



UNIVERSIDAD TÉCNICA DE AMBATO

**FACULTAD DE INGENIERÍA EN SISTEMAS, ELECTRÓNICA E
INDUSTRIAL**

CARRERA DE TELECOMUNICACIONES

Tema:

**SISTEMA IOT CON ANÁLISIS PREDICTIVO PARA EL MONITOREO DE
LA SALUD DE SISTEMAS DE BOMBEO DE CAMPOS PETROLEROS**

Trabajo de titulación modalidad Proyecto de Investigación, presentado previo a la
obtención del título de Ingeniero en Telecomunicaciones

ÁREA: Programación y redes

LÍNEA DE INVESTIGACIÓN: Electrónica, Telecomunicaciones y Sistemas de
control.

AUTOR: Erick Bladimir Bonilla Bonilla

TUTOR: Ing. Elizabeth Paulina Ayala Baño, Mg.

**Ambato - Ecuador
enero - 2026**

APROBACIÓN DEL TUTOR

En calidad de tutor del trabajo de titulación con el tema: SISTEMA IOT CON ANÁLISIS PREDICTIVO PARA EL MONITOREO DE LA SALUD DE SISTEMAS DE BOMBEO DE CAMPOS PETROLEROS, desarrollado bajo la modalidad Proyecto de Investigación por el señor Erick Bladimir Bonilla Bonilla, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, de la Universidad Técnica de Ambato, me permito indicar que el estudiante ha sido tutorado durante todo el desarrollo del trabajo hasta su conclusión, de acuerdo a lo dispuesto en el Artículo 17 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.3 del instructivo del reglamento referido.

Ambato, enero 2026.

**ELIZABETH
PAULINA
AYALA
BANO**

Firmado digitalmente por ELIZABETH
PAULINA AYALA BANO
DN: cn=ELIZABETH PAULINA
AYALA BANO c=EC o=SECURITY
DATA S.A. 2 ou=ENTIDAD DE
CERTIFICACION DE INFORMACION
Motivo: Soy el autor de este
documento
Ubicaci: n:
Fecha: 2026-01-08 13:52:05:00

Ing. Elizabeth Paulina Ayala Baño, Mg.
TUTOR

AUTORÍA

El presente trabajo de titulación con el tema: SISTEMA IOT CON ANÁLISIS PREDICTIVO PARA EL MONITOREO DE LA SALUD DE SISTEMAS DE BOMBEO DE CAMPOS PETROLEROS es absolutamente original, auténtico y personal y ha observado los preceptos establecidos en la Disposición General Quinta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato. En tal virtud, el contenido, efectos legales y académicos que se desprenden del mismo son de exclusiva responsabilidad del autor.

Ambato, enero 2026.



Erick Bladimir Bonilla Bonilla

C.C. 1804889465

AUTOR

DERECHOS DE AUTOR

Autorizo a la Universidad Técnica de Ambato para que reproduzca total o parcialmente este trabajo de titulación dentro de las regulaciones legales e institucionales correspondientes. Además, cedo todos mis derechos de autor a favor de la institución con el propósito de su difusión pública, por lo tanto, autorizo su publicación en el repositorio virtual institucional como un documento disponible para la lectura y uso con fines académicos e investigativos de acuerdo con la Disposición General Cuarta del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato.

Ambato, enero 2026.



Erick Bladimir Bonilla Bonilla

C.C. 1804889465

AUTOR

APROBACIÓN DEL TRIBUNAL DE GRADO

En calidad de par calificador del informe final del trabajo de titulación presentado por el señor Erick Bladimir Bonilla Bonilla, estudiante de la Carrera de Telecomunicaciones, de la Facultad de Ingeniería en Sistemas, Electrónica e Industrial, bajo la Modalidad Proyecto de Investigación, titulado SISTEMA IOT CON ANÁLISIS PREDICTIVO PARA EL MONITOREO DE LA SALUD DE SISTEMAS DE BOMBEO DE CAMPOS PETROLEROS, nos permitimos informar que el trabajo ha sido revisado y calificado de acuerdo al Artículo 19 del Reglamento para la Titulación de Grado en la Universidad Técnica de Ambato y el numeral 6.4 del instructivo del reglamento referido. Para cuya constancia suscribimos, conjuntamente con el señor Presidente del Tribunal.

Ambato, enero 2026.



Ing. Franklin Oswaldo Mayorga Mayorga, Mg.
PRESIDENTE DEL TRIBUNAL



Ing. Patricio Córdova Córdova, Mg.
PROFESOR CALIFICADOR



Dr. Freddy Benalcázar Palacios, Mg.
PROFESOR CALIFICADOR

DEDICATORIA

A mi madre, Elizabeth Guevara, y a Marina Bonilla, por su amor incondicional, por cada consejo, sacrificio y gesto de cariño que han marcado mi vida. Gracias por creer en mí incluso cuando las circunstancias fueron difíciles, por enseñarme a ser perseverante y por ser el motor que me impulsó a seguir adelante en este camino académico y personal.

A mi padre, Samuel Bonilla Acosta, por su esfuerzo, trabajo incansable y el ejemplo de responsabilidad y entrega que siempre me ha demostrado. Gracias por brindarme su apoyo, por motivarme a superarme cada día y por estar presente en los momentos más importantes de mi formación.

Finalmente, a mis amigos Erick, Jonathan, Kevin y Paul, por su compañía sincera, por las palabras de ánimo en los momentos de estrés, por las risas compartidas y por hacer de este proceso una etapa más llevadera y significativa.

Este logro se lo dedico a cada uno de ustedes, porque sin su apoyo, comprensión y cariño, este sueño no habría sido posible.

AGRADECIMIENTO

En primer lugar, agradezco a Dios por la vida, la salud y la fortaleza concedidas para culminar esta etapa de mi formación; Su guía y protección han sido fundamentales para alcanzar este logro.

