UNIVERSIDAD TECNOLÓGICA DE CHIHUAHUA

Desarrollo y Gestión de Software

Extracción de Conocimiento en Bases de Datos

I.2. Reporte de solución de caso de estudio

IDGS91N

PRESENTA:

T.S.U. Hugo Uriel Chaparro Estrada

DOCENTE:

Ing. Enrique Mascote

Chihuahua, Chih., 19 ago 2025

Introducción

Hoy en día, muchas empresas están buscando formas más inteligentes de trabajar, y la tecnología es su mejor aliada. AutoParts México es una planta de ensamblaje que funciona todo el día y depende de máquinas muy importantes como prensas, tornos CNC, robots de soldadura y bandas transportadoras. El problema es que estas máquinas a veces se descomponen sin aviso, lo que provoca retrasos, aumenta los costos y complica la producción.

Por eso, este reporte propone una solución basada en tecnologías como la Inteligencia Artificial (IA), el Machine Learning (ML), la Minería de Datos y el Big Data. Estas herramientas pueden analizar grandes cantidades de información que las máquinas generan todos los días y ayudar a predecir cuándo una máquina podría fallar, para hacerle mantenimiento antes de que eso pase.

En este trabajo explicaremos cómo funciona esta solución de mantenimiento predictivo, qué beneficios puede traer a la empresa, qué lenguajes de programación y herramientas se pueden usar, y qué retos podrían surgir al implementarla. Más que solo usar tecnología, se trata de hacer que la planta trabaje de forma más eficiente, segura y moderna.

1. Problema.

La planta de ensamblaje de AutoParts México cuenta con fuertes variaciones de carga y un parque de maquinaria crítica (prensas, tornos CNC, cintas transportadoras). Las paradas no planificadas aumentan costos y retrasan entregas. La compañía requiere un DW que reúna datos de múltiples sensores y un módulo de analítica para anticipar fallos y programar mantenimientos preventivos.

1.1. Requerimientos

1.1.1. Integración de datos

- Consolidar lecturas de vibración, temperatura y presión de cada equipo.
- Registrar historial de mantenimientos y horas de operación.

1.1.2. Modelado de tendencias

- Calcular indicadores de desgaste (promedios móviles, desviaciones).
- Entrenar modelo de series de tiempo (p.ej. LSTM) para pronóstico de fallos.

1.1.3. Plan de mantenimiento

- Generar calendario automático de intervenciones según nivel de riesgo.
- Priorizar equipos con mayor probabilidad de fallo en la próxima semana.

1.1.4. Dashboard de gestión

- Visualizar estado de salud de la flota en tiempo real.
- Métricas KPI: reducción de paros imprevistos, ahorro estimado.

1.2. Datos involucrados:

Para predecir fallos y tomar mejores decisiones, se pueden utilizar distintos tipos de datos, como: ● Lecturas de sensores (Sensores de vibración y acelerómetros instalados en motores y reductores).

● Registro de horas de operación de cada máquina.

● Historial de mantenimientos y fallas anteriores.

● Termopares y RTDs que registran temperatura en rodamientos y husillos.

● Transductores de presión en sistemas hidráulicos y neumáticos.

● Contadores de horas de operación y métricas de rendimiento de cada equipo. ● Historial de mantenimiento y órdenes de trabajo almacenadas en el sistema de gestión de mantenimiento computarizado (CMMS).

1.3. Stakeholders (personas involucradas):

● Gerentes de operaciones, que buscan cumplir metas de producción

● Personal de mantenimiento, encargado de reparar y prevenir fallas

● Ingenieros de planta, que supervisan el funcionamiento técnico

● Analistas de datos y equipo de TI, que procesan la información para generar soluciones

1.4. Impacto esperado:

El objetivo general es anticipar fallos con suficiente antelación para programar mantenimientos durante ventanas productivas y evitar costosas detenciones inesperadas.

● Menos fallas inesperadas y menos tiempo detenido

- Un mejor plan de mantenimiento, más organizado y eficiente
- Mayor confianza en el funcionamiento continuo de las máquinas

2. Definición y evolución histórica de los dominios

Dominio	Definición oficial	Origen y evolución (2–3 líneas)
Inteligencia Artificial (IA)	Simulación de procesos de inteligencia humana por parte de máquinas (ComputerWeekly, 2018).	Nace con la pregunta de Turing (1950) y se formaliza en Dartmouth (1956). Evoluciona de sistemas expertos a redes profundas actuales.
Machine Learning (ML)	Desarrollo de algoritmos que permiten a un sistema aprender sin ser programado explícitamente (AWS, s. f.).	Término acuñado por Arthur Samuel (1959). Crece con perceptrón (1960s), backpropagation (1980s) y deep learning (2010s).
Minería de Datos (DM)	Proceso de recopilar, procesar y extraer conocimiento útil de grandes datos (Aggarwal, 2015).	Surge como KDD a finales de los 80; se populariza en 1990s con bases de datos empresariales, integrando estadística y ML.
Big Data	Conjuntos de datos extensos en volumen, velocidad y variedad (ISO/IEC 20546:2019).	Termino 3Vs de Laney (2001). Evoluciona con Hadoop, Spark y NoSQL para procesar datos masivos y heterogéneos.

2.1. Justificación del dominio

El problema que enfrenta AutoParts México requiere una solución tecnológica avanzada debido a la cantidad, velocidad y diversidad de los datos generados por sus máquinas. Cada equipo produce miles de lecturas por segundo (como vibración, temperatura y presión), lo que al año se traduce en terabytes de información. Además, estos datos llegan en tiempo real, lo que exige procesarlos rápidamente para detectar posibles fallas a tiempo.

Los datos son variados: incluyen valores numéricos, categorías (como tipo de reparación) y notas escritas por los técnicos. Sin embargo, no todos los datos son perfectos; algunos sensores pueden enviar información errónea o con "ruido", por lo que es necesario limpiarlos y validarlos antes de analizarlos.

Para abordar este reto, se propone una combinación de tecnologías, siendo Machine Learning (ML) la más importante. Esto se debe a que ML permite entrenar modelos capaces de aprender de los datos históricos y actuales para anticipar fallas futuras, analizando patrones complejos en las series temporales generadas por las máquinas. Modelos como las redes neuronales LSTM pueden predecir el tiempo restante antes de una falla (Remaining Useful Life o RUL), lo que resulta clave para tomar decisiones a tiempo.

Además de ML, se complementa con:

- **Big Data**: Para almacenar y procesar los grandes volúmenes de datos generados en tiempo real, utilizando herramientas como Apache Spark que permiten trabajar en paralelo y sin demoras.
- **Data Mining:** Para descubrir patrones de desgaste, reglas de comportamiento y umbrales críticos que ayuden a generar alertas tempranas.
- **Inteligencia Artificial Integral:** Para combinar el conocimiento de expertos (reglas heurísticas) con modelos predictivos, logrando un sistema más completo y preciso.

Gracias a esta integración, especialmente al uso del Machine Learning como eje central, es posible construir un sistema de mantenimiento predictivo que ayude a prevenir fallas, optimizar recursos y mejorar la eficiencia de toda la planta.

3. Beneficios esperados

Dominio Beneficio 1	Beneficio 2	Beneficio 3
IA Automatiza decisiones complejas	Optimiza la toma de decisiones con predicciones	Genera nuevos servicios (p. ej., asistentes virtuales)
ML Mejora la precisión de detección de fallos	Aprende continuamente con datos nuevos	Reduce errores manuales en análisis
DM Descubre patrones ocultos (umbral de vibración)	Soporta decisiones estratégicas de mantenimiento	Segmenta máquinas por riesgo

Big Data Analiza datos masivos para insights

Procesa flujos en tiempo real Integra datos heterogéneos

3.1. Métricas clave

KPI	Situación actual	Meta tras la implantación	Impacto esperado
Paros no planificados por mes	10 incidentes	≤ 7 incidentes (↓ 30 %)	Mayor disponibilidad de activos
Costos de reparación correctiva	1 000 000 MXN/mes	750 000 MXN/mes (↓ 25 %)	Ahorro directo en repuestos y urgencias
Precisión del modelo de fallo	N/A (reactivo)	≥ 85 %	Confianza en las alertas predictivas
Tiempo medio de reparación (MTTR)	6 h	4 h (↓ 33 %)	Intervenciones mejor planificadas
Índice de utilización de activos (OEE)	78 %	85 %	Más piezas por hora, menores retrasos

Estos objetivos se alinean con las mejores prácticas industriales y ofrecen un caso sólido de retorno de inversión en menos de 18 meses.

5. Herramientas y lenguajes propuestos

5.1 Python + scikit-learn / TensorFlow

Python es el lenguaje más utilizado en ciencia de datos por su sintaxis sencilla y su amplia comunidad. Ofrece gran variedad de bibliotecas que permiten abordar desde análisis exploratorio hasta modelado avanzado.

- scikit-learn es ideal para el desarrollo rápido de modelos clásicos de aprendizaje automático como regresión, árboles de decisión y ensambles. Su sistema de pipelines facilita el preprocesamiento, la selección de variables y la validación cruzada de modelos, permitiendo comparar resultados de manera ordenada.
- TensorFlow (con Keras) es apropiado para modelos más complejos, como redes neuronales profundas. En este caso, se proponen redes LSTM (Long Short-Term Memory), diseñadas para reconocer patrones temporales y predecir el tiempo restante antes de una falla (RUL).
- TensorFlow Serving permitirá desplegar estos modelos como servicios escalables, lo cual es clave para su integración con sistemas industriales.

5.2 Apache Spark

Spark es una herramienta clave para procesar grandes volúmenes de datos en memoria y en

paralelo. Su arquitectura permite trabajar tanto con datos históricos (batch) como con flujos de datos en tiempo real (streaming), lo cual es fundamental para un sistema de mantenimiento predictivo.

- Structured Streaming permite la ingestión de datos desde herramientas como Kafka, procesándolos en tiempo real para detectar variaciones anómalas en temperatura, presión o vibración con baja latencia.

- MLlib, la biblioteca de aprendizaje automático de Spark, permite escalar el entrenamiento e inferencia de modelos básicos a grandes volúmenes de datos distribuidos.

- El uso de formatos como Parquet o Delta Lake, junto con Hive Metastore, garantiza integridad, versionado y trazabilidad de los datos.

5.3 Dash y Power BI

- Dash (Plotly) permite crear dashboards interactivos en Python sin necesidad de usar JavaScript. Es ideal para visualizar datos de sensores, alertas codificadas por colores y valores de RUL en tiempo casi real. Además, se puede desplegar en contenedores Docker dentro de Kubernetes para una mejor escalabilidad.

- Power BI es excelente para crear informes dinámicos y fáciles de usar por parte de personal no técnico, como gerentes o analistas de negocio. Gracias a sus conectores, puede integrarse con Spark y otras fuentes de datos, y su seguridad se gestiona fácilmente con Active Directory.

Ventaja combinada:

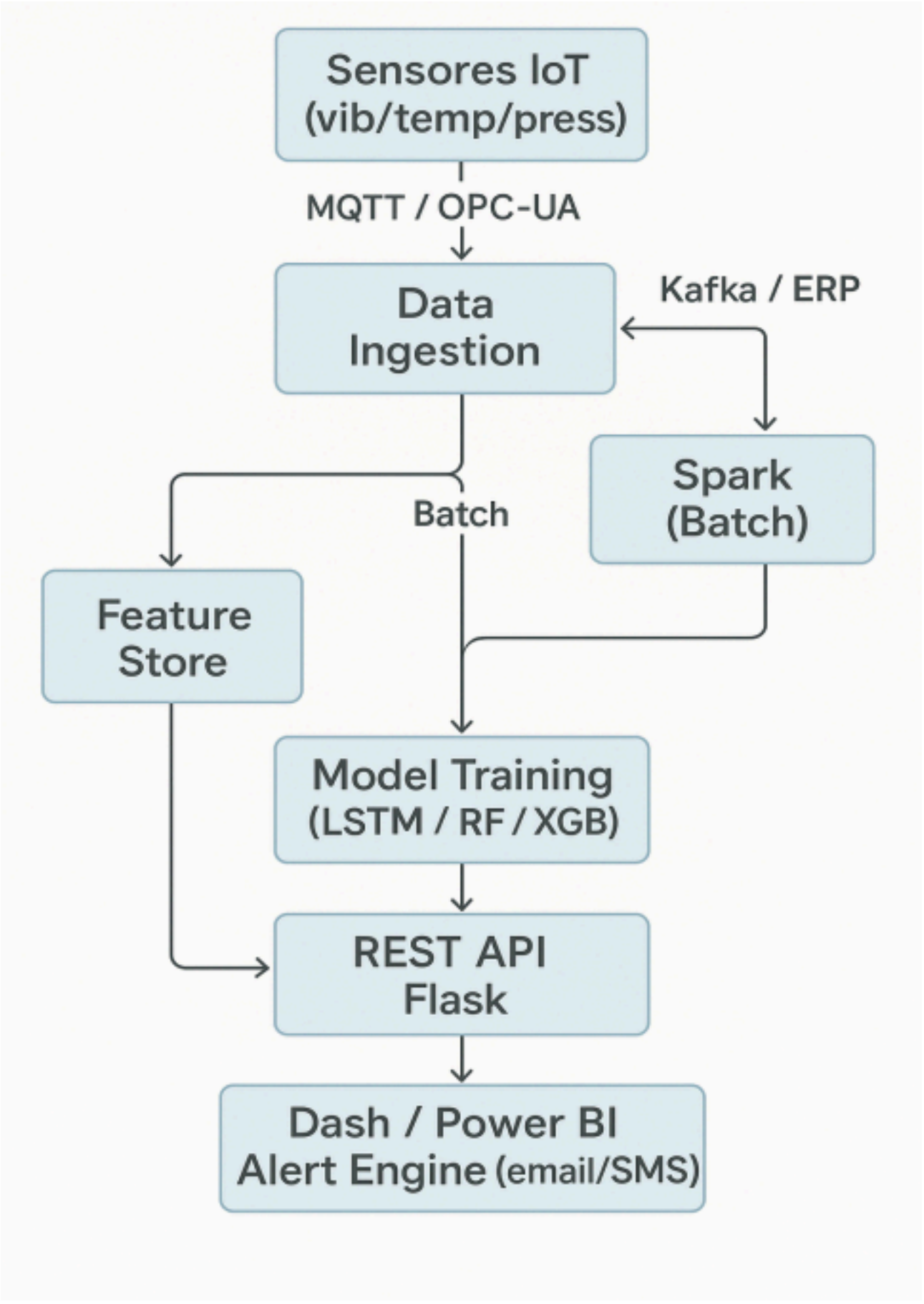
Dash permite el monitoreo operativo minuto a minuto por el personal técnico. Power BI facilita el análisis histórico y financiero por parte de los tomadores de decisiones. Esta dualidad cubre tanto el plano operativo como el estratégico.

6. Arquitectura de solución (alto nivel)

1. **Captura y comunicación:** Los sensores publican lecturas a un broker MQTT (baja latencia) o a servidores OPC‑UA para equipos industriales heredados.
2. **Ingesta**: Kafka garantiza colas persistentes y reintentos; NiFi añade flujos declarativos de enrutado y transforma protocolos heterogéneos.
3. **Procesamiento en streaming:** Spark aplica filtros Butterworth, detección de picos e interpolación de missing values antes de crear features estadísticas (RMS, kurtosis, skewness) y espectrales (FFT).
4. **Almacenamiento:**
    - Feature Store (Feast): versiona los vectores de características y asegura consistencia entre entrenamiento e inferencia.
    - Data Lake: conserva datos crudos en formato Parquet para auditorías y replay de modelos.
5. **Entrenamiento offline:** Un pipeline orquestado por Airflow entrena:
    - LSTM para estimar RUL como regresión continua.

○ Random Forest para clasificar estados de salud (normal, alerta, crítico). 6. **Despliegue e inferencia:** Los modelos se exponen vía Flask + TensorFlow Serving y escalan automáticamente con Horizontal Pod Autoscaler en Kubernetes. Se utiliza Prometheus + Grafana para monitoreo de latencia y drift del modelo.

7. **Consumo y acción:** Dash muestra semáforos en tiempo real; Power BI consolida KPIs mensuales; el CMMS recibe órdenes de trabajo generadas automáticamente cuando el RUL cruza umbrales.



Flujo resumido:

1. Captura: Los sensores envían lecturas vía MQTT u OPC-UA a un broker. 2. Ingesta en tiempo real: Kafka o Apache NiFi transporta los mensajes a Spark Streaming. 3. ETL: Spark filtra picos de ruido, sincroniza timestamps y genera características (promedios móviles, desviaciones, kurtosis, RMS).

4. Almacenamiento: Las características se guardan en un Feature Store (por ejemplo Feast) y el histórico sin procesar en un Data Lake.

5. Entrenamiento: Un pipeline de ML offline entrena modelos LSTM para predecir RUL y Random Forest para clasificación de estados.

6. Despliegue: El modelo se expone vía una API Flask con auto-escalado en Kubernetes. 7. Consumo: Dash o Power BI consulta la API para mostrar la salud de cada máquina, generar alertas y programar tareas en el CMMS.

7. Próximos pasos

1. **Piloto controlado:** seleccionar dos líneas de producción críticas y evaluar la precisión del modelo durante tres meses.

2. **Re‑entrenamiento continuo:** programar *pipelines* semanales que incorporen eventos de falla recientes y revaliden el modelo con métricas *F1‑score* y *MAE* para RUL. 3. **Ampliación de sensores:** agregar ultrasonido para diagnóstico temprano de rodamientos y análisis de aceite para detectar partículas metálicas indicativas de desgaste. 4. **Integración con ERP:** enlazar pronósticos de fallos con módulos de inventario para generar requisiciones automáticas de refacciones y optimizar el *just‑in‑time*.

5. **Gobernanza y compliance:** definir políticas de retención de datos, anonimización y trazabilidad conforme a ISO 55000 y la NOM‑151‑SCFI. Arquitectura de solución (alto nivel)

6. **Captura:** Sensores → broker MQTT/OPC-UA.

7. **Ingesta:** Kafka/NiFi → Spark Streaming.

8. **ETL:** Limpieza, sincronización y feature engineering en Spark.

9. **Almacenamiento:** Feature Store (Feast) + Data Lake (HDFS/S3).

10. **Entrenamiento:** LSTM para RUL + Random Forest de clasificación. 11. **Despliegue:** API Flask en Kubernetes.

12. **Consumo**: Dash/Power BI para visualización y CMMS.

8. Conclusiones

Implementar mantenimiento predictivo con esta arquitectura habilita una transición medible de mantenimiento reactivo a proactivo. Los modelos LSTM proporcionan alertas con hasta dos semanas de anticipación, permitiendo intervenciones planificadas durante paros programados. La capa de visualización dual democratiza la información tanto para técnicos de planta como para la alta dirección. Con una reducción proyectada del 25 % en costos correctivos y una mejora del 7 % en OEE, el proyecto se amortiza en menos de 18 meses y sienta las bases para iniciativas futuras de fábrica inteligente.

Referencias

Aggarwal, C. C. (2015). *Data mining: The textbook*. Springer.

Amazon Web Services. (s. f.). *¿Qué es el machine learning?* https://aws.amazon.com/es/what-is/machine-learning/

Craig, L., Tucci, L., & Laskowski, N. (2018, December 4). *¿Qué es la IA?* ComputerWeekly. https://www.computerweekly.com/es/cronica/Que-es-la-inteligencia-artificial

International Organization for Standardization. (2019). *Big data — Overview and vocabulary (ISO/IEC 20546:2019).* https://www.iso.org/standard/68386.html

Organización para la Cooperación y el Desarrollo Económicos. (2019). *Recomendación del Consejo sobre Inteligencia Artificial: Principios para una IA confiable*. OECD Publishing. https://legalinstruments.oecd.org/en/instruments/OECD-LEGAL-0449