resumen y bloquea los duplicados, es decir, no deja registrar dos veces la misma combinación de fecha, turno, máquina y operador. Los datos no se guardan hasta confirmar.

Vista de gerencia

El gerente entra a un panel con cuatro pestañas. Rendimiento muestra los indicadores clave del período filtrado por máquina, producto, operador y turno, junto con las métricas de los modelos, con opción de ver el detalle por entrenamiento, validación y prueba y de descargar los datos en Excel. Monitoreo da la vista del día. Histórico resume el desempeño por año. La pestaña Simulador IA es la que conecta con esta tesis.

El Simulador IA predice la eficiencia y el cumplimiento de plan de un turno antes de que ocurra. El usuario fija la combinación de máquina, producto, operador y turno con su planificación, y el sistema consulta el historial reciente de esa máquina para armar los rezagos y devolver la eficiencia estimada, con un velocímetro de color, y la probabilidad de cumplir el plan. Sus dos funciones, la predicción puntual y el ranking de combinaciones por productividad, son las que se muestran en las Figuras 24 y 25 del Capítulo 4. El manual de usuario completo incluye además los requisitos de navegador, las preguntas frecuentes y el soporte técnico.

CAPÍTULO 7: CONCLUSIONES, LIMITACIONES Y TRABAJO FUTURO

Conclusiones

El proyecto cumple con la función para ayudar la decisión de asignar recursos al arranque de cada turno aplicando modelos de regresión. En las dos fases del trabajo el mejor modelo era XGBoost, de manera que el sistema se construyó sobre un solo algoritmo y la comparación con Regresión Lineal Múltiple y Random Forest justificó dicha elección.

La productividad se predice correctamente. Con el solo contexto que se tiene antes del turno, XGBoost alcanza un $R^2 = 0.71$ en el conjunto de testeo. Este es el resultado que respalda el Simulador IA al ordenar combinaciones de máquina - operador - turno por productividad esperada.

La eficiencia es un caso distinto. En ella no se puede avanzar hacia el futuro desde el contexto base porque su nivel medio presenta variaciones importantes de un año a otro, y un modelo entrenado con el pasado va a partir en desventaja. Esto no quiere decir que no haya relación: con particiones aleatorias el Random Forest alcanza un R^2 próximo al 0.48. La cuestión es la transferencia a través del tiempo, no la falta de señal.

La segunda fase resuelve esta limitación a través de la ingeniería de variables y no con un nuevo modelo. El mismo XGBoost, ahora con rezagos e históricos por máquina (efficiency_lag_1, efficiency_roll_30 y otros) lleva la eficiencia a un R^2 en test de 0.816. La mejora viene de la información que se le agrega al modelo. A esto hay que añadir el clasificador M2 que predice si el turno va a cumplir el plan y que, además, mantiene su comportamiento entre validación y test.

El sistema se encontró en producción. El pipeline cumple con una arquitectura Medallion (Bronze, Silver y Gold) en Supabase, corriendo la aplicación en Streamlit Cloud y versionando los modelos como artefactos. El simulador realiza la consulta de la capa Silver a la hora de la

predicción, trayendo los rezagos exactos de cada máquina, evitando así el desajuste entre el entrenamiento y el servicio.

Limitaciones

El cambio de eficiencia depende fundamentalmente de efficiency_lag_1. Para una máquina nueva o sin historial reciente, el modelo pierde fuerza, por lo que es bueno poder describir la forma en la que se comporta el sistema en ese caso.

Dos variables del modelo de eficiencia quedan por revisar. had_overtime se calcula con las horas de producción, que se conocen al terminar el turno, así que podría estar usando información posterior. data_quality_score es una marca de calidad del dato y no una variable del proceso, y podría estar separando registros por año o por formato de origen. Lo correcto es justificarlas con un argumento claro o volver a entrenar sin ellas y confirmar cuánto del 0,816 se mantiene.

Las métricas publicadas deben salir de la misma definición de variables y de la misma partición temporal en todo el documento y en el simulador, para que un mismo indicador no aparezca con dos cifras distintas. La productividad mantiene su R^2 de 0,71 como cifra de referencia de la línea base, y la eficiencia se reporta con la salvedad de que solo es confiable con los rezagos de la segunda fase.

Trabajo futuro

Quedan varias líneas abiertas. Reentrenar el modelo de eficiencia sin had_overtime ni data_quality_score y comparar el resultado. Recuperar el modelo de tasa de rechazo cuando haya suficientes registros, que fue lo que lo dejó fuera. Incorporar el supervisor del turno y el ciclo planificado como predictores, ya que se conocen antes del turno y hoy no se usan. Y extender la validación a nuevos años para vigilar si el régimen de la eficiencia se estabiliza o vuelve a cambiar.

REFERENCIAS

Aggarwal, C. C. (2017). Outlier analysis (2.a ed.). Springer. https://doi.org/10.1007/978-3-319-47578-3

Amat, J. M. (2020). Machine learning para análisis de datos con Python y R... CreateSpace Independent Publishing Platform.

Breiman, L. (2017). Classification and regression trees. Routledge.

Chambers, B., & Zaharia, M. (2018). Spark: The definitive guide. O'Reilly Media.

Chapman, P., Clinton, J., Kerber, R., Khabaza, T., Reinartz, T., Shearer, C., y Wirth, R. (2000). CRISP-DM 1.0: Step-by-step data mining guide. SPSS Inc.

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

Databricks. (2021). Introducing the *Medallion* Architecture. Databricks Documentation. https://docs.databricks.com/lakehouse/medallion.html

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

Géron, A. (2017). Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems. O'Reilly Media.

GitHub. (s.f.). GitHub: Let’s build from here. https://github.com/

Hastie, T., Tibshirani, R., y Friedman, J. (2009). The elements of statistical learning (2.ª ed.). Springer.

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

James, G., Witten, D., Hastie, T., Tibshirani, R., y Taylor, J. (2023). An Introduction to Statistical Learning: with Applications in Python. Springer.

Kaufman, S., Rosset, S., Perlich, C., & Stitelman, O. (2012). Leakage in data mining: Formulation, detection, and avoidance. ACM Transactions on Knowledge Discovery from Data, 6(4), 1-21. https://doi.org/10.1145/2382577.2382579

Kleppmann, M. (2017). Designing data-intensive applications. O'Reilly Media.

Kuhn, M., & Johnson, K. (2019). Feature engineering and selection: A practical approach for predictive models. CRC Press. https://doi.org/10.1201/9781315108230

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825-2830.

Mailund, T. (2017). Introduction to Data Science: A Python Approach to Machine Learning. Springer.

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

Project Jupyter. (s.f.). Project Jupyter: Home. https://jupyter.org/

Rahm, E., & Do, H. H. (2000). Data cleaning: Problems and current approaches. IEEE Data Engineering Bulletin, 23(4), 3–13. http://sites.computer.org/debull/A00dec/rahm.pdf

Streamlit. (s.f.). Streamlit: A faster way to build and share data apps. https://streamlit.io/

Supabase. (s.f.). Supabase: The open source Firebase alternative. https://supabase.com/

Sculley, D., Holt, G., Golovin, D., Davydov, E., Phillips, T., Ebner, D., Chaudhary, V., Young, M., Crespo, J. F., & Dennison, D. (2015). Hidden technical debt in machine learning systems. Advances in Neural Information Processing Systems, 28, 2503–2511. https://proceedings.neurips.cc/paper/2015/hash/86df7dcfd896fcaf2674f757a2463eba-Abstract.html

Tukey, J. W. (1977). Exploratory data analysis. Addison-Wesley.

VanderPlas, J. (2017). Python Data Science Handbook: Essential Tools for Working with Data. O'Reilly Media.

Wuest, T., Weimer, D., Irgens, C., & Thoben, K. D. (2016). Machine learning in manufacturing: Advantages, challenges, and applications. Production & Manufacturing Research, 4(1), 23-45. https://doi.org/10.1080/21693277.2016.1192517

Zhu, J., Zou, H., Rosset, S., y Hastie, T. (2009). Multi-class AdaBoost. Statistics and Its Interface, 2(3), 349-360.

Dobra, P., & Jósvai, J. (2023). *Cumulative and Rolling Horizon Prediction of Overall Equipment Effectiveness (OEE) with Machine Learning*. Big Data and Cognitive Computing, 7(3),

138.https://doi.org/10.3390/bdcc7030138

PDF: PDF artículo MDPI

Dobra, P., & Jósvai, J. (2022). *Assembly Line Overall Equipment Effectiveness (OEE) Prediction from Human Estimation to Supervised Machine Learning*. Journal of Manufacturing and Materials Processing, 6(3), 59. https://doi.org/10.3390/jmmp6030059

PDF: PDF artículo OEE Prediction

Kiangala, K. S., & Wang, Z. (2021). *An effective adaptive customization framework for small manufacturing plants using XGBoost and Random Forest ensemble learning algorithms in an Industry 4.0 environment*. Machine Learning with Applications, 4, 100024. https://doi.org/10.1016/j.mlwa.2021.100024

Artículo: ScienceDirect artículo XGBoost manufactura

Longard, L., Prein, T., & Metternich, J. (2023). *Intraday forecasting of OEE through sensor data and machine learning*. Procedia CIRP, 120, 93–98. https://doi.org/10.1016/j.procir.2023.08.017

Artículo: ScienceDirect forecasting OEE

Zhang, Y., Lin, R., Zhang, H., & Peng, Y. (2022). *Vibration prediction and analysis of strip rolling mill based on XGBoost and Bayesian optimization*. Complex & Intelligent Systems, 9, 133–145. https://doi.org/10.1007/s40747-022-00795-6

PDF: Springer PDF XGBoost industrial

Chai, T., & Draxler, R. R. (2014). Root mean square error (RMSE) or mean absolute error (MAE)? –

Arguments against avoiding RMSE in the literature. Geoscientific Model Development, 7(3), 1247-1250.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. International

Journal of Forecasting, 22(4), 679-688.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An Introduction to Statistical Learning (Vol.

112). Springer.

Willmott, C. J., & Matsuura, K. (2005). Advantages of the mean absolute error over the root mean

square error in assessing average model performance. Climate Research, 30(1), 79-82.

ANEXO A: DICCIONARIO DE DATOS (MAPA DE ESQUEMAS)

Índice de tablas:

Esquema	Tabla / Archivo	Descripción
BRONZE	`bronze.production_raw`	44 campos
BRONZE	`bronze.ingestion_log`	8 campos
SILVER	`silver.production_clean`	51 campos · tabla de hechos principal
SILVER	`silver.machines`	7 campos · 20 registros
SILVER	`silver.operators`	11 campos · 232 registros
SILVER	`silver.supervisors`	8 campos · 72 registros
SILVER	`silver.products`	5 campos · 264 registros
SILVER	`silver.shifts`	8 campos · 3 registros
GOLD	`gold.experiments`	15 campos · model registry
GOLD	`gold.model_metrics`	13 campos · métricas por split
GOLD	`gold.predictions`	13 campos · monitoreo drift
GOLD	`gold.monthly_kpi`	14 campos · vista calculada

ANEXO B: DESPLIEGUE DE LA APLICACIÓN

Github:

https://github.com/plaprodml-admin/elaplasml/blob/main/README.md

Base de datos:

https://supabase.com/dashboard/sign-in

Aplicación:

https://elaplasml.streamlit.app/

ANEXO C: MANUAL DE USUARIO

ELAPLAS DEL ECUADOR S.A.

Sistema de Control de Producción

MANUAL DE USUARIO

elaplasml · Streamlit Cloud

Versión:	1.0 - Junio 2026
Plataforma:	Web (*Streamlit Cloud*)
Usuarios:	Operadores y Gerentes de Planta

Índice de contenidos

Índice de tablas

Índice de figuras

1. Introducción

El Sistema de Control de Producción de Elaplas del Ecuador S.A. es una aplicación web diseñada para registrar, monitorear y analizar la producción diaria de la planta. Funciona directamente desde el navegador y no requiere instalación de software especializado.

1.1 ¿Para qué sirve?

- Registrar la producción por turno: máquina, producto, operador, cantidades y paradas.
- Monitorear en tiempo real si los turnos del día han sido reportados.
- Analizar la eficiencia histórica por año, mes, máquina y turno.
- Predecir el rendimiento de un turno antes de que ocurra, usando modelos de IA entrenados con datos reales de la planta.

1.2 Roles del sistema

Tabla 1

Roles del sistema

Rol	Función principal	Acceso
Operador	Registrar producción del turno	Formulario de registro
Gerente de Planta	Ver dashboards, reportes y predicciones	Panel completo con 4 secciones

2. Requisitos previos

La aplicación es 100% web. No necesitas instalar Python, bases de datos ni ningún framework. Solo necesitas un navegador actualizado y conexión a internet.

2.1 Navegadores compatibles

Tabla 2

Listado de navegadores

Navegador	Versión mínima	Recomendado
Google Chrome	110 o superior	Sí
Microsoft Edge	110 o superior	Sí
Mozilla Firefox	110 o superior	Sí
Safari (Mac/iOS)	16 o superior	Funcional
Internet Explorer	Cualquier versión	No compatible

2.2 Instalación del navegador-Windows

Google Chrome (recomendado)

- Abre Microsoft Edge (viene preinstalado en Windows).
- Ve a: https://www.google.com/chrome
- Haz clic en "Descargar Chrome".

- Ejecuta el archivo descargado (ChromeSetup.exe) y sigue el asistente.
- Una vez instalado, abre Chrome y escribe la URL de la aplicación en la barra de direcciones.

2.3 Instalación del navegador-Mac

Google Chrome (recomendado)

- Abre Safari (viene preinstalado en Mac).
- Ve a: https://www.google.com/chrome
- Haz clic en "Descargar Chrome".
- Abre el archivo .dmg descargado.
- Arrastra el ícono de Chrome a la carpeta Aplicaciones.
- Abre Chrome desde Launchpad o Spotlight (Cmd + Espacio, escribe "Chrome").

2.4 Conexión a internet

- Se requiere conexión a internet activa en todo momento, la aplicación no funciona sin conexión.
- Velocidad mínima recomendada: 5 Mbps de descarga.
- La aplicación está alojada en *Streamlit Cloud*.
- No requiere VPN ni configuración de red especial.

3. Acceso al sistema

3.1 Ingresar a la aplicación

- Abre tu navegador (Chrome, Edge o Firefox).
- Escribe o pega la URL en la barra de direcciones:

https://elaplasml.streamlit.app

- Presiona Enter. Verás la pantalla de inicio de sesión.

3.2 Pantalla de inicio de sesión

En la pantalla de login verás el logo de Elaplas, el nombre del sistema y dos campos:

Figura 1

Login de acceso al sistema

- Usuario: ingresa el nombre de usuario que fue asignado por el administrador.
- Contraseña: ingresa la contraseña.
- Completa los campos y haz clic en el botón "Ingresar".
- Si los datos son correctos, el sistema cargará automáticamente la vista correspondiente a tu rol. Existen dos roles: operador y gerente. Cada uno accede a una vista diferente.
- Si aparece el mensaje "Usuario o contraseña incorrectos", verifica que no haya espacios adicionales y que el Bloq Mayús esté desactivado.

3.3 Cerrar sesión

Para cerrar sesión de forma segura:

Figura 2

Cierre de sesión de la aplicación

- En el panel lateral izquierdo, hacer clic en el botón "Cerrar sesión".
- El sistema regresará a la pantalla de inicio de sesión.

4. Vista Operador-Registro de Producción

Esta vista está disponible para usuarios con rol Operador. Permite registrar los datos de producción de cada turno de trabajo.

4.1 Estructura del formulario

El formulario está dividido en tres secciones que deben completarse en orden:

Tabla 4

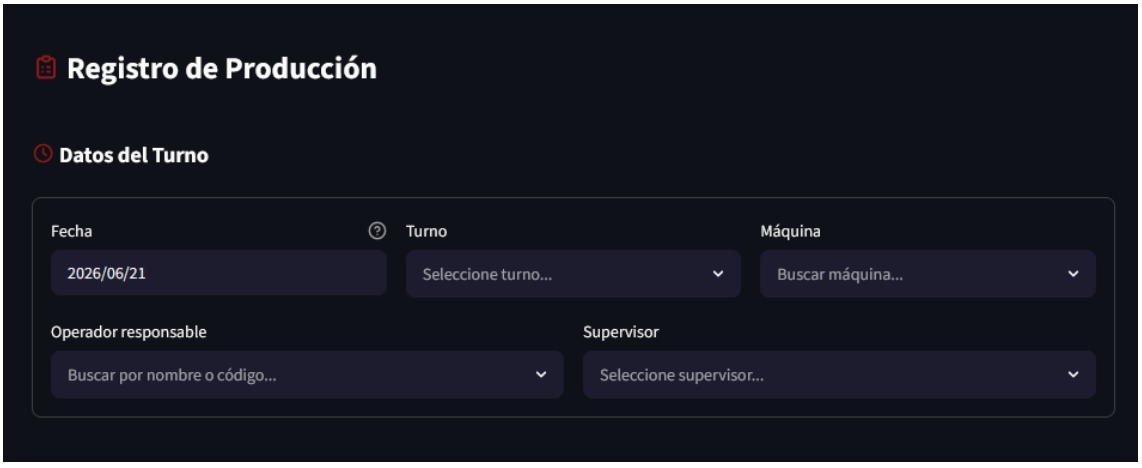
Formulario registro de productos

Sección	Campos incluidos
Datos del Turno	Fecha, Turno, Máquina, Operador responsable, Supervisor
Reporte de Producto	Producto fabricado, Horas planificadas, Unidades planificadas, Unidades buenas, Peso piezas buenas (kg), Peso purga (kg)
Parada (opcional)	Horas de parada, Motivo de parada

4.2 Sección 1-Datos del Turno

Figura 3

Registro del turno desde el módulo del operador



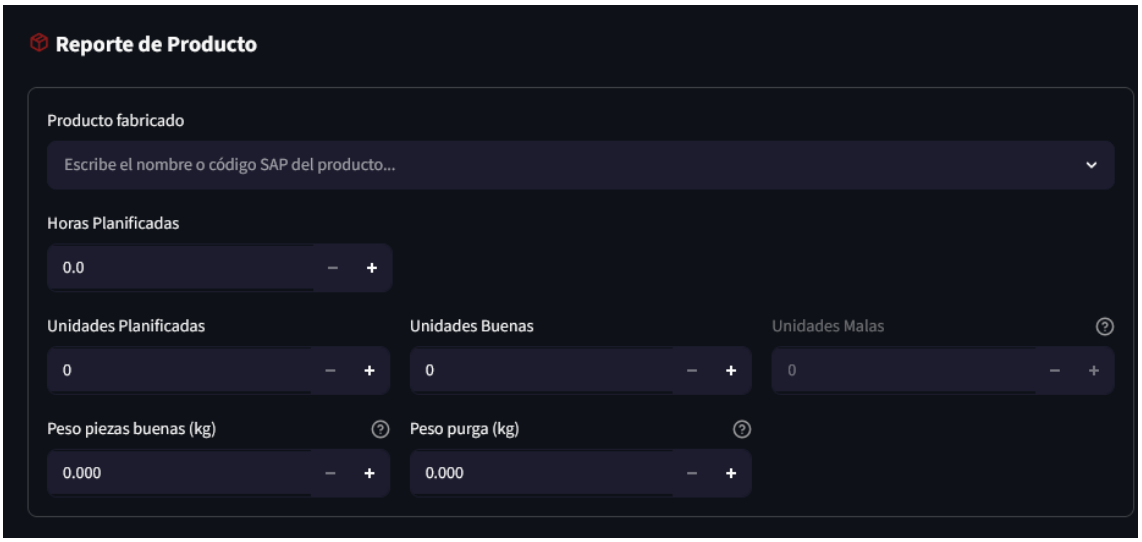
- Fecha: selecciona la fecha del turno. No se permiten fechas futuras.
- Turno: elige entre los turnos disponibles (Mañana, Tarde, Noche).

- Máquina: busca por código o nombre de máquina en el selector.
- Operador responsable: busca por nombre o código del operador.
- Supervisor: selecciona el supervisor a cargo.

4.3 Sección 2-Reporte de Producto

Figura 4

Registro del producto desde el módulo del operador



- Producto fabricado: escribe el nombre o código SAP del producto para buscarlo.
- Horas Planificadas: ingresa las horas de trabajo planificadas para el turno.
- Unidades Planificadas: cantidad de piezas que se esperaban producir.
- Unidades Buenas: cantidad de piezas que pasaron control de calidad.
- Unidades Malas: se calcula automáticamente (Planificado − Buenas). No puedes editarlo.
- Peso piezas buenas (kg): peso total en kilogramos de las piezas buenas.
- Peso purga (kg): material plástico desperdiciado durante el proceso.

Nota.

Si ingresas el peso de piezas buenas y la purga, el sistema calculará automáticamente el rendimiento de materia prima.

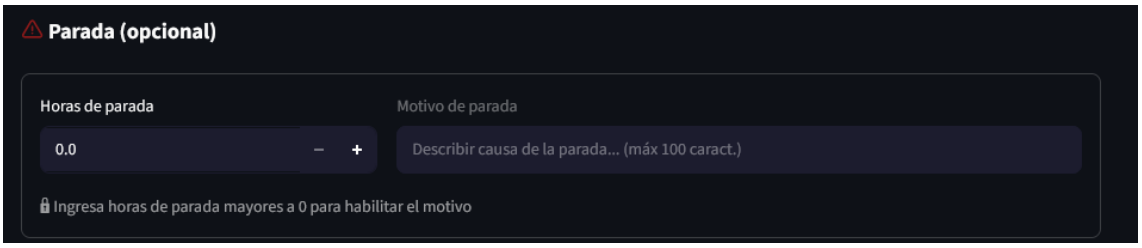
Fórmula: Rendimiento = Peso buenas / (Peso buenas + Purga) × 100

Este indicador aparece en la tarjeta de confirmación antes de guardar.

4.4 Sección 3-Parada (opcional)

Figura 5

Registro datos de parada de la máquina desde el módulo del operador

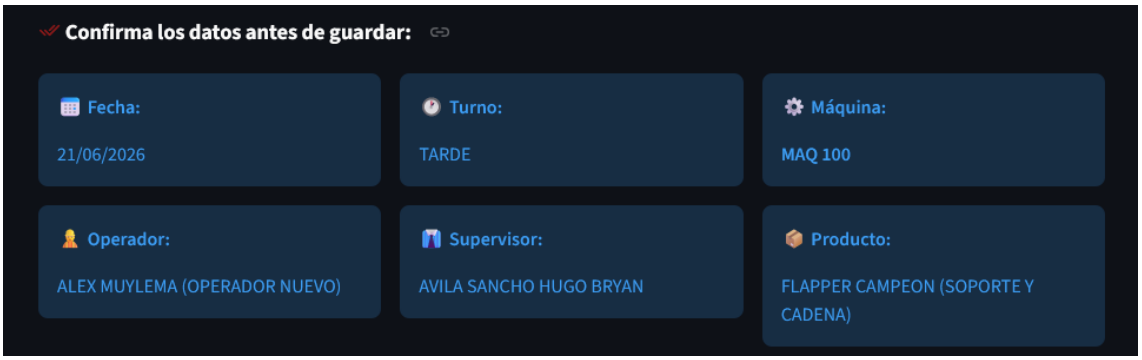


- Horas de parada: si el turno tuvo una parada, ingresa cuántas horas duró (ej. 1.5).
- Motivo de parada: se habilita automáticamente cuando ingresas horas mayores a 0. Describe brevemente la causa (máximo 100 caracteres).
- Si el turno no tuvo paradas, deja las horas en 0.0 y el campo de motivo permanecerá bloqueado.

4.5 Revisión y confirmación

Figura 6

Tarjetas de metadatos del turno



Al hacer clic en "Revisar antes de guardar", el sistema valida todos los campos y muestra una tarjeta de resumen con todos los datos ingresados. Revisar que todo esté correcto antes de continuar.

Validaciones automáticas del sistema:

- Todos los campos obligatorios deben estar completos (Turno, Máquina, Operador, Supervisor, Producto y Planificado > 0).
- Las unidades buenas no pueden superar las planificadas.
- Las horas de parada no pueden superar las horas planificadas.
- El sistema detecta automáticamente si ya existe un registro para esa combinación de fecha, turno, máquina, operador y producto para evitar duplicados.

4.6 Guardar el registro

- Revisa la tarjeta de confirmación. Si algo está mal, haz clic en "Volver a editar".
- Si todo es correcto, haz clic en "Confirmar y guardar".

- Aparecerá un mensaje verde de confirmación: "Registro guardado correctamente."
- El formulario se reinicia automáticamente para el siguiente registro.

Nota.

Una vez guardado, el registro pasa automáticamente por la capa *Bronze* → *Silver* del sistema.

En pocos minutos aparecerá reflejado en los dashboards de gerencia.

No es necesario hacer ninguna acción adicional.

5. Vista Gerencia-Panel de Control

Esta vista está disponible para usuarios con rol Gerente de Planta. Muestra información consolidada de producción con cuatro secciones navegables desde el panel lateral.

Figura 7

Menú lateral módulo gerente

5.1 Navegación

En el sidebar izquierdo encontrarás la sección OPCIONES con cuatro pestañas:

Tabla 5

Opciones del menú vista del gerente

Pestaña	Contenido
Rendimiento	KPIs operativos + métricas de modelos ML + gráficos de producción y eficiencia
Monitoreo	Matriz de cumplimiento de reportes del día por máquina y turno
Histórico	Resumen de producción por año con gráficos comparativos
Simulador IA	Predicción de eficiencia y cumplimiento de plan para un turno futuro

5.2 Filtros

Figura 8

Filtros del submódulo de rendimiento

En la pestaña Rendimiento, el sidebar muestra la sección FILTROS con cuatro selectores:

- Año: filtra todos los datos por año (2020–2026 o Todos).
- Mes: filtra por mes dentro del año seleccionado.
- Máquina: filtra por una máquina específica.
- Turno: filtra por turno (Mañana, Tarde, Noche).

Para limpiar todos los filtros a la vez, haz clic en el botón "Limpiar filtros" al final del sidebar.

5.3 Pestaña Rendimiento

KPI’s Operativos

Muestra seis indicadores clave en la parte superior:

Figura 9

Detalle KPI's rendimiento

Tabla 6

Resumen indicadores KPI's rendimiento

Indicador	Descripción
Turnos registrados	Total de registros en el período seleccionado
Eficiencia promedio	Promedio de eficiencia (%). La meta es 95%
Cumple plan	% de turnos que alcanzaron la cantidad planificada
Unidades buenas	Total de piezas buenas producidas
Unidades malas	Total de piezas que no pasaron calidad
Turnos con parada	% de turnos que tuvieron alguna parada

Rendimiento de Modelos ML

Muestra las métricas de los dos modelos de inteligencia artificial entrenados con datos de la planta:

Figura 10

Detalle métricas modelos ML

- M2-Predicción cumple plan: ROC-AUC del modelo clasificador (meta: 97,7%).
- M2-F1 Score: equilibrio entre precisión y recall (meta: 98,9%).
- M1-Error eficiencia (MAE): error promedio en puntos porcentuales (meta: 9,6 pts).
- M1-R² eficiencia: varianza explicada por el modelo (meta: 0,816).

Se puede expandir la sección "Ver métricas completas por split" para ver el detalle por conjunto de entrenamiento, validación y prueba.

Gráficos de producción

Figura 11

Resumen gráficas de producción

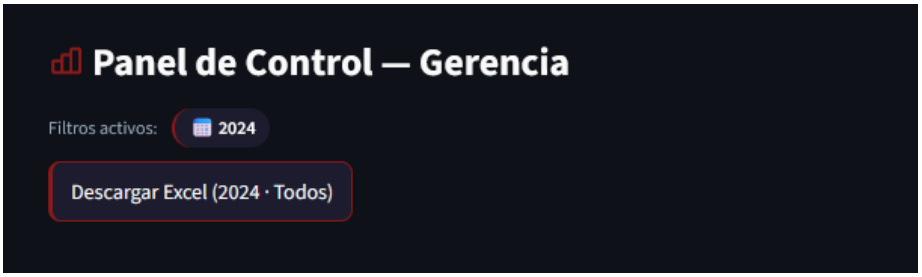
- Volumen de producción por máquina y turno.

- Top 10 productos más fabricados.
- Evolución histórica de volumen mensual.
- Tendencia de eficiencia mensual con línea de meta al 95%.
- Distribución de eficiencia por rangos (<70%, 70-85%, 85-95%, 95-110%, >110%).
- Eficiencia promedio por máquina.
- Producción buenas vs malas por turno.
- Tabla de rendimiento por operador.

Puedes descargar los datos filtrados en Excel haciendo clic en el botón "Descargar Excel".

Figura 12

Descarga de datos



5.4

Pestaña Monitoreo

Muestra una vista del día actual (o la fecha que selecciones en el sidebar) con:

Figura 13

Resumen pantalla monitoreo

- Tres KPIs: reportes esperados, reportes recibidos y porcentaje de cumplimiento.
- Barra de adherencia: indicador visual del avance del día.
- Matriz de reportes: tabla con todas las máquinas en filas y los turnos en columnas. ✅ indica que el turno fue reportado, ❌ que aún no.
- Detalle de registros: si hay datos del día, muestra la tabla con el detalle de cada turno registrado.

Nota.

En el sidebar de la pestaña Monitoreo aparece el selector "Fecha de inspección".

Puedes navegar hacia fechas anteriores para revisar el historial de cumplimiento.

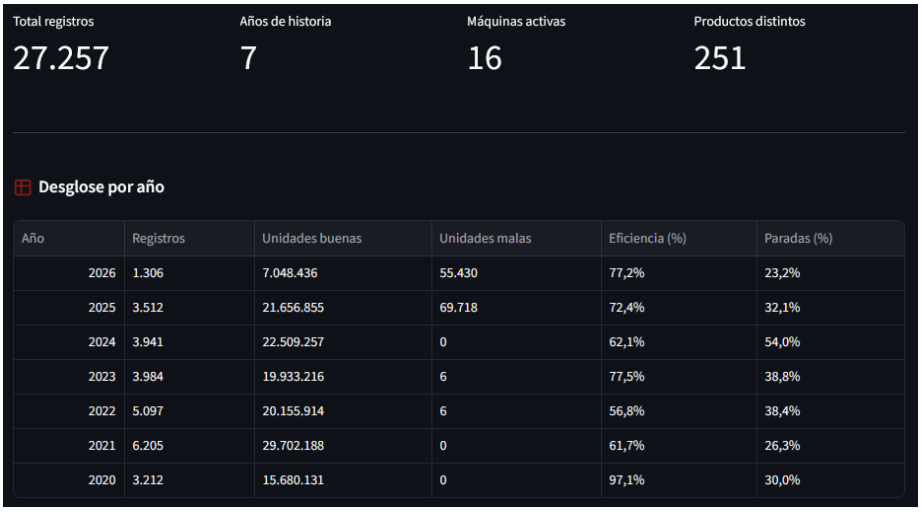
La fecha máxima seleccionable es el día actual.

5.5 Pestaña Histórico

Muestra un resumen consolidado por año con:

Figura 14

Resumen pantalla histórico



Año	Registros	Unidades buenas	Unidades malas	Eficiencia (%)	Paradas (%)
2026	1.306	7.048.436	55.430	77,2%	23,2%
2025	3.512	21.656.855	69.718	72,4%	32,1%
2024	3.941	22.509.257	0	62,1%	54,0%
2023	3.984	19.933.216	6	77,5%	38,8%
2022	5.097	20.155.914	6	56,8%	38,4%
2021	6.205	29.702.188	0	61,7%	26,3%
2020	3.212	15.680.131	0	97,1%	30,0%

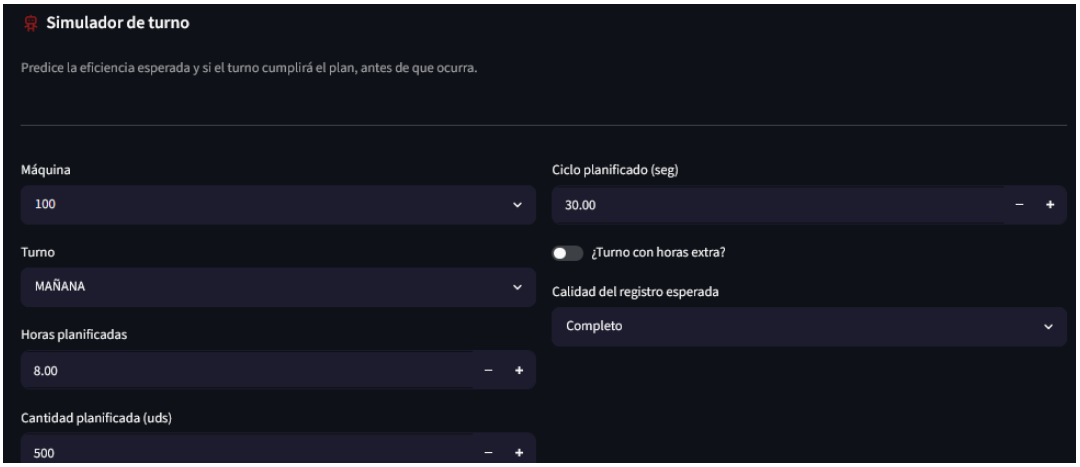
- Cuatro KPIs globales: total de registros, años de historia, máquinas activas y productos distintos.
- Tabla de desglose por año: registros, unidades buenas/malas, eficiencia promedio y tasa de paradas.
- Gráfico de registros por año.
- Gráfico de eficiencia promedio por año con línea de meta al 95%.
- Gráfico comparativo de producción buenas vs malas por año.
- Gráfico de tasa de paradas por año.

5.6 Pestaña Simulador IA

Permite predecir la eficiencia y el cumplimiento de plan de un turno antes de que ocurra, usando los modelos de machine learning entrenados con datos reales de la planta.

Figura 15

Registro datos para el simulador



Cómo usar el Simulador

- Selecciona la máquina para el turno a simular.
- Elige el turno (Mañana, Tarde o Noche).
- Ingresa las horas planificadas (entre 1 y 12).
- Ingresa la cantidad planificada de unidades.
- Ingresa el ciclo planificado en segundos (tiempo por pieza).
- Activa el toggle si el turno tendrá horas extra.
- Selecciona la calidad del registro esperada (Completo, Menor o Crítico).

El sistema consulta automáticamente el historial reciente de la máquina seleccionada para calcular los indicadores de contexto (eficiencia de los últimos 30 turnos, cumplimiento del último turno, etc.).

Resultados del Simulador

Figura 16

Pantalla resumen métricas del simulador

Tabla 7

Resumen métricas simulador

Resultado	Qué indica
Eficiencia estimada	Porcentaje de eficiencia esperada para ese turno
¿Cumplirá el plan?	Sí o NO con la probabilidad asociada
Confianza del modelo	Qué tan seguro está el modelo de su predicción

El velocímetro visual muestra la eficiencia estimada con zonas de color:

- Rojo oscuro (0-70%): eficiencia crítica.

- Rojo (70-85%): eficiencia baja.
- Grafito (85-95%): eficiencia aceptable.
- Grafito oscuro (95-130%): eficiencia óptima (cumple la meta).

Nota. Precisión de los modelos

M2 (clasificador cumple plan): ROC-AUC = 97,7% en datos de prueba 2026.

M1 (regressor eficiencia): R^2 = 0,82 y MAE = 9,57 puntos porcentuales en datos de prueba 2026.

Las predicciones son estimaciones basadas en patrones históricos. No garantizan resultados exactos.

6. Preguntas frecuentes

¿Qué hago si olvidé mi contraseña?

Contacta al administrador del sistema para que restablezca tu contraseña. El sistema no tiene función de recuperación automática por correo electrónico.

¿Por qué los datos tardan en aparecer en el dashboard?

El sistema actualiza los datos en caché cada 5 minutos. Si acabas de guardar un registro y no aparece aún, espera unos minutos y recarga la página.

¿Puedo acceder desde mi celular?

Sí. La aplicación funciona en navegadores móviles (Chrome para Android, Safari para iOS). Para mejor experiencia, usa el dispositivo en modo horizontal o en una tablet.

¿Qué pasa si cierro la pestaña sin guardar?

Los datos del formulario no se guardan automáticamente. Si cierras la pestaña o refrescas antes de hacer clic en "Confirmar y guardar", perderás los datos ingresados y deberás volver a llenar el formulario.

¿Puedo registrar dos veces el mismo turno?

No. El sistema detecta duplicados automáticamente. Si intentas guardar un turno con la misma combinación de fecha, turno, máquina, operador y producto, mostrará un mensaje de error y no procederá el guardado.

El sistema muestra "Error al cargar datos". ¿Qué hago?

- Verifica que tienes conexión a internet activa.
- Recarga la página.
- Si el error persiste, espera 2-3 minutos, puede ser una pausa temporal de la base de datos en la nube.
- Si el problema continúa por más de 10 minutos, contacta al administrador.

7. Glosario

Tabla 8

Glosario de términos

Término	Definición
Turno	Período de trabajo: Mañana, Tarde o Noche.
Eficiencia (%)	Relación entre unidades buenas y unidades planificadas × 100
Cumple plan	Indica si las unidades buenas alcanzaron la meta planificada (Sí/No)
Purga (kg)	Material plástico desperdiciado durante el arranque o cambio de producto
MAE	Error Absoluto Medio-promedio de la diferencia entre predicción y valor real
R^2	Coeficiente de determinación-mide cuánta variación del dato real explica el modelo (1.0 = perfecto)

ROC-AUC	Métrica de clasificación-capacidad del modelo para distinguir entre "cumple" y "no cumple" (1.0 = perfecto)
F1 Score	Media armónica entre precisión y recall del modelo clasificador
Silver layer	Capa de datos limpios y procesados donde se almacenan los registros validados
Bronze layer	Capa de datos crudos, tal como llegan del formulario o los archivos Excel
Streamlit Cloud	Plataforma en la nube donde está alojada la aplicación
Cache	Datos guardados temporalmente para acelerar la carga. Se actualiza cada 5 minutos

8. Soporte técnico

Para reportar errores o solicitar ayuda con el sistema, contacta al administrador de la plataforma.

Tabla 9

Problemas técnicos

Tipo de problema	Acción recomendada
No puedo iniciar sesión	Contactar al administrador para restablecer credenciales
Error al guardar un registro	Tomar captura de pantalla del error y reportarlo al administrador
Datos incorrectos en el dashboard	Verificar el registro en la sección Monitoreo y reportar si hay diferencias
La app no carga	Verificar conexión, recargar página, esperar 5 min y reportar si persiste
Solicitar nuevo usuario	Solicitar al administrador del sistema con nombre completo y rol requerido