A mi madre, Elizabeth Guevara Bonilla, a Marina Bonilla Sánchez y a mi padre, Samuel Bonilla Acosta, por su amor incondicional, su apoyo permanente y su ejemplo de esfuerzo y perseverancia. Sus consejos, sacrificios y motivación fueron el pilar que me sostuvo en los momentos más desafiantes.

A mi tutora de tesis, Ing. Elizabeth Paulina Ayala Baño, por su orientación académica, sus valiosas sugerencias y el tiempo dedicado a la revisión y mejora de este trabajo, aportando de manera decisiva al desarrollo de la investigación.

A los docentes de la FISEI, por compartir sus conocimientos y experiencias, contribuyendo de forma significativa a mi formación profesional y personal.

Finalmente, expreso mi agradecimiento a familiares, amigos, compañeros y a todas las personas que, con su apoyo y palabras de ánimo, fueron parte de este importante logro en mi vida.

Erick Bladimir Bonilla Bonilla.

ÍNDICE GENERAL DE CONTENIDOS

PORTADA.....	i
APROBACIÓN DEL TUTOR.....	ii
AUTORÍA.....	iii
DERECHOS DE AUTOR	iv
APROBACIÓN DEL TRIBUNAL DE GRADO.....	v
DEDICATORIA.....	vi
AGRADECIMIENTO	vii
ÍNDICE GENERAL DE CONTENIDOS	viii
ÍNDICE DE TABLAS.....	xii
ÍNDICE DE FIGURAS.....	xiv
ÍNDICE DE ANEXOS	xviii
RESUMEN EJECUTIVO	xix
ABSTRACT	xx
CAPÍTULO I. MARCO TEÓRICO.....	1
1.1 Tema de investigación.....	1
1.1.1 Planteamiento del problema.....	1
1.2 Antecedentes investigativos	3
1.3 Fundamentación teórica	8

1.3.1 Transformación Digital Industrial.....	8
1.3.2 Monitoreo inteligente y automatización.....	10
1.3.3 Sistema IoT con análisis predictivo	12
1.3.4 Algoritmos de Machine Learning	18
1.3.5 Análisis y visualización de datos en tiempo real	29
1.3.6 Indicadores de desgaste mecánico y eficiencia	33
1.3.7 Parámetros de funcionamiento óptimo.....	35
1.3.8 Mantenimiento y ciclo de vida.....	36
1.3.9 Hardware y software	38
1.3.10 Enfoque metodológico para Machine Learning.....	48
1.4 Objetivos	50
1.4.1 Objetivo general.....	50
1.4.2 Objetivos específicos	50
CAPÍTULO II. METODOLOGÍA.....	51
2.1 Materiales.....	51
2.2 Métodos.....	51

2.2.1 Modalidad de la investigación	51
2.2.2 Población y muestra	52
2.2.3 Recolección de información.....	52
2.2.4 Procesamiento y análisis de datos	53
CAPÍTULO III. RESULTADOS Y DISCUSIÓN.....	55
3.1 Desarrollo de la metodología CRISP-DM	56
3.1.1 Comprensión de negocio.....	56
3.1.2 Comprensión de datos.....	57
3.1.3 Preparación de datos	62
3.1.4 Modelado.....	67
3.1.5 Evaluación.....	70
3.1.6 Despliegue.....	80
3.2 Pruebas de funcionamiento del sistema	90
3.2.1 Resultados de pruebas de funcionamiento	93
3.2.2 Evaluación y validación del modelo	95
3.3 Presupuesto	100
CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES	102
4.1 Conclusiones	102
4.2 Recomendaciones.....	104

REFERENCIAS BIBLIOGRÁFICAS.....	105
ANEXOS.....	112

ÍNDICE DE TABLAS

Tabla 1. Estándares IoT de industria petrolera.....	16
Tabla 2 Alineación del servicio de Conectividad IoT con la industria de hidrocarburos en el Ecuador	17
Tabla 3. Comparación de modelos de entrenamiento Random Forest, XGBoost, LightGBM y CatBoost	23
Tabla 4. Estructura de la matriz de confusión de clasificación.....	25
Tabla 5. Análisis comparativo de las plataformas de visualización IoT	32
Tabla 6. Indicadores Mecánicos de Desgaste directos e indirectos.	34
Tabla 7. Parámetros de Funcionamiento de la Bomba Electrosumergible (ESP).....	36
Tabla 8. El ciclo operacional y los niveles de mantenimiento.....	37
Tabla 9. Especificaciones técnicas del equipo portátil	38
Tabla 10. Comparativa de entornos de entrenamiento Cloud para Machine Learnig.	40
Tabla 11. Análisis comparativo de plataformas de desarrollo orientadas al control y gestión de información.....	42
Tabla 12. análisis comparativo de bases de datos de series temporales.....	44
Tabla 13. Características de instalación de Docker Desktop.	45
Tabla 14. Recolección de información.....	53
Tabla 15. Distribución de registros por tipo de variables y fuente de datos.	59
Tabla 16. Estructura del DataSet_BombaESP.csv	61
Tabla 17. Codificación unificada de las etiquetas de salida.....	63

Tabla 18. Distribución de registros por tipo de falla.....	66
Tabla 19. Hiperparámetros configurados y su función en cada modelo.	69
Tabla 20. Análisis Comparativo de los Modelos Entrenados.	70
Tabla 21. Métricas para la etiqueta variable_falla.	71
Tabla 22. Métricas para la etiqueta subcomponente_falla.	71
Tabla 23. Métricas para la etiqueta severidad_falla.	72
Tabla 24. Métricas para la etiqueta estado_bomba.	72
Tabla 25. Métricas para la etiqueta falla_predicha_24h.	72
Tabla 26. Métricas globales de validación del modelo entrenado.	73
Tabla 27. Métricas de clasificación por clase para la salida variable_falla	96
Tabla 28. Métricas de clasificación por clase para la salida subcomponente_falla ...	97
Tabla 29. Métricas de clasificación por clase para la salida severidad_falla.	98
Tabla 30. Métricas de clasificación por clase para la salida estado_bomba.	99
Tabla 31. Métricas de clasificación por clase para la salida falla_predicha_24h. ...	100
Tabla 32. Resumen de métricas globales del modelo desplegado para cada salida de clasificación.....	100
Tabla 33. Recursos de hardware utilizados para el desarrollo y despliegue del sistema IoT-ESP.....	101

ÍNDICE DE FIGURAS

Figura 1. La Industria 4.0 y sus componentes para el sector O&G.	8
Figura 2. Arquitectura IoT de 3 y 4 capas.....	12
Figura 3. Matriz de confusión de tipo clasificación multiclase.	27
Figura 4. Arquitectura general de Grafana.....	30
Figura 5. Arquitectura de Kibana.....	31
Figura 6. Arquitectura de Power BI.	32
Figura 7. Arquitectura de Docker.....	46
Figura 8. Fases de trabajo de la metodología CRISP-DM.	49
Figura 9. Estructura del sistema IoT.	55
Figura 10. Distribución de registros por variable.....	60
Figura 11. Distribución de registros por variable.....	62
Figura 12. Script de unificación de identificadores y variables numéricas.....	63
Figura 13. Script de limpieza de duplicados, valores faltantes y outliers.	64
Figura 14. Aplicación de SMOTE para el rebalanceo de clases.	65
Figura 15. Dataset consolidado de bombas ESP.	65
Figura 16. Distribución de registros por tipo de falla.	66
Figura 17. División del dataset para el entrenamiento.	67
Figura 18. Proceso de entrenamiento de modelos machine learning.	68
Figura 19. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida variable _falla.	74

Figura 20. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida subcomponente_falla.	74
Figura 21. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida severidad_falla.	75
Figura 22. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida estado_bomba.	75
Figura 23. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida falla_predicha_24h.....	76
Figura 24. Curvas de confianza entre Precision–Recall variable falla.....	77
Figura 25. Curvas de confianza entre Precision–Recall subcomponente falla.	78
Figura 26. Curvas de confianza entre Precision–Recall severidad falla.	78
Figura 27. Curvas de confianza entre Precision–Recall de falla en 24 horas.	79
Figura 28. Curvas de confianza entre Precision–Recall del estado de la bomba.....	79
Figura 29. Proceso de entrenamiento del modelo Random Forest.....	80
Figura 30. Estructura del servicio de inferencia de la API.	82
Figura 31. Arquitectura del cliente y registro en InfluxDB.	83
Figura 32. Estructura del directorio IoT_ESP para la contenerización del sistema monitoreo predictivo.	84
Figura 33. Selección de la ruta previa al despliegue.	85
Figura 34. Construcción de los contenedores.	85
Figura 35. Levantamiento de los contenedores del sistema IoT_ESP.	86
Figura 36. Interfaz de Docker Desktop con los contenedores activos.	86
Figura 37. Interfaz de configuración inicial de InfluxDB.....	88

Figura 38. Configuración de la fuente de datos InfluxDB en Grafana.	88
Figura 39. Diseño de dashboard en Grafana.	89
Figura 40. Configuración del canal de notificación por correo electrónico en Grafana.	90
Figura 41. Panel de Grafana para la Bomba 1.	90
Figura 42. Panel de Grafana para la Bomba 11.	91
Figura 43. Panel de Grafana para la Bomba 6.	92
Figura 44. Registro de correos electrónicos de alerta en Gmail.	93
Figura 45. Registro de predicciones del modelo desplegado.	94
Figura 46. Matriz de confusión de la salida variable _falla del modelo de clasificación multietiqueta.	95
Figura 47. Matriz de confusión de la salida subcomponente _falla del modelo de clasificación multietiqueta.	96
Figura 48. Matriz de confusión de la salida severidad _falla del modelo de clasificación multietiqueta.	97
Figura 49. Matriz de confusión de la salida estado _bomba del modelo de clasificación multietiqueta.	98
Figura 50. Matriz de confusión de la salida falla _predicha _24h del modelo de clasificación multietiqueta.	99
Figura 51. Variable Dependiente.	112
Figura 52. Variable Independiente.	112
Figura 53. Constelación de ideas de la variable dependiente.	113
Figura 54. Constelación de ideas de la variable independiente.	113

Figura 55. Dashboard de grafana.	114
Figura 56. Respuesta de API.	114
Figura 57. Archivo docker-compose.yml.	119

ÍNDICE DE ANEXOS

Anexo A. Categorización de variables.....	112
Anexo B. Pruebas Adicionales.....	114
Anexo C. Código de entrenamiento del modelo Random Forest.....	115
Anexo D. Líneas de despliegue para Docker.	119

RESUMEN EJECUTIVO

La operación de bombas electrosumergibles (ESP) en campos petroleros se ve afectada por fallas, tiempos de parada y una limitada utilización de herramientas digitales para el monitoreo de condición. Frente a esta problemática, se desarrolló un sistema IoT con analítica predictiva orientado al monitoreo de la salud de bombas ESP, con el propósito de apoyar la toma de decisiones de mantenimiento y disminuir la ocurrencia de fallas críticas en contextos de producción continua.

Para el desarrollo de la solución se aplicó la metodología CRISP-DM, definiendo un flujo de trabajo de análisis predictivo a partir de datos históricos. En esta etapa se integraron, estructuraron y depuraron registros operativos con variables eléctricas, hidráulicas, mecánicas y térmicas, y se ejecutó un preprocesamiento con validaciones, limpieza y transformación de características para asegurar consistencia y calidad. Con esta base, se construyó un conjunto de datos multietiqueta con cinco objetivos de clasificación, lo que permitió abordar de forma conjunta tareas de diagnóstico y pronóstico. Posteriormente, se compararon modelos de ensamble basados en árboles de decisión y se adoptó una solución basada en Random Forest multietiqueta por su equilibrio entre desempeño, robustez y capacidad de generalización.

Durante pruebas en un entorno controlado, el sistema obtuvo una precisión ponderada global aproximada del 85 % y una exactitud del 82 % en variable de falla, 83 % en subcomponente, 82 % en severidad, 83 % en estado de la bomba y 91 % en predicción de falla a 24 horas. Finalmente, el despliegue se realizó con Docker, una API en FastAPI, InfluxDB y paneles en Grafana, habilitando la visualización de variables y del estado estimado, y evidenciando la viabilidad técnica para mantenimiento predictivo.

Palabras claves: Internet de las cosas (IoT), mantenimiento predictivo, bombas electrosumergibles (ESP), aprendizaje automático, Grafana.

ABSTRACT

The operation of electric submersible pumps (ESPs) in oil fields is impacted by failures, downtime, and the limited use of digital tools for condition monitoring. To address this challenge, an IoT system with predictive analytics was developed to monitor the health of ESPs, with the aim of supporting maintenance decision-making and reducing the occurrence of critical failures in continuous production environments.

To develop the solution, the CRISP-DM methodology was applied, defining a predictive analytics workflow based on historical data. In this stage, operational records were integrated, structured, and refined using electrical, hydraulic, mechanical, and thermal variables, and a preprocessing pipeline was implemented—including validation, cleaning, and feature transformation—to ensure data consistency and quality. On this basis, a multi-label dataset with five classification targets was built, enabling the joint execution of diagnostic and prognostic tasks. Subsequently, ensemble models based on decision trees were compared, and a multi-label Random Forest approach was adopted due to its balance of performance, robustness, and generalization capability across multiple classes and operational variability.

During controlled-environment testing, the system achieved an approximate overall weighted precision of 85%, with per-label accuracies of 82% for failure variable, 83% for subcomponent, 82% for severity, 83% for pump state, and 91% for 24-hour failure prediction. Finally, deployment was carried out using Docker, a FastAPI inference service, InfluxDB, and Grafana dashboards, enabling visualization of process variables and estimated operating status, and demonstrating the technical feasibility of the proposed solution for predictive maintenance.

Keywords: Internet of Things (IoT), predictive maintenance, electric submersible pumps (ESP), machine learning, Grafana.

CAPÍTULO I. MARCO TEÓRICO

1.1 Tema de investigación

“SISTEMA IOT CON ANÁLISIS PREDICTIVO PARA EL MONITOREO DE LA SALUD DE SISTEMAS DE BOMBEO DE CAMPOS PETROLEROS”

1.1.1 Planteamiento del problema

La industria del petróleo, si bien sigue siendo un pilar para la economía global, continúa teniendo una adopción de tecnologías IoT muy limitada, lo que beneficia las brechas operativas que surgen con el uso de sistemas heredados que carecen de sensores e infraestructura moderna. Esta falta de digitalización contribuye a niveles superiores de fallas en los activos, de tiempos muertos no planificados y de costos operativos, incluso cuando se ha demostrado que la implementación de IoT puede disminuir gastos entre un 10 % y un 25 % como resultado de monitoreos en tiempo real [1], [2]. A esto se suman las barreras culturales, administrativas y técnicas que dificultan la posibilidad de integrar automatización en los procesos y que afectan el mantenimiento de activos críticos. Así, surge la necesidad de crear análisis predictivos basados en IA, Machine Learning y gemelos digitales que anticipen fallas y aumenten la confiabilidad del sistema [3].

En América Latina se han identificado limitaciones estructurales que afectan la digitalización del sector energético, destacándose los altos costos de implementación y una regulación inadecuada que restringe el avance en un 37 %. Asimismo, un 33 % de los actores percibe una infraestructura tecnológica débil, acompañada de una limitada cultura digital y un déficit de competencias técnicas. La adopción de IoT, sensores y sistemas AMI permanece en fase incipiente, lo que dificulta la captación de datos en tiempo real necesaria para la optimización operativa. Como consecuencia, la ausencia de tecnologías habilitadas por IoT, como los gemelos digitales, impide el desarrollo de un mantenimiento predictivo eficaz, incrementando los costos correctivos y el riesgo de interrupciones en la continuidad del servicio [4].

En el caso de Ecuador, se puede ver que la adopción de tecnologías avanzadas como la IA y la digitalización es escasa, en comparación con Brasil, México y Colombia, que han alcanzado notas media y alta. Esta diferencia se debe a que hay infraestructura convencional, hay poca digitalización de los procesos y poca formalización operativa en el sector de la energía. La falta de políticas gubernamentales claras pone también límites en la adopción de estas tecnologías, creando tiempos de respuesta bajos a las fallas y los costes adicionales que viene dado por los modelos administrativos tradicionales. La adopción completa de tecnologías relacionadas con la IA puede en esta forma verse como una oportunidad para mejorar la eficiencia operativa y mejorar la producción del país. [5].

En la región del Oriente ecuatoriano, donde se desarrolla la producción petrolera, la digitalización se torna aún más intensa, puesto que muchos de los pozos siguen funcionando con modelos manuales y tradicionales que limitan el uso de IoT y las capacidades de análisis avanzado de datos. Incluso aunque sí se han incorporado sensores sísmicos en fases de exploración, la IA sigue en una fase rudimentaria y no hay plataformas con redes de sensores que habiliten retroalimentación en tiempo real. Esta falta de digitalización sostiene problemas operativos de larga data, como una baja capacidad de respuesta ante fallos, así como una gestión ineficiente de los campos maduros, puesta de manifiesto por el hecho de que algunos sistemas SCADA usados carecen de algoritmos que hagan predicciones. Por este motivo, la utilización de mantenimiento predictivo y la optimización de caudales apoyada en IA e IoT se presentan como alternativas claves para disminuir riesgos, aumentar las eficiencias y atender las demandas ambientales y sociales de los territorios amazónicos [5].

El escaso grado de digitalización y automatización IoT en la industria del petróleo está vinculado de manera prevalente a la escasa inversión en innovación tecnológica, a la débil infraestructura digital y a la no existencia de políticas públicas para la modernización industrial. Todo ello hace que sea difícil desarrollar plataformas inteligentes para recolectar, procesar y analizar datos en tiempo real, lo que limita la eficiencia operativa. Le añade el escaso número de personas especialistas y la resistencia al cambio tecnológico que demoran su adopción como una solución para la toma de decisiones y la gestión energética. Por todo ello, la modernización de la misma

dependerá de tomar medidas a fin de solventar dicha brecha estructural y pasar a realizar procesos más seguros, eficientes y sostenibles.

El sector petrolero sufre una disminución en la competitividad y la eficiencia de la operación y de su pavimentación por insuficiencia de capacidad para reaccionar de una forma rápida y precisa ante fallas o cambios de los sistemas de bombeo, lo cual provoca un aumento de los precios de mantenimiento y un incremento en los tiempos de actividad. Esto implica, además, que el patrimonio técnico exponga a una situación de mayor riesgo con respecto a la ciberseguridad, lo cual afecta la continuidad operativa. Por otro lado, la baja integración de las tecnológicas impide el uso de energías renovables y, por ende, obstaculiza la sostenibilidad a corto o largo plazo. Por lo que hace falta la adopción de soluciones avanzadas que fortalezcan la respuesta operativa y fortalezcan la resiliencia del sistema energético.

Por esto, se propone un desarrollo de un sistema IoT con análisis predictivo de los sistemas de bombeo para reducir el tiempo de inactividad y de incrementación de la durabilidad de los sistemas, en pro de fortalecer la eficiencia energética y de la sostenibilidad económica de la infraestructura del sector.

1.2 Antecedentes investigativos

En la última década, el auge de sensores inteligentes, plataformas IoT industriales y algoritmos de aprendizaje automático ha transformado el mantenimiento de equipos críticos en campos petroleros. Esta revisión integra cinco artículos y dos proyectos de investigaciones de modelos aprendizaje automático (Machine Learning, ML) en las bombas sumergibles eléctricas (ESP) empleadas en la producción petrolera. Contribuyendo como partida y siendo referencia para apoyar esta investigación.

En el año 2024, en Diponegoro (Indonesia), los investigadores Singgih Saptadi, Achmad Widodo, Muhammad Fitrah Athaillah y Muhammad Farid Ayyasyi, desarrollaron el análisis titulado “Implementing Machine Learning Methods in Predicting Failures in Electrical Submersible Pump Machines”. En este estudio proponían un modelo de Machine Learning orientado en predecir la probabilidad de fallas en sistemas de bombas sumergibles eléctricas (ESP). Como resultado final de la

investigación, se obtuvo un tablero interactivo (dashboard) que ser capaz de visualizar en tiempo real los datos sensoriales y la predicción obtenida del modelo, el cual permite detectar anomalías operativas en forma temprana, a partir de analizar los registros obtenidos por medio de los sensores de las bombas ESP. El modelado de los datos fue posible a partir de la comparación de dos algoritmos supervisados: K-Nearest Neighbor (KNN) y Decision Tree (Árbol de Decisión). Ambos métodos fueron optimizados mediante la técnica de sobremuestreo SMOTE (Synthetic Minority Over-sampling Technique) para poder equilibrar las clases y mejorar la capacidad de generalización del modelo. Como resultado, el algoritmo KNN se encontró como el mejor, con métricas promedio (Macro Average) de precisión (Accuracy) 0.9127, precisión (Precision) 0.91, sensibilidad (Recall) 0.91 y un F1-Score 0.91, lo que evidencia un alto nivel de exactitud y estabilidad en la predicción de fallas en bombas sumergibles eléctricas [6].

Asimismo, en la investigación desarrollada por Saker Ibtissem en la Université Kasdi Merbah Ouargla (Argelia) en el año 2024, titulada “Predictive Electrical Submersible Pumps (ESPs) Performance for Better Production Sustainability Using Machine Learning”, se estableció un modelo predictivo fundamentado en datos orientado a detectar de forma temprana fallas en bombas sumergibles eléctricas (ESP). Se pretendió con ello facilitar el tránsito de las estrategias de mantenimiento reactivo hacia un modelo más proactivo y predictivo (Predictive Maintenance, PdM), contribuyendo a lograr una mayor sostenibilidad en la producción petrolera. El flujo de trabajo analítico se definió dentro de la plataforma KNIME (Konstanz Information Miner), una herramienta de código abierto para la integración, el procesamiento y el análisis de datos, además de H2O, una plataforma de Machine Learning de código abierto para Big Data y el análisis predictivo a gran escala. Los experimentos realizaron una evaluación de diferentes algoritmos de Machine Learning, entre los que se incluyó SMOTE (Synthetic Minority Over-Sampling Technique), Random Forest (RF) y Decision Tree (DT). Los resultados del trabajo mostraron que el algoritmo Random Forest ofrecía el mejor rendimiento predictivo, alcanzando valores de exactitud (Accuracy) de hasta el 100 % y un área bajo la curva (AUC) de 1, lo que implica una capacidad de clasificación perfecta y además un alto grado de fiabilidad en la detección de fallas de las bombas ESP. [7].

En el año 2025, en Teherán (Irán), investigadores de la Amirkabir University of Technology, Yasin Khalili, Mohammad Ahmadi y Mostafa Keshavarz Moraveji, desarrollaron un estudio titulado “Mantenimiento predictivo consciente del tiempo de bombas sumergibles eléctricas usando aprendizaje de conjunto CatBoost y etiquetado basado en tendencias”. La investigación establece un sistema de mantenimiento predictivo para el monitoreo de bombas sumergibles eléctricas (ESP) a través de técnicas avanzadas de machine learning. El algoritmo desarrollado lleva a cabo la clasificación del estado operativo de las bombas en tres categorías: estable, inestable y falla. Así mismo, incluye una estrategia de etiquetado basada en tendencias (trend based labeling), así como también una tubería de procesamiento híbrida que une Isolation Forest con el objetivo de la detección de valores atípicos y PCA (Análisis de Componentes Principales) para la reducción de la dimensionalidad. En la evaluación del entrenamiento se utilizaron cuatro modelos: CatBoost (Gradiente de Aumento Categórico), ExtraTrees (Extremely Randomized Trees), Support Vector Machines (SVM), y Multilayer Perceptrons (MLP). Los datos utilizados provienen de sensores SCADA y contemplan variables que incluyen temperatura de motor, presión de la admisión, presión de descarga, corriente, voltaje del motor y vibraciones en ejes X e Y. El modelo CatBoost tuvo el mejor desempeño, alcanzando una precisión del 99,85 %, exactitud del 99,85 %, recall del 92,82 % y un F1-Score de 99,83 %. Además, logró una exhaustividad de 96,5 % en la clase de falla, que es un aspecto clave al reducir la tasa de falsos negativos cuando se trata de la detección de anomalías críticas [8].

En el año 2025, en la McDougall School of Petroleum Engineering de la University of Tulsa (Tulsa, OK, EE. UU.), los investigadores Haiwen Zhu, Hong Yu, Qiang Sun, Qiuchen Wang, Haorong Jing y Rakhymzhan Abdikadyrov publicaron el artículo titulado “Deep Learning-Based Performance Prediction of Electric Submersible Pumps Under Viscous and Gas–Liquid Flow Conditions”. Se planteó en este estudio el uso de un ámbito basado en aprendizaje profundo, donde se utilizó una Red Neuronal Totalmente Conectada (Fully Connected Neural Network, FNN) para hacer predicción del rendimiento de las Bombas Sumergidas Eléctricas (BSE o ESP) bajo condiciones de flujo viscoso con o sin gas. Los parámetros considerados para la evaluación del modelo fueron los caudales de líquido y de gas, la velocidad de rotación de la bomba (RPM), los diferenciales de presión a través de etapas, la viscosidad del

fluido de trabajo y la relación gas-líquido. Con el fin de aumentar la exactitud en las predicciones también se consideraron variables adicionales como la fracción de vacío de gas (Gas Void Fraction, GVF) así como la densidad de la mezcla de fluidos y la viscosidad. El modelo FNN presentó mejores resultados frente a otros seis métodos de aprendizaje automático al tener mayor capacidad de predicción, flexibilidad y robustez al predecir el funcionamiento de las bombas bajo condiciones operativas complejas, a la vez que registró los menores valores de error cuadrático medio (MSE) y error absoluto medio (MAE), confirmando así su capacidad de modelar el comportamiento dinámico del sistema de bombeo con alta precisión. [9].

En el año 2025, en Beijing (China), los investigadores Cong Zhang, Guoqing Han, Liping Zhao, Chunsheng Wu, Bin Fan, Bin Liu, Zezhou Li y Jiping Deng desarrollaron el estudio titulado “Data-Driven Fault Prediction for Electric Submersible Progressing Cavity Pump Wells”. En este estudio se presentó un framework PCA–LSTM reproducible e interpretable para el diagnóstico y la predicción de fallas basado en datos, con capacidad de emitir alertas tempranas sobre posibles anomalías operativas. El modelo de red neuronal fue ejecutado sobre la plataforma TensorFlow 2.17.0 y fue aplicado al análisis de bombas de cavidad progresiva electrosumergibles (ESPCP) utilizadas en Gas de Lecho de Carbón (CBM). Esta implementación centrándose en el enfoque que integró Analítica de Componentes Principales (PCA) con la red Long Short-Term Memory (LSTM) para determinar características temporales y reducir la dimensión de los datos. Como modelos de referencia (baselines) fueron empleados los ARIMA (AutoRegressive Integrated Moving Average) y GBDT (Gradient Boosting Decision Trees), mientras que la técnica K-Means clustering junto a PCA permitió la clasificación de las diferentes condiciones de operación. El modelo PCA–LSTM resultó con el mayor desempeño de los demás modelos, con un 15 % de mejora en comparación con el modelo ARIMA y un 9 % en comparación con el modelo GBDT, alcanzando así una exactitud de validación del 72,22 %, junto con una ratio de aceptación del 86,79 %. La mayor exactitud se obtuvo en la clase "Perforation" (Perforación) con un 81 % y la menor en "Sand Sticking" (Pegado por Arena) con un 65 %. Finalmente, el error absoluto medio (MAE) fue de 0,0326 en el conjunto de entrenamiento y de 0,0345 en el conjunto de validación, probando así la robustez del modelo en el pronóstico temprano de fallas en sistemas ESPCP. [10].

En el año 2025, en Qingdao (China), los investigadores Faming Gong, Siyuan Tong, Chengze Du, Zhenghao Wan y Shiyu Qiu desarrollaron el estudio titulado “A Fault Diagnosis Model of an Electric Submersible Pump Based on Mechanism Knowledge”. En este trabajo se formuló un modelo de diagnóstico de fallas correspondiente a las bombas sumergibles eléctricas (ESP), llamado MK-ESPFD (Mechanism Knowledge-Integrated Electric Submersible Pump Fault Diagnosis Model), que presenta un diseño de integración entre un conocimiento mecanicista y las técnicas de aprendizaje en un marco diagnóstico como es el caso de la tecnología de diagnóstico multinivel. El modelo es una conjunción de distintas arquitecturas diseñadas de inteligencia artificial: una Red Neuronal Convolutiva de Contracción basada en Meta-Aprendizaje (Meta-Learning Convolutional Shrinkage Neural Network, SNN) orientada a la inferencia de síntomas de fallas; una red de memoria larga-corta bidireccional de campo aleatorio condicional (BiLSTM-CRF) empleada para el diagnóstico específico de fallas; una Red Neuronal Gráfica (GNN) utilizado el análisis estructural de los subgrafos de fallas; y una máquina de inferencia basada en reglas expertas (Expert Rule-Based Inference Machine, ERIM) para incorporar las limitaciones de precisión de los modelos neuronales. La propuesta de MK-ESPFD alcanzó el máximo rendimiento diagnóstico general, con un valor de recall del 93,35 % y un valor de F1-Score un 89,93 %, mostrando la capacidad para detectar fallas en los sistemas ESP entendiendo fallas de manera precisa y contextualizada mediante el uso de conocimiento experto y aprendizaje profundo. [11].

En el mismo sentido, en la investigación titulada “Detecting Electrical Submersible Pump (ESP) Failures and Estimating Run Life Using Artificial Neural Networks”, desarrollada por Mostafa Ahmed Sobhy en Egipto durante el año 2025, fue elaborado un marco integral de Machine Learning (ML) y Deep Learning (DL) aplicado a las bombas sumergibles eléctricas (ESP) que presenta un sistema que es capaz de estimar mediante la información generativa el Remaining Useful Life (RUL) e igualmente predecir/clasificar el tipo de fallo que puede presentar este tipo de equipos. Así mismo el autor elaboró el marco de dos etapas Integrado de Modos de Fallo con causa raíz (IFMRC) que relacionaba los modos de fallo con las causas raíz. Para ello se trabajó con 39 variables predictoras numéricas, (los parámetros: operacionales, eléctricos y de presión/flujo) que fueron procesadas en Python con el desarrollo Visual Studio Code.

Modelos de Regresión y Clasificadores se aplicaron a los datos; destacando el algoritmo Random Forest que con el objetivo de predecir el multi-clasificador de tipo de fallo se presentada con una exactitud (Accuracy) del 99,84 %, precisión (Precision) 0,999, recuperación (Recall) 0,999 y F1-Score de 1,000, evidenciando que este algoritmo alcanzó resultados casi perfectos en todas las categorías de prueba los que muestran la fiabilidad y la robustez del modelo que clasifica diez tipos posibles de fallos para las bombas sumergibles eléctricas [12].

1.3 Fundamentación teórica

1.3.1 Transformación Digital Industrial

La Transformación Digital Industrial (TDI) es un fenómeno de naturaleza muy complicada y polivalente, dado que se trata de la aplicación de tecnologías digitales a todo el negocio [13], [14]. Es un objetivo unificado basado en la mejora, optimización y digitalización de los procesos y sistemas existentes [15], utilizando las tecnologías digitales para transformar los modelos de negocio, generando nuevos valores y oportunidades [2]. La TDI busca la excelencia operativa y está muy centrada en la mejora del rendimiento, la seguridad, la reducción de costes y la eficacia [2], [14]. Esta transformación se extiende a la cultura organizacional, al liderazgo y a la capacidad de adaptarse a los cambiantes paisajes digitales [2]. En la Figura 1 se muestra un ejemplo de la industria 4.0 y sus componentes para el sector O&G.



Figura 1. La Industria 4.0 y sus componentes para el sector O&G [15].

a. La digitalización se sitúa dentro del marco de la Cuarta Revolución Industrial (Industria 4.0)

- La Cuarta Revolución Industrial ha promovido un gran número de los cambios que experimenta el O&G, todos ellos con el único enclavamiento de mejorar, optimizar y digitalizar los métodos y procesos existentes [13].
- El O&G es considerado el líder en la transición de la Industria 3.0 a la Industria 4.0 [13], [15].
- La transformación digital se considera un objetivo estratégico; permite optimizar la eficiencia, reducir el tiempo y acelerar la toma de decisiones [2], [15].

b. Las tecnologías capaces de establecer la Industria 4.0 (IoT, IA, Big Data) son las capacidades que promueven la TDI.

- Internet de las Cosas (IoT): El IoT es la capacidad de conectar dispositivos, sensores y máquinas, y permitir el intercambio de información a tiempo real [13].
- Análisis de Datos (Big Data Analytics): Este pilar es fundamental para extraer conocimientos a partir de la inmensa y compleja cantidad de datos que se generan en el O&G [13].
- Inteligencia Artificial (IA) y Aprendizaje Automático (ML): La IA es una tecnología muy relevante para lograr una mejor toma de decisiones a partir del soporte o la optimización de los métodos de trabajo de toda la cadena de valor [13].

c. La Transformación Digital para el Análisis Predictivo.

La Transformación Digital es la base indispensable para la implementación de un sistema IoT con capacidades de análisis predictivo, ya que trata de resolver los problemas culturales, económicos y de infraestructura que se tienen que resolver antes de que la tecnología pueda empezar a generar resultados positivos. El análisis

predictivo, que se basa en datos en tiempo real (IoT) y algoritmos (IA/ML), no tiene sentido sin un soporte organizacional y tecnológico [14].

1.3.2 Monitoreo inteligente y automatización

Dentro del mundo industrial el monitoreo inteligente utilizando IoT introduce nuevos estándares para la administración y el control de los diferentes sistemas: por ejemplo, sistemas de mantenimiento predictivo que bondadosamente utilizan sensores y algoritmos de inteligencia artificial para analizar los parámetros operativos y diagnosticar posibles errores [16], [17]. Los esquemas de mantenimiento predictivo que se basan en IoT están orientados a conseguir la menor cantidad de paradas de los sistemas mediante la identificación temprana de fallos en las máquinas de producción. Estos esquemas proporcionan una supervisión completa y en tiempo real, lo que permite la identificación rápida de potenciales fallos junto con una estrategia de mantenimiento que favorezca las paradas; todo ello se hace bajo la premisa de la reducción al mínimo de las interrupciones. Los esquemas utilizan algoritmos de machine learning, para buscar y detectar patrones complejos, predecir las tendencias futuras y generar detallados informes para los usuarios finales [16].

a. La Automatización mediante Flujos de Trabajo Inteligentes

La automatización es el elemento central del marco operativo, basado en flujos de trabajo clave y algoritmos avanzados:

- ***Flujos de Trabajo Automatizados.*** El sistema de gestión consigue una gestión eficiente en base a tres componentes centrales: el flujo de datos, el flujo de trabajo del monitoreo automatizado y el flujo de trabajo de análisis proactivo [17].
- ***Monitoreo Automatizado Jerárquico.*** El flujo de trabajo de monitoreo automatizado se basa en data en tiempo real y algoritmos avanzados para identificar y clasificar automáticamente los posibles problemas en el tráfico de producción [17].

- **Optimización automática.** el sistema permite la optimización automática de parámetros adaptándose a objetivos específicos y estos esquemas pueden ser transmitidos problemas a la plataforma de producción en tiempo real [17].

b. Monitoreo Inteligente y Predicción

El monitoreo inteligente se fundamenta mayormente en el aprendizaje automático y la IA para permitir que el sistema pase de ser reactivo a ser proactivo:

- **Diagnóstico proactivo.** Por medio del flujo de trabajo de análisis proactivo, el sistema no se limita a reaccionar ante los cambios actuales de las condiciones de las variables analizadas, sino que también los anticipa.
- **Machine learning y deep learning.** El sistema se basa en el uso combinado de técnicas de Machine Learning supervisado y métodos de Deep Learning para detectar patrones complejos en los datos operativos [17]. Usando variables eléctricas, hidráulicas y de proceso importantes, estas técnicas ayudan a anticipar problemas, como fallos o pérdidas de eficiencia, y a calcular cuánto tiempo más puede funcionar el equipo antes de necesitar un reemplazo [17].
- **Detección de anomalías complejas.** Se integran en el sistema algoritmos como el Análisis de Componentes Principales PCA y el Clustering para ser capaces de analizar las interacciones operativas multiparámetro, así como poder segregalas en clusters, así como identificar outliers que son indicativos de anomalías y fallos [17].
- **Mantenimiento predictivo.** En el dominio industrial, los sistemas basados en IoT se crean para funcionar en la predicción de daños, ya que permite la minimización del tiempo sin disponibilidad. Se obtienen los datos de parámetros operativos procedentes de dispositivos de captura como los sensores y su posterior resultante de una interpretación algorítmica para el monitoreo en tiempo real y la anticipación de fallos futuros [16].

1.3.3 Sistema IoT con análisis predictivo

La arquitectura IoT para el ámbito profesional y de la industria puede considerarse como una plataforma de tipo escalable, enormemente personalizable, esencial para el control centralizado y la supervisión de los elementos heterogéneos. La arquitectura consta de un conjunto de dispositivos físicos de red de comunicaciones, un gateway (puerta de enlace) y la plataforma IoT central. Esta plataforma IoT central, proveerá la gestión de los distintos dispositivos del sistema, el almacenamiento de datos, así como el aprovechamiento de los robustos elementos de seguridad y de analítica avanzada para la satisfacción de las exigencias de la industria [18].

La arquitectura en tres capas es el patrón más elemental y La arquitectura en cuatro capas supone una mejora del modelo simple de tres capas, frecuentemente en respuesta a la necesidad de resolver los problemas de seguridad y funcionalidad. En la Figura 2 se muestra el patrón arquitectural de 3 capas y el de 4 capas:

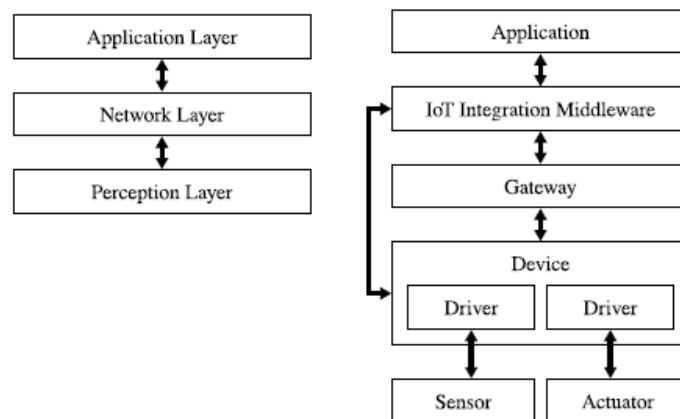


Figura 2. Arquitectura IoT de 3 y 4 capas [18].

- **Capa de Percepción (Perception Layer).** Corresponde a la capa inferior de la arquitectura de tres capas que también se conoce como Capa de Sensor, se corresponde en hacer la recolección de información del mundo físico [19].
- **Capa de Red (Network Layer).** Su función es la de actuar como un puente para hacer pasar la información capturada en la Capa de Percepción a la Capa de Aplicación. Es la capa que proporciona la conexión a los dispositivos inteligentes, los dispositivos de red y las redes [19].

- **Capa de Aplicación (Application Layer).** Es la capa superior que da lugar a todas las aplicaciones que emplean la tecnología IoT. Su misión es proporcionar servicios específicos a estas aplicaciones, teniendo los mismos como punto de partida la información que ha sido recolectada por los sensores [19].
- **Capa de Soporte (Support Layer).** Esta es la capa de soporte que se ha añadido como consecuencia de la falta de seguridad de la arquitectura en tres capas. La información extraída de la Capa de Percepción es enviada aquí [19].

a. Protocolos de comunicación.

Los protocolos de comunicación en el marco del Internet de las Cosas (IoT) son el conjunto de normas y estructuras que hacen posible la transmisión de información entre dispositivos, sensores, gateways y sistemas centralizados. Su pluralidad resulta fundamental ya que permite la adaptación flexible a las distintas exigencias de cada contexto, no sólo permