Sistema de Prognosis Industrial Adaptativa Basado en Ensemble Híbrido y Aprendizaje Incremental para la Detección Temprana de Fallas

Juan David Valencia Piedrahita

Resumen

En el contexto de la Industria 4.0, la anticipación de fallas en activos dinámicos es un desafío crítico que impacta directamente en la eficiencia operativa y la rentabilidad. Este documento presenta una arquitectura para un sistema de prognosis industrial que es, por diseño, adaptativo, agnóstico al dominio y gobernable. A diferencia de los enfoques tradicionales que dependen de modelos físicos o umbrales estáticos, nuestra solución implementa un pipeline de aprendizaje automático de extremo a extremo que transforma mediciones multivariadas de sensores en probabilidades de falla accionables.

Las contribuciones clave de este trabajo son:

1. Un método de selección de variables multifactorial que puntúa y prioriza mediciones en sus propiedades estadísticas intrínsecas (varianza, estabilidad, tendencia y correlación), permitiendo un enfoque agnóstico y adaptable al activo.

2. Un modelo de línea base de comportamiento normal mediante un ensemble híbrido, que combina la robustez estadística de SARIMAX, la flexibilidad de Prophet para tendencias y estacionalidades complejas, y la sensibilidad de Isolation Forest para la detección de anomalías en los residuos del modelo.

3. Una estrategia de aprendizaje incremental y versionado de modelos, fundamentada en principios de MLOps, que mitiga el concept drift y garantiza la trazabilidad, auditabilidad y evolución del sistema a través de una persistencia transaccional en PostgreSQL.

4. Un motor de inferencia probabilística que integra desviaciones, tendencias y límites adaptativos para estimar la probabilidad de falla por variable, categorizar alertas (‘WARNING‘, ‘CRITICAL‘) y generar reportes auditables que guían la toma de decisiones operativas.

Esta memoria técnica detalla exhaustivamente la arquitectura, los fundamentos teóricos de cada algoritmo, los procesos operativos y la validación del sistema, constituyendo una guía completa para profesionales e investigadores que busquen implementar o escalar soluciones de mantenimiento predictivo robustas en entornos productivos.

Palabras clave: prognosis, series temporales, detección de anomalías, validación temporal, mantenimiento predictivo.

Abstract

In the context of Industry 4.0, prognostics and health management (PHM) for dynamic industrial assets is critical for minimizing unplanned downtime and optimizing operational efficiency. This paper presents a novel architecture for an adaptive industrial prognosis system designed to be domain-agnostic, robust, and evolvable. Unlike traditional approaches that rely on physics-based models or static thresholds, our solution leverages an end-to-end machine learning pipeline to transform raw multivariate sensor data into actionable failure probabilities.

The main contributions of this work are threefold: 1. We introduce a domain-agnostic, multifactorial variable selection method that quantitatively scores and ranks sensors based on their intrinsic statistical properties, including variance, temporal stability, trend, and correlation.

2. The system learns the asset's normal behavior baseline using a hybrid ensemble model that synergistically combines SARIMAX for linear and seasonal patterns, Prophet for complex trends and seasonalities, and an Isolation Forest to monitor model residuals for anomalous behavior.

3. To address the challenge of concept drift, the framework implements an incremental learning strategy and is grounded in MLOps principles, ensuring full traceability, auditability, and governance via a transactional PostgreSQL backend that versions all models, parameters, and performance metrics.

4. Finally, we embed a probabilistic inference engine that fuses deviations, adaptive bounds, and trend signals to estimate failure likelihoods, trigger categorized alerts ('WARNING', 'CRITICAL'), and deliver auditable reports that support maintenance planning.

This technical memory provides a comprehensive description of the system's architecture, the theoretical underpinnings of each algorithm, and its operational validation. The result is a robust, scalable, and reliable framework for the early detection of faults in complex industrial assets, intended for practitioners and researchers aiming to deploy advanced predictive maintenance solutions.

Índice

1 Resumen **7**

2 Planteamiento del problema **8**

2.1 Contexto industrial 8

2.2 Situación inicial 8

2.3 Objetivo general 8

2.4 Objetivos específicos 8

3 Requisitos funcionales y no funcionales **9**

3.1 Requisitos funcionales 9

3.2 Requisitos no funcionales 9

4 Marco teórico **10**

4.1 Sistemas dinámicos y observabilidad 10

4.2 Estadística descriptiva y control de variabilidad 10

4.3 Selección multifactorial de variables 11

4.4 Modelos SARIMAX 13

4.5 Modelo Prophet 14

4.6 Detección de anomalías con Isolation Forest 15

4.7 Fase de Inferencia y Estimación de Probabilidad de Falla 15

4.8 Aprendizaje incremental y control de drift 16

4.9 Persistencia en PostgreSQL y MLOps 17

5 Arquitectura del sistema **18**

5.1 Visión general 18

5.2 Componentes de software 18

6 Descripción detallada por fases **18**

6.1 Fase 1: Ingesta y acondicionamiento 19

6.1.1 Propósito 19

6.1.2 Entradas 19

6.1.3 Procesos 19

6.1.4 Salidas 19

6.1.5 Persistencia y trazabilidad 19

6.1.6 Métricas de control 20

6.2 Fase 2: Selección multifactorial 20

6.2.1 Propósito 20

6.2.2 Entradas 20

6.2.3 Procesos principales 20

6.2.4 Salidas 20

6.2.5 Persistencia 21

6.2.6 Métricas de control 21

6.3 Fase 3: Línea base adaptativa 21

6.3.1 Propósito 21

6.3.2 Entradas 21

6.3.3 Procesos 21

6.3.4 Salidas 22

6.3.5 Controles 23

6.4 Fase 4: Inferencia, probabilidades y alertas 23

6.4.1 Propósito 23

6.4.2 Entradas 23

6.4.3 Procesos 23

6.4.4 Salidas 24

6.4.5 Persistencia 24

7 Flujos de datos **24**

7.1 Flujo general 24

7.2 Diagramas de flujo 25

7.2.1 Pipeline general 25

7.2.2 Selección de variables 27

7.3 Flujo por fase 27

8 Casos de uso industrial **27**

8.1 Energía y distribución eléctrica 27

8.2 Manufactura continua 27

8.3 Petróleo y gas 28

8.4 Transporte ferroviario 28

9 Evaluación y validación **28**

9.1 Métricas cuantitativas 28

9.2 Validación temporal 28

9.3 Evaluación cualitativa 28

10 Gobernanza y auditoría **29**

10.1 Versionado y trazabilidad 29

10.2 Auditorías periódicas 29

10.3 Seguridad y control de acceso 29

11 Lineamientos operativos **29**

11.1 Puesta en marcha 29

11.2 Operación continua 30

12 Plan de evolución **30**

13 Conclusiones **30**

A Tablas principales en PostgreSQL **30**

B Parámetros clave del sistema **31**

C Glosario de clases y métodos **32**

D Buenas prácticas de operación **32**

E Referencias **32**

1 Resumen

la capacidad de transformar datos en inteligencia accionable es un pilar fundamental para la competitividad. Los activos industriales, como motores, transformadores y líneas de producción, son sistemas dinámicos complejos cuyo estado de salud se degrada con el tiempo debido a factores operativos y ambientales. Las fallas no planificadas en estos activos críticos resultan en costos exorbitantes debido a paradas de producción, reparaciones de emergencia y riesgos de seguridad. La prognosis industrial, emerge como una solución estratégica que busca anticipar estas fallas antes de que ocurran.

Este documento presenta una memoria técnica exhaustiva de un sistema de prognosis industrial adaptativo. La solución está diseñada para ser agnóstica al activo y evolutiva, abordando los desafíos clave del mundo real: la heterogeneidad de los datos, la necesidad de criterios cuantitativos para la selección de variables y el fenómeno del concept drift(cambios en el comportamiento del sistema a lo largo del tiempo).

El sistema implementa un pipeline completo que abarca:

- **Fase de ingesta y acondicionamiento**:Un robusto proceso de ETL (Extract, Transform, Load) que limpia, normaliza y estandariza los datos crudos, asegurando su calidad y persistencia en una base de datos transaccional (PostgreSQL).
- **Fase de selección multifactorial de variables**:Un sistema de puntuación que evalúa cada variable de sensor según cuatro ejes: varianza, estabilidad, tendencia y correlación. Esto permite una clasificación cuantitativa y adaptativa de las variables en çríticasz de "monitoreo", enfocando los recursos computacionales donde más se necesitan.
- **Fase de línea base adaptativa**: El corazón del sistema, donde se aprende el çomportamiento normal"de cada variable crítica mediante un ensemble híbrido de modelos estadísticos (SARIMAX) y de machine learning (Prophet). Los residuos de este ensemble son vigilados por un modelo de detección de anomalías (Isolation Forest), creando una línea base de una sensibilidad y robustez excepcionales.
- **Fase de inferencia y alertas**: Un motor de decisión que compara los datos en tiempo real con la línea base adaptativa, calcula una probabilidad de falla integrada y emite alertas categorizadas (‘WARNING‘, ‘CRITICAL‘) con reportes auditables.

La solución es agnóstica al tipo de activo porque se apoya en propiedades estadísticas generales y en algoritmos que no requieren información específica del dominio físico. Los parámetros, pesos y umbrales se justifican teóricamente y se extraen del análisis profundo del notebook operativo ‘Prognosis01 Set2.ipynb‘. Todo el sistema está construido sobre una base de MLOps (Machine Learning Operations), con un fuerte énfasis en el aprendizaje incremental para controlar el drift, y un

versionado y persistencia rigurosos en PostgreSQL para garantizar la trazabilidad, auditabilidad y gobernanza del ciclo de vida del modelo.

2 Planteamiento del problema

2.1 Contexto industrial

Los activos productivos como motores eléctricos, transformadores, compresores, bombas o líneas de proceso presentan dinámicas complejas influenciadas por condiciones ambientales, carga operativa y desgaste. La interrupción inesperada de un solo activo puede generar pérdidas significativas. La prognosis anticipada permite planificar intervenciones, optimizar inventarios y reducir el costo total de propiedad.

2.2 Situación inicial

El análisis de la operación evidenció:

1. Convergencia de datos provenientes de múltiples sensores y sistemas de monitoreo con nomenclaturas heterogéneas.
2. Ausencia de criterios cuantitativos para priorizar las variables que realmente aportan información diagnóstica.
3. Falta de modelos actualizables que reflejen el comportamiento normal de cada activo frente a cambios graduales.
4. Necesidad de auditoría y trazabilidad de los resultados analíticos para soportar procesos regulatorios y de mejora continua.

2.3 Objetivo general

Diseñar e implementar un sistema de prognosis industrial adaptativa que integre procesos de calidad de datos, selección estadística, modelado predictivo híbrido, aprendizaje incremental y gobierno de la información para transformar mediciones multivariadas en probabilidades de falla y alertas confiables.

2.4 Objetivos específicos

1. Estandarizar y documentar las mediciones históricas garantizando que cada transformación sea reproducible.

2. Evaluar cada variable mediante indicadores cuantitativos que permitan clasificarla según su relevancia para la prognosis.
3. Construir líneas base robustas mediante modelos complementarios capaces de capturar patrones lineales, estacionales y anomalías residuales.
4. Implementar estrategias de actualización incremental que mitiguen el *concept drift* sin sacrificar estabilidad.
5. Persistir datos, modelos, métricas y reportes en PostgreSQL utilizando estructuras normalizadas y campos JSONB para asegurar trazabilidad.
6. Emitir probabilidades de falla y alertas categorizadas integradas a reportes de calidad.

3 Requisitos funcionales y no funcionales

3.1 Requisitos funcionales

- Procesar mediciones de múltiples activos con frecuencias de muestreo variables.
- Generar reportes de calidad que documenten valores faltantes, columnas constantes, estadísticas descriptivas y decisiones de imputación.
- Calcular un score multifactorial para cada variable y clasificarla en categorías críticas o de monitoreo.
- Entrenar modelos SARIMAX y Prophet por variable crítica, monitorear residuos con Isolation Forest y versionar los resultados.
- Permitir actualizaciones incrementales en lotes diarios con promedios ponderados y control de versiones.
- Producir probabilidades de falla por variable, estado general del sistema y alertas categorizadas.

3.2 Requisitos no funcionales

- Trazabilidad completa de transformaciones, puntuaciones y modelos mediante persistencia transaccional.
- Modularización del código en clases especializadas (‘DataPreprocessor‘, ‘KeyVariableSelector‘, ‘BaselineModeler‘, ‘IndustrialFailurePredictor‘).
- Configuración agnóstica al activo, con parámetros ajustables a través de estructuras de configuración (‘SystemConfig‘) cuyo propósito conceptual es centralizar todos los parámetros de configuración del sistema.

- Escalabilidad a través de procesamiento paralelo usando ‘ThreadPoolExecutor‘.
- Seguridad en la conexión a PostgreSQL con autenticación y control de versiones.

4 Marco teórico

El marco teórico se organiza en cuatro dominios complementarios: fundamentos de sistemas dinámicos y medición, estadística multivariante aplicada a datos industriales, algoritmos de prognosis y aprendizaje automático, y estrategias de aprendizaje incremental orientadas a MLOps.

4.1 Sistemas dinámicos y observabilidad

Un activo industrial es, fundamentalmente, un sistema físico cuyo comportamiento evoluciona en el tiempo. En la teoría de control, estos se modelan como sistemas dinámicos usando un formalismo de espacio de estados:

Ecuación de Estado: $\dot{x}(t) = f(x(t), u(t))$ Ecuación de Medición:$y(t) = h(x(t)) + v(t)$

El vector $x(t)$ representa el estado interno del sistema (ej., la fatiga acumulada en un metal, la degradación química del aceite de un transformador). Este estado es la verdadera medida de la "salud"del activo, pero es, por naturaleza, inobservable directamente. Lo que sí podemos observar es el vector $y(t)$ de mediciones de sensores (vibración, temperatura, corriente), que son una manifestación indirecta y ruidosa ($v(t)$) del estado interno.

El problema de la prognosis se convierte en un problema de estimación de estado. Dado que no tenemos un modelo físico perfecto (las funciones f y h son desconocidas o demasiado complejas), no podemos usar observadores clásicos como el Filtro de Kalman directamente. En su lugar, este sistema adopta un enfoque basado en datos: construye un modelo estadístico del comportamiento normal de las mediciones $y(t)$. Este modelo actúa como nuestro .ºbservador de estado de salud". Una desviación significativa del comportamiento predicho por nuestro modelo implica un cambio en el estado interno no observable $x(t)$, que interpretamos como una falla incipiente.

4.2 Estadística descriptiva y control de variabilidad

La calidad de los datos para la prognosis se asegura mediante un proceso riguroso de perfilado y estandarización, seguido por el cálculo continuo de métricas que caracterizan el comportamiento de la línea base.

- La estandarización mediante Z-score clásico, expresada como $z = (x - \mu)/\sigma$, es un paso inicial en la Celda 2. Esta estandarización, realizada utilizando StandardScaler de scikit-learn, transforma

las variables numéricas para que tengan una media de 0 y una desviación estándar de 1. Esto es un prerequisito para homogeneizar escalas, lo cual es crucial para los algoritmos de Machine Learning sensibles a la magnitud de las características

- Para el manejo de datos faltantes, los valores nulos en variables numéricas se imputan con la mediana ($\tilde{x}$). La mediana se utiliza específicamente por ser robusta a valores atípicos (outliers), preservando mejor la forma de la distribución que la media

- Media (μ) y Desviación Estándar (σ): Se calculan para la línea base. Durante el procesamiento incremental, estas métricas se actualizan utilizando un promedio ponderado, dando un peso mayor a la historia (ej., 70) y menor al nuevo lote de datos (ej., 30). Este enfoque ponderado es el corazón del componente adaptativo, asegurando que el modelo sea más estable y menos reactivo a fluctuaciones menores en los lotes pequeños

- Varianza (σ^2): Se calcula en la Celda 3 para la selección de variables clave. Las variables con varianza prácticamente nula (ej., $< 1e-6$) son descartadas ya que no son discriminatorias para el modelado de series temporales

La estandarización mediante z-score clásico se expresa como $z = (x - \mu)/\sigma$ y homogeniza escalas, facilitando la comparación entre variables y la aplicación de umbrales estadísticos.

4.3 Selección multifactorial de variables

No todas las mediciones son igualmente informativos. El sistema emplea un método cuantitativo para identificar las variables más relevantes, basado en cuatro pilares. Este proceso es agnóstico porque las métricas son puramente estadísticas, no requieren entender la física detrás de la variable.

1. **Varianza Normalizada (S_var)**

   Fórmula: $S_{var} = \log(1 + |\text{Var}(X)|) \times \frac{\text{Nº de valores únicos}}{\text{Nº total de valores}}$ Intuición: Mide cuánta información o dinamismo contiene una señal. Justificación en Prognosis: Una variable que es constante (Varianza = 0) o casi constante no puede indicar un cambio de estado. Es una señal muerta. Este score favorece a las variables que fluctúan. La transformación logarítmica (log1p) es crucial para manejar varianzas de diferentes órdenes de magnitud sin que las variables de gran escala dominen. La multiplicación por el ratio de valores únicos penaliza a las variables que, aunque no son constantes, tienen una baja cardinalidad (ej. una señal que solo alterna entre 0 y 1), favoreciendo señales continuas más ricas.

2. **Estabilidad temporal (S_estab)**

   Fórmula: $S_{estab} = 1 - \frac{\text{media}(\sigma_{\text{móvil}})}{\sigma_{\text{total}}}$ Intuición: Mide la previsibilidad de la volatilidad de una señal. ¿Es su ruido de fondo constante o errático? Justificación en Prognosis: Este es un concepto

profundo. Para detectar una anomalía, primero debemos definir una normalidad estable. Una variable ideal para prognosis debe tener una volatilidad consistente en condiciones normales (un S_estab alto). Esto permite establecer umbrales de confianza estrechos alrededor de su línea base. Si una variable es intrínsecamente errática (S_estab bajo), requeriría umbrales muy amplios, haciendo imposible la detección de desviaciones sutiles (baja sensibilidad) o generando constantes falsas alarmas (baja especificidad). La conclusión .Este enfoque garantiza equilibrio entre variabilidad y estabilidad"se materializa aquí: al ponderar 'S_var' y 'S_estab' por igual, el sistema busca un "punto dulce": variables que son dinámicas ('S_var' alto) pero no caóticas ('S_estab' alto).

3. **Tendencia estructural (S_trend)**

   Fórmula: $S_{trend} = |\text{pendiente de la regresión lineal sobre la serie}|$ Intuición: Mide si la variable está experimentando un cambio sostenido en una dirección. Justificación en Prognosis:Las tendencias son a menudo el síntoma más temprano de una falla por degradación. El desgaste de un rodamiento no causa una falla instantánea; causa un aumento lento y gradual de la vibración y la temperatura. S_trend está diseñado para capturar precisamente estos fenómenos de deriva a largo plazo.

4. **Correlación media Absoluta (S_corr)**

   Fórmula: $S_{corr} = \frac{1}{N-1} \sum_{i=1, i \neq j}^{N} |\text{Corr}(X_j, X_i)|$ Intuición:Mide qué tan conectada está una variable con el resto del sistema. Justificación en Prognosis:Los activos industriales son sistemas de componentes interconectados. Una falla en un componente a menudo causa efectos en cascada. Una variable que está altamente correlacionada con muchas otras (un S_corr alto) es un buen barómetro de la salud general del sistema. Es más probable que sus desviaciones reflejen un cambio de estado sistémico en lugar de un problema localizado o ruido de sensor.

El score final integra ponderaciones empíricas: $Score = 0{,}3S_{Var} + 0{,}3S_{Estab} + 0{,}2S_{Trend} + 0{,}2S_{Corr}$. Las categorías se calculan en cada lote mediante percentiles dinámicos: variables por encima del percentil 80 se etiquetan como *críticas*, entre los percentiles 50 y 80 como *monitoreo*, y el resto se reserva para auditoría. Este esquema multifactorial materializa el principio operativo 'equilibrio entre variabilidad y estabilidad': las variables seleccionadas son lo suficientemente sensibles para reaccionar a eventos relevantes pero mantienen coherencia estadística que evita alarmas espurias. A medida que ingresan nuevos loteos, los scores se recalculan sobre datos actualizados; por ello, una variable puede migrar de monitoreo a crítica si su varianza informativa aumenta o si su tendencia se acentúa, alineando el sistema con el carácter agnóstico y adaptativo requerido en entornos multiactivo.

Desde la teoría de la varianza, la información aportada por una medición se relaciona con la magnitud de su dispersión respecto a la media: una varianza casi nula implica ausencia de señal útil, mientras que variancias excesivas sin estabilidad indican ruido blanco. La estabilidad temporal se fundamenta en la comparación del proceso con un proceso de segundo orden estable; si la

desviación estándar local es estacionaria, la serie satisface condiciones de ergodicidad que permiten proyectar comportamiento futuro. Para la tendencia se recurre a pruebas de hipótesis sobre la pendiente (estadístico t), asegurando que sólo tendencias estadísticamente significativas eleven el score. Finalmente, la correlación media se apoya en la teoría de grafos de correlación: variables con alta centralidad (alto S_{Corr}) actúan como nodos centinela y permiten detectar fallas sistémicas.

La metodología es agnóstica porque ninguna métrica depende de la física específica del activo; se emplean únicamente propiedades estadísticas universales. Cada reevaluación se ejecuta lote a lote siguiendo el pseudocódigo:

1. Extraer ventana temporal alineada con el lote actual y recalcular $S_{Var}, S_{Estab}, S_{Trend}, S_{Corr}$ utilizando parámetros de ventana parametrizables.
2. Comparar scores con percentiles históricos almacenados en ‘variable_analysis_history‘ para detectar saltos estructurales.
3. Actualizar ‘selected_variables_history‘ registrando cambios de categoría y motivos (ej. “aumento de tendencia”), habilitando auditorías.

Este ciclo asegura que la clasificación de variables se mantenga vigente frente a cambios en condiciones operativas, evitando dependencia de expertos humanos y garantizando reproducibilidad.

4.4 Modelos SARIMAX

SARIMAX (Seasonal AutoRegressive Integrated Moving Average with eXogenous regressors) extiende ARIMA incorporando estacionalidad y regresores externos. Su ecuación general es:

$$\Phi_p(B)\Phi_P(B^s)(1-B)^d(1-B^s)^D y_t = \Theta_q(B)\Theta_Q(B^s)\varepsilon_t + \mathbf{x}_t^\top \boldsymbol{\beta},$$

donde B es el operador de retardo, s el periodo estacional y $\mathbf{x}_t$ el vector de regresores. En la implementación actual, la clase ‘SARIMAX‘ se emplea sin regresores exógenos, por lo que el modelo opera efectivamente como un SARIMA con órdenes fijos $(1,1,1)(1,1,1,s)$ definidos en la configuración para agilizar el procesamiento incremental. La selección automática de órdenes mediante inspección ACF/PACF y protocolo Box–Jenkins se reconoce como una mejora pendiente documentada en los entregables técnicos.

Prueba ADF. La prueba de Dickey–Fuller aumentada contrasta la hipótesis nula de raíz unitaria. Se aplica rescribiendo la serie en diferencias y ajustando una regresión con rezagos adecuados: si el estadístico τ es menor que el valor crítico, se concluye estacionariedad. En la implementación se elige el número de rezagos mediante información de Akaike, asegurando residuos no autocorrelacionados.

Prueba KPSS. La prueba de Kwiatkowski–Phillips–Schmidt–Shin contrasta la hipótesis nula de estacionariedad frente a una alternativa de raíz unitaria. Se calcula sobre residuos de una regresión contra constante o tendencia. Un estadístico elevado indica necesidad de diferenciación. El uso combinado de ADF y KPSS evita decisiones unilaterales: se diferencia sólo cuando ADF no rechaza raíz unitaria y KPSS rechaza estacionariedad.

Pertinencia. SARIMAX es ideal cuando las variables exhiben memoria lineal y estacionalidad explícita (ej. ciclos diarios). Además permite incluir regresores exógenos, como consignas de control, que explican parte de la varianza, reduciendo residuos y mejorando la vigilancia posterior. Su naturaleza parametrizada facilita la persistencia: vector de coeficientes, matrices de covarianza y órdenes se almacenan en JSONB, habilitando comparaciones versionadas.

4.5 Modelo Prophet

Prophet formaliza la serie como $y(t) = g(t) + s(t) + h(t) + \varepsilon_t$, combinando tendencia $g(t)$ piecewise lineal o logística, estacionalidades periódicas $s(t)$ modeladas con series de Fourier y efectos de vacaciones/eventos $h(t)$. La detección automática de *changepoints* controla la flexibilidad: un prior sobre la tasa de cambio impone suavidad y evita sobreajuste. Prophet es pertinente cuando las series exhiben múltiples estacionalidades (día, semana) y patrones no lineales; su inferencia bayesiana permite incorporar incertidumbre y ajustar la sensibilidad con pocos hiperparámetros, característica crítica para operación agnóstica.

Descomposición aditiva. El modelo asume componentes independientes sumables. El término de tendencia se parametriza con puntos de cambio $\{\tau_k\}$; los coeficientes δ_k regulan la magnitud de los quiebres. Las estacionalidades se codifican mediante pares seno/coseno de frecuencias conocidas, lo que facilita representar repeticiones complejas sin necesidad de suponer periodicidad estricta.

Regularización y pertinencia. En teoría, el hiperparámetro ‘changepoint_prior_scale‘ controla la respuesta ante cambios abruptos: valores bajos priorizan estabilidad (útil en variables con ruido bajo), mientras que valores altos reaccionan rápidamente a nuevas tendencias (adecuado para sensores susceptibles a recalibración). Sin embargo, la implementación actual utiliza valores fijos por defecto para los hiperparámetros principales (incluyendo ‘changepoint_prior_scale‘ y ‘seasonality_prior_scale‘) como punto de partida estable; la calibración automática según la estabilidad de cada serie se mantiene como una optimización planificada. Prophet es apropiado en el ensemble porque captura no linealidades y efectos de calendario que un SARIMA puramente lineal no modela bien.

Persistencia. Los parámetros de Prophet (coeficientes de tendencia, amplitudes de series de Fourier y contribuciones de eventos) se serializan y almacenan en ‘baseline_models‘. La estructura aditiva facilita recalcular predicciones históricas, requisito para auditorías y reconstrucción de alertas.

4.6 Detección de anomalías con Isolation Forest

Isolation Forest aísla observaciones mediante particiones aleatorias en árboles binarios. El score se aproxima por $s(x, n) = 2^{-E[h(x)]/c(n)}$, donde $E[h(x)]$ es la profundidad promedio para aislar x y $c(n)$ la profundidad promedio de un árbol aleatorio. El algoritmo no requiere supuestos distribucionales y detecta anomalías como puntos que se aíslan en pocas particiones. Al aplicarlo directamente sobre la serie temporal estandarizada, Isolation Forest identifica anomalías en el comportamiento general de la variable. La aplicación sobre residuos residuales (las desviaciones no explicadas por SARIMA o Prophet) es un objetivo de mejora que reduciría la tasa de falsos positivos. El parámetro ‘contamination‘ se calibra con el ratio histórico de eventos confirmados, permitiendo que el modelo se adapte por activo sin depender de etiquetas exhaustivas.

Vigilancia de anomalías en la serie temporal. La hipótesis clave es que, bajo operación normal, los residuos del ensemble se distribuyen alrededor de cero con varianza acotada. Isolation Forest identifica trayectorias residuales que rompen esta hipótesis, señalando fallas mecánicas no previstas por los modelos de pronóstico. El umbral de decisión se fija mediante el percentil de los scores históricos, documentado en ‘baseline_metrics‘ para auditoría.

Persistencia. En la implementación actual, cada actualización incremental inserta un registro en ‘baseline_incremental_control‘ con el último ‘timestamp‘ procesado, el número y tamaño del lote, garantizando trazabilidad operativa básica. Una oportunidad de mejora, ya identificada para futuras iteraciones, consiste en extender la columna JSONB ‘learning_metrics‘ para almacenar métricas de drift (por ejemplo, divergencia de Jensen–Shannon) y hashes de los modelos; esto permitiría auditar la deriva estadística y verificar la integridad de cada versión de forma más rigurosa.

4.7 Fase de Inferencia y Estimación de Probabilidad de Falla

Esta fase es donde el sistema pasa de observar a diagnosticar. No se limita a decir "hay una desviación", sino que cuantifica la probabilidad de que esa desviación represente una falla.

Cálculo de Desviaciones Normalizadas :Para un nuevo punto de datos y_t, el sistema obtiene la predicción de la línea base $\hat{y}_t$ y la desviación estándar histórica de los residuos, σ_{res}. La desviación

normalizada es $z_t = (y_t - \hat{y}_t)/\sigma_{res}$. Este valor 'z' indica cuántas desviaciones estándar se aleja el punto actual de lo que se considera "normal".

Integración de factores para la probabilidad de falla. La estimación de riesgo no se apoya en observaciones aisladas, sino en el comportamiento reciente de cada variable. Para cada ventana de análisis se sintetizan tres componentes: (i) la *magnitud de la desviación*, entendida como el promedio de las desviaciones normalizadas respecto a la línea base; (ii) la *frecuencia de la desviación*, medida como la proporción de puntos que exceden los límites adaptativos; y (iii) la *tendencia de la desviación*, cuantificada mediante la pendiente de una regresión lineal sobre la serie de desviaciones. Estos términos se combinan a través de una función de puntuación ponderada y el resultado se escala al intervalo $[0, 1]$ para obtener la Probabilidad de Falla (P(Falla)). En términos formales, se trata de un score de riesgo que captura qué tan plausible es una falla inminente; no corresponde a una probabilidad frecuentista estricta, pero ofrece una métrica consistente para priorizar alertas.

Categorización de Alertas y Reportes Auditables : Se aplican umbrales a este score de riesgo para generar alertas accionables: 'NORMAL' (P(Falla) <0.7):** El sistema opera dentro de los parámetros esperados. 'WARNING' (0.7 <= P(Falla) <0.9):** Indica una desviación notable que requiere monitoreo. Es una alerta temprana. 'CRITICAL' (P(Falla) >= 0.9):** Indica una desviación severa y persistente, sugiriendo una alta probabilidad de falla inminente. Requiere acción inmediata.

Cada alerta generada se registra en la base de datos con su timestamp, nivel, la variable que la causó y la probabilidad calculada, creando un reporte auditable esencial para el análisis post-mortem y la mejora continua.

4.8 Aprendizaje incremental y control de drift

El *concept drift* describe cambios en la distribución $P(X, y)$ con el tiempo. Para capturarlo sin perder memoria histórica se implementa aprendizaje incremental basado en mini-lotes condicionado por métricas de estabilidad:

- **Ingesta incremental**: cada lote diario de hasta 1000 registros (Lote configurable) actualiza estadísticas suficientes (media, varianza, autocovarianzas) sin reentrenar desde cero.
- **Actualización exponencial**: Las métricas estadísticas clave (media, desviación estándar, asimetría, curtosis) se ajustan con promedios ponderados $\theta_t = (1\alpha)\theta_{t1} + \alpha\theta_{\text{lote}}$, con $\alpha = 0,3$. Esta regla preserva el comportamiento histórico (70) y captura cambios persistentes (30). La fusión de patrones de series temporales (estacionariedad, estacionalidad) utiliza una lógica de selección de evidencia (ej., mínimo p-value) en lugar de la ponderación exponencial.

- **Control de versiones**: se conservan hasta cinco modelos activos en ‘baseline_model_versions‘, etiquetando cada versión con metadatos de lote, hiperparámetros y desempeño. Esta política facilita rollback inmediato ante detección de degradación.

El enfoque incremental mantiene agnosticismo respecto al activo: los mismos hiperparámetros se aplican a cualquier medición porque las actualizaciones dependen únicamente de estadísticas empíricas, no del dominio físico.

Persistencia. En la implementación actual, cada actualización incremental inserta un registro en ‘baseline_incremental_control‘ con el último ‘timestamp‘ procesado, el número y tamaño del lote, garantizando trazabilidad operativa básica. Una oportunidad de mejora, ya identificada para futuras iteraciones, consiste en extender la columna JSONB ‘learning_metrics‘ para almacenar métricas de drift (por ejemplo, divergencia de Jensen–Shannon) y hashes de los modelos; esto permitiría auditar la deriva estadística y verificar la integridad de cada versión de forma más rigurosa.

4.9 Persistencia en PostgreSQL y MLOps

La arquitectura de persistencia utiliza PostgreSQL con SQLAlchemy. Las tablas relevantes son:

- ‘normalized_data_table‘: datos estandarizados con índice temporal.
- ‘variable_mapping‘: mapeo entre nombres originales y normalizados.
- ‘selected_variables_history‘ y ‘variable_analysis_history‘: histórico de scores y categorizaciones.
- ‘baseline_models‘, ‘baseline_model_versions‘, ‘baseline_metrics‘, ‘baseline_validation‘, ‘baseline_incremental_control‘, ‘baseline_quality_reports‘: repositorio integral de modelos, métricas y auditoría.

Los campos JSONB almacenan parámetros del modelo, intervalos de confianza, métricas y reportes. Esta estrategia facilita la trazabilidad y respalda las prácticas de MLOps.

Un punto de mejora fundamental en la arquitectura actual es que los objetos de modelo entrenados (SARIMA, Prophet e Isolation Forest) no se serializan directamente dentro de la base de datos; únicamente se persisten sus parámetros y métricas en formato JSONB. Esto significa que para ejecutar inferencia es necesario reconstruir los modelos a partir de dichos parámetros o disponer de una estrategia externa de serialización (por ejemplo, persistencia mediante ‘pickle‘ en columnas ‘BYTEA‘). Incorporar este mecanismo de serialización directa reduciría tiempos de despliegue y evitaría inconsistencias entre versiones.

5 Arquitectura del sistema

5.1 Visión general

La arquitectura se divide en capas:

1. **Adquisición y acondicionamiento**: lectura de archivos, ETL y normalización ejecutada por ‘DataPreprocessor‘.
2. **Selección estadística**: cálculo de scores y categorización implementados en ‘KeyVariableSelector‘.
3. **Línea base adaptativa**: entrenamiento, validación y versionado coordinados por ‘BaselineModeler‘.
4. **Inferencia**: cálculo de probabilidades, alertas y visualizaciones a cargo de ‘IndustrialFailurePredictor‘.
5. **Gobierno de datos**: persistencia y auditoría gestionadas a través de PostgreSQL y SQLAlchemy.

5.2 Componentes de software

- `DataPreprocessor.load_data`, `clean_data`, `normalize_data`: gestionan carga, limpieza, imputación (medianas) y estandarización (`StandardScaler`).
- `KeyVariableSelector.analyze_variables`: ejecuta el scoring multifactorial, registra métricas y clasifica variables.
- `BaselineModeler.process_incremental`: administra lotes, entrena modelos SARIMAX y Prophet, aplica Isolation Forest a residuos, calcula métricas (RMSE, MAE, AIC, BIC), almacena patrones y controla versiones.
- `IndustrialFailurePredictor.predict_failures`: calcula desviaciones, tendencias, probabilidades y genera alertas.

6 Descripción detallada por fases

Cada fase se describe con estructura uniforme: propósito, entradas, procesos, salidas, persistencia, métricas y controles.

6.1 Fase 1: Ingesta y acondicionamiento

6.1.1 Propósito

Transformar datos brutos en conjuntos estandarizados, trazables y auditables.

6.1.2 Entradas

- Archivos CSV/Excel provenientes de sistemas SCADA o historiadores.
- Tablas operativas consultadas mediante ‘DatabaseSetup.create_database_if_not_exists‘ y conexiones ETL.
- Metadatos de unidades y nomenclaturas.

6.1.3 Procesos

1. Normalización de nombres con eliminación de acentos (función ‘_normalize_column_name‘).
2. Perfilamiento estadístico: medias, varianzas, asimetrías, curtosis, conteo de nulos, detección de columnas irrelevantes.
3. Imputación robusta de valores faltantes mediante medianas o categorías explícitas.
4. Estandarización con ‘StandardScaler‘, almacenando parámetros para reproducibilidad.
5. Generación de reportes (‘quality_report‘) con secciones de advertencias y mapeo de variables.
6. Inserción en PostgreSQL (‘normalized_data_table‘, ‘variable_mapping‘).

6.1.4 Salidas

- ‘cleaned_data‘ y ‘normalized_data‘ como marcos de datos listos para análisis.
- Reportes de calidad impresos en consola y almacenados en JSON.
- Tablas actualizadas en PostgreSQL.

6.1.5 Persistencia y trazabilidad

- ‘normalized_data_table‘: incluye columna ‘timestamp‘ obligatoria.
- ‘variable_mapping‘: conserva correspondencia entre nombres normalizados y originales.
- Reportes de calidad en la tabla ‘baseline_quality_reports‘ durante fases posteriores.

6.1.6 Métricas de control

- Porcentaje de valores faltantes por variable.
- Cantidad de columnas constantes o eliminadas.
- Rango de valores post-estandarización.

6.2 Fase 2: Selección multifactorial

6.2.1 Propósito

Identificar variables con mayor valor diagnóstico y mantener histórico de cambios.

6.2.2 Entradas

- ‘normalized_data_table‘ con índice temporal.
- Mapeo de nombres (‘variable_mapping‘).

6.2.3 Procesos principales

1. Validación de conexión a la base de datos (‘_validate_db_connection‘).
2. Creación de tablas de histórico (‘selected_variables_history‘, ‘variable_analysis_history‘).
3. Cálculo de scores individuales (‘_analyze_single_variable‘) con validaciones de valores únicos > 3, ratio de valores únicos $> 0{,}001$ y varianza mínima.
4. Registro de variables descartadas con razones explícitas (constantes, baja diversidad).
5. Clasificación en categorías usando percentiles 80 y 50.
6. Inserción de resultados y scores en PostgreSQL.

6.2.4 Salidas

- Diccionario ‘selected_variables‘ con categorías y scores.
- Registros en ‘selected_variables_history‘ y ‘variable_analysis_history‘.
- Métricas de categorización (‘categorization_metrics‘).

6.2.5 Persistencia

- Cada variable se almacena con nombre normalizado, nombre original, score, categoría y marca temporal.
- Análisis detallado guardado en JSONB (varianza, estabilidad, tendencia, correlación, score final).

6.2.6 Métricas de control

- Número total de variables analizadas y descartadas.
- Conteo de variables críticas y de monitoreo.
- Listado de variables excluidas con motivo (documentado en ‘removed_variables‘).

6.3 Fase 3: Línea base adaptativa

6.3.1 Propósito

Aprender el comportamiento normal de las variables críticas, actualizarlo con nueva evidencia y registrar métricas de desempeño.

6.3.2 Entradas

- Variables críticas identificadas en la fase anterior.
- Datos históricos y lotes incrementales provenientes de ‘normalized_data_table‘.
- Configuración del sistema (‘SystemConfig‘): ventana de entrenamiento (90 días), frecuencia de actualización (1 día), tamaño mínimo de datos (1000 registros), niveles de confianza (95 %), tamaño de lote (1000), máximo de versiones (5).

6.3.3 Procesos

1. Creación de tablas (‘baseline_models‘, ‘pattern_registry‘, ‘baseline_metrics‘, ‘baseline_validation‘, ‘baseline_incremental_control‘, ‘baseline_model_versions‘).
2. Obtención de estadísticas globales (‘get_data_statistics‘) para verificar cobertura.
3. Selección de variables críticas; fallback a variables numéricas si no hay crítica.
4. Procesamiento paralelo (‘ThreadPoolExecutor‘) con máximo 3 hilos para escalar.

5. Para cada variable crítica se ejecuta un extbfpipeline teórico-operativo:

   (a) Evaluación de estacionariedad con pruebas ADF y KPSS; decisión de diferenciación y determinación de orden de integración.

   (b) Análisis espectral y descomposición STL para identificar estacionalidades múltiples. Las funciones ACF/PACF guían la selección de órdenes autorregresivos y de medias móviles.

   (c) Entrenamiento de SARIMAX calibrado con análisis de residuos y diagnóstico asociado.

   (d) Entrenamiento de Prophet con la configuración base establecida (hiperparámetros por defecto) como referencia robusta para todas las variables.

   (e) Generación de límites adaptativos (bandas de confianza) por modelo. La contribución de cada componente al diagnóstico se materializa en la Fase 4, donde se contrastan los datos recientes contra estos límites, en lugar de promediar pronósticos con ponderaciones por RMSE.

   (f) Vigilancia de anomalías en la serie temporal y aplicación de Isolation Forest (‘contamination‘ calibrado por variable) para capturar anomalías residuales. Se calculan estadísticos de conformidad (media, varianza, proporción de anomalías).

   (g) Cálculo de métricas de desempeño (RMSE, MAE, AIC, BIC) documentadas en ‘baseline_-metrics‘.

   (h) Registro de patrones específicos (estacionalidad dominante, rupturas de tendencia, correlaciones cruzadas relevantes) en ‘pattern_registry‘ como anotaciones auditables.

   (i) Control de versión: almacenamiento de parámetros, métricas y metadata de entrenamiento; activación inmediata tras el procesamiento incremental exitoso.

   (j) Actualización incremental: combinación 70/30 de parámetros anteriores y nuevos; registro del lote procesado en ‘baseline_incremental_control‘ para mantener la continuidad operativa.

6.3.4 Salidas

- Modelos activos en ‘baseline_models‘ con parámetros (JSONB), estadísticas base, intervalos de confianza y bandera ‘is_active‘.
- Versiones históricas en ‘baseline_model_versions‘ con indicadores de rendimiento.
- Patrones registrados (tendencias, estacionalidades, rupturas) en ‘pattern_registry‘.
- Métricas y resultados de validación temporal en ‘baseline_metrics‘ y ‘baseline_validation‘.
- Reportes de calidad en ‘baseline_quality_reports‘.

6.3.5 Controles

- Monitoreo de ratios de anomalías y estabilidad de residuos (evaluación de la normalidad) para detectar degradación del modelo.
- Límite de versiones activas (5) para evitar crecimiento indefinido y garantizar trazabilidad manejable.

6.4 Fase 4: Inferencia, probabilidades y alertas

6.4.1 Propósito

Comparar el comportamiento reciente con la línea base, calcular probabilidades de falla y emitir alertas accionables.

6.4.2 Entradas

- Modelos versionados con límites adaptativos (baseline, límites superior e inferior).
- Datos recientes (últimas 24 horas) extraídos de ‘normalized_data_table‘.
- Umbrales de decisión (‘warning = 0.7‘, ‘critical = 0.9‘, ‘min_deviation = 0.1‘).

6.4.3 Procesos

1. Cálculo de desviaciones normalizadas respecto a la línea base (‘_calculate_deviations‘) utilizando residualización doble (predicción ensemble y límites adaptativos).
2. Análisis de tendencias mediante regresión lineal para cuantificar la pendiente de las desviaciones. La confirmación de monotonicidad mediante pruebas como el test de Mann–Kendall se mantiene como mejora futura.
3. Integración de factores para probabilidades: desviación media (40 %), proporción fuera de límites (40 %), fuerza de la tendencia (20 %), ajustados por la correlación media de la variable para ponderar sensores sistémicos.
4. Estimación bayesiana de probabilidad de falla combinando el score con una prior histórica derivada de eventos confirmados.
5. Generación de alertas ‘WARNING‘ y ‘CRITICAL‘ según umbrales; clasificación auditada con descripciones generadas automáticamente.

6. Cálculo del estado general mediante promedio ponderado por criticidad y correlación para reflejar el aporte de cada sensor al riesgo global.

7. Generación de visualizaciones (barras, gauge, evolución vs. línea base) y reportes textuales con trazabilidad a la versión del modelo que originó la alerta.

6.4.4 Salidas

- Diccionario de probabilidades por variable.
- Lista de alertas con timestamp, nivel, mensaje y probabilidad.
- Estado del sistema con probabilidad agregada.
- Reporte textual (‘generate_report‘) con resumen ejecutivo.

6.4.5 Persistencia

- Las métricas de inferencia y probabilidades se almacenan en ‘baseline_quality_reports‘ con referencias a ‘baseline_model_versions‘, garantizando auditoría cruzada.
- Las alertas categorizadas se persisten en colecciones JSONB con campos de justificación (desviación máxima, tendencia, correlación asociada) para soportar análisis post-mortem.

7 Flujos de datos

7.1 Flujo general

Etapa	**Descripción del flujo**
Ingesta y acondicionamiento	Captura de datos → normalización de nombres → perfilamiento → imputación → estandarización → reporte de calidad → carga a PostgreSQL.
Selección multifactorial	Recuperación de datos limpios → cálculo de scores → clasificación → almacenamiento de resultados y métricas.
Línea base adaptativa	Obtención de variables críticas → entrenamiento y validación → cálculo de métricas → registro de versiones → control incremental.
Inferencia y alertas	Extracción de datos recientes → cálculo de desviaciones y tendencias → integración probabilística → emisión de alertas → reporte final.

7.2 Diagramas de flujo

Para complementar la comprensión operativa, se incluyen diagramas de flujo que reflejan el pipeline completo y el detalle de la fase de selección de variables.

7.2.1 Pipeline general

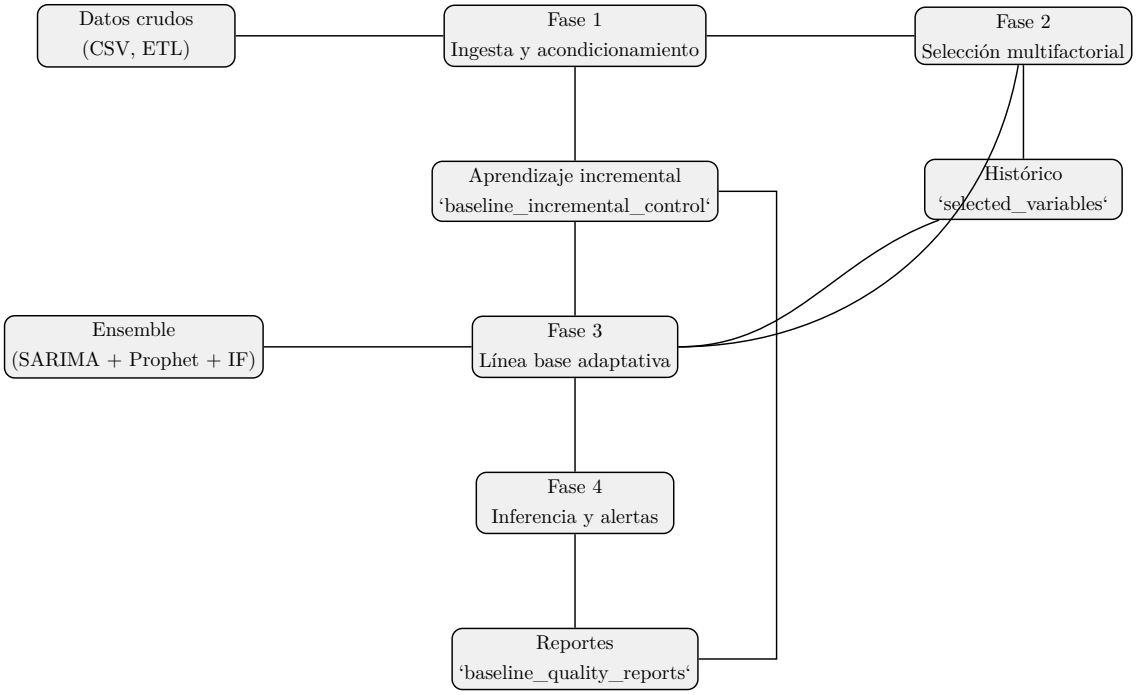

Figura 1: Flujo general del sistema de prognosis adaptativa.

7.2.2 Selección de variables

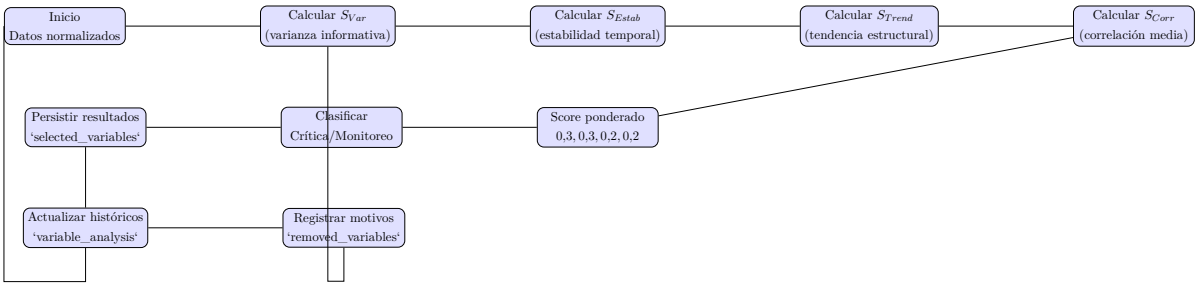

Figura 2: Flujo detallado de la selección multifactorial de variables.

7.3 Flujo por fase

1. **Ingesta**: datos brutos → 'DataPreprocessor' → 'normalized_data_table' + 'variable_mapping' + reporte de calidad.
2. **Selección**: 'normalized_data_table' → 'KeyVariableSelector' → 'selected_variables_history' + 'variable_analysis_history'.
3. **Línea base**: variables críticas → 'BaselineModeler' → 'baseline_models' + 'baseline_model_versions' + 'pattern_registry' + métricas asociadas.
4. **Inferencia**: datos recientes + límites adaptativos → 'IndustrialFailurePredictor' → probabilidades, alertas, estado del sistema, reportes.

8 Casos de uso industrial

8.1 Energía y distribución eléctrica

Los transformadores de potencia y subestaciones presentan estacionalidades ligadas a la demanda. El ensemble permite diferenciar entre picos estacionales esperados y desviaciones anómalas provocadas por sobrecargas.

8.2 Manufactura continua

En líneas de producción con motores y bombas, la detección temprana de vibraciones y temperaturas fuera de rango permite programar mantenimientos antes de fallas catastróficas.

8.3 Petróleo y gas

Las plantas de compresión requieren monitoreo constante de presión y caudal. El sistema identifica patrones de desgaste y fugas incipientes, incorporando mediciones exógenas (temperatura ambiente, composición del fluido).

8.4 Transporte ferroviario

Variables como intensidad de corriente de tracción y temperatura de ejes muestran variaciones condicionadas por horarios y pendientes. La combinación de Prophet y SARIMAX captura estacionalidades diarias y semanales, mientras que Isolation Forest alerta sobre comportamientos anómalos.

9 Evaluación y validación

9.1 Métricas cuantitativas

- **RMSE y MAE**: cuantifican la precisión de las predicciones respecto a los datos históricos.
- **AIC y BIC**: guían la selección de órdenes SARIMAX y la comparación entre modelos.
- **Ratio de anomalías**: porcentaje de residuos señalados por Isolation Forest, indicador de estabilidad del modelo.
- **Probabilidades promedio**: monitoreo del estado global del sistema.

9.2 Validación temporal

‘BaselineModeler‘ utiliza particiones temporales (80 % entrenamiento, 20 % validación) para evaluar el desempeño en ventanas recientes. Se resguardan resultados en ‘baseline_validation‘ para auditorías.

9.3 Evaluación cualitativa

Los reportes de calidad y las métricas almacenadas permiten a analistas revisar la congruencia de alertas, identificar variables sensibles y ajustar umbrales operativos.

10 Gobernanza y auditoría

10.1 Versionado y trazabilidad

Cada modelo dispone de identificador, versión, parámetros, métricas y metadata. Los analistas pueden reconstruir estados históricos recuperando registros de ‘baseline_model_versions‘ y ‘baseline_quality_reports‘.

10.2 Auditorías periódicas

Se recomienda un ciclo mensual de revisión que incluya:

1. Revisión de variables críticas y movimientos recientes en ‘selected_variables_history‘.
2. Análisis de patrones detectados y su impacto en decisiones.
3. Evaluación del drift mediante métricas de cobertura y ratio de anomalías.
4. Validación de alertas ‘CRITICAL‘ con condiciones reales del activo.

10.3 Seguridad y control de acceso

Las credenciales de PostgreSQL se gestionan a través de ‘DB_CONFIG‘. Se recomienda implementar roles con privilegios diferenciados para lectura, escritura y administración.

11 Lineamientos operativos

11.1 Puesta en marcha

1. Crear la base de datos ‘prognosis_db‘ mediante ‘DatabaseSetup.create_database_if_not_exists‘.
2. Ejecutar el módulo de ingesta para cargar datos históricos y generar el primer reporte de calidad.
3. Correr la selección multifactorial para identificar variables críticas.
4. Entrenar la línea base con historia suficiente (mínimo 1000 registros por variable crítica).
5. Validar resultados iniciales y ajustar umbrales de alertas según políticas del negocio.

11.2 Operación continua

- Automatizar ingesta diaria y ejecución del procesamiento incremental.
- Monitorear logs para identificar variables descartadas o fallas de conexión.
- Revisar reportes de probabilidades y alertas para coordinar acciones de mantenimiento.
- Respaldar bases de datos y metadatos enrutando copias de seguridad programadas.

12 Plan de evolución

- Integrar modelos adicionales (por ejemplo, LSTM) en el ensemble cuando la dinámica lo requiera.
- Incorporar diagnósticos explicables (‘SHAP‘, ‘LIME‘) sobre residuos de Isolation Forest.
- Extender la persistencia para capturar acciones tomadas tras cada alerta y medir efectividad.
- Desplegar dashboards interactivos alimentados por ‘baseline_quality_reports‘ y métricas en tiempo real.

13 Conclusiones

El sistema de prognosis industrial adaptativa convierte mediciones multivariadas en una cadena de valor completa: limpieza, selección, modelado, actualización, inferencia y gobierno. La selección multifactorial demuestra que es posible equilibrar variabilidad y estabilidad para priorizar sensores que reaccionan a cambios relevantes sin perder consistencia estadística, garantizando decisiones basadas en la dinámica real del activo. El ensemble SARIMAX–Prophet, vigilado por Isolation Forest, justifica teóricamente la captura simultánea de componentes lineales, no lineales y residuales inesperados, mientras que las pruebas ADF y KPSS aseguran que cada modelo opere en condiciones de estacionariedad controlada. El aprendizaje incremental y el control de drift mantienen la agnosticidad del sistema frente a distintos activos y lotes, preservando memoria histórica mediante versionado y persistencia transaccional en PostgreSQL. La metodología documentada es aplicable a múltiples industrias y constituye una referencia técnica para equipos de mantenimiento predictivo que buscan trazabilidad, auditabilidad y capacidad de adaptación continua.

A Tablas principales en PostgreSQL

Tabla	Descripción

normalized_data_table	Datos estandarizados con columna ‘timestamp‘, valores numéricos normalizados y metadatos asociados.
variable_mapping	Relación entre nombres originales y normalizados para mantener la trazabilidad.
selected_variables_history	Histórico de variables críticas y de monitoreo con scores y categorías.
variable_analysis_history	Almacena métricas (varianza, estabilidad, tendencia, correlación, score final) por variable y fecha de análisis.
baseline_models	Modelos activos del ensemble con parámetros, estadísticas base e intervalos de confianza.
baseline_model_versions	Versiones históricas con parámetros, métricas y metadata de entrenamiento.
baseline_metrics	Registros de RMSE, MAE, AIC, BIC, ratios de anomalías y otros indicadores.
baseline_validation	Resultados de validaciones temporales y diagnósticos asociados.
baseline_incremental_control	Control de lotes procesados, métricas de aprendizaje incremental y estado de actualización.
baseline_quality_reports	Reportes de calidad generados para auditoría y análisis de desempeño.

B Parámetros clave del sistema

Parámetro	**Valor en la implementación actual**
Usuario de base de datos	‘postgres‘
Contraseña	‘industrial2024‘ (configurable en despliegue)
Host y puerto	‘localhost:5432‘
Base de datos	‘prognosis_db‘
Ventana de entrenamiento	90 días (ajustable)
Frecuencia de actualización	1 día
Tamaño mínimo de datos	1000 registros por variable
Tamaño de lote incremental	1000 registros
Número máximo de versiones	5
Umbral de mejora significativa	0.1
Umbrales de alerta	‘warning=0.7‘, ‘critical=0.9‘

C Glosario de clases y métodos

Elemento	Descripción
`DataPreprocessor`	Clase que administra carga, limpieza, imputación, normalización y persistencia de datos.
`KeyVariableSelector`	Clase encargada de calcular scores multifactoriales, clasificar variables y registrar resultados.
`BaselineModeler`	Clase que coordina entrenamiento de SARIMAX y Prophet, vigilancia con Isolation Forest, versionado y control incremental.
`IndustrialFailurePredictor`	Clase responsable de calcular probabilidades de falla, generar alertas y producir reportes.
`SystemConfig`	Encapsula configuración de base de datos, parámetros del modelo y umbrales estadísticos.

D Buenas prácticas de operación

1. Validar periódicamente que las columnas de ‘normalized_data_table‘ mantengan consistencia con el mapeo y las unidades esperadas.
2. Revisar logs de ‘KeyVariableSelector‘ para detectar variables que pierden relevancia o sufren degradación de calidad.
3. Actualizar credenciales de base de datos según las políticas de la organización.
4. Documentar acciones tomadas tras cada alerta crítica y retroalimentar el sistema para mejorar la calibración de umbrales.
5. Planificar capacidad de almacenamiento y cómputo considerando el crecimiento del número de variables y la frecuencia de muestreo.

E Referencias

Referencias

[1] Aggarwal, C. C. (2017). *Outlier Analysis* (2.ª ed.). Springer.

[2] Box, G. E. P., Jenkins, G. M., Reinsel, G. C., & Ljung, G. M. (2015). *Time Series Analysis: Forecasting and Control* (5.ª ed.). Wiley.

[3] Hyndman, R. J., & Athanasopoulos, G. (2021). *Forecasting: Principles and Practice* (3.ª ed.). OTexts.

[4] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 413–422.

[5] Montgomery, D. C., Jennings, C. L., & Kulahci, M. (2015). *Introduction to Time Series Analysis and Forecasting* (2.ª ed.). Wiley.

[6] Ogata, K. (2010). *Ingeniería de control moderna* (5.ª ed.). Prentice Hall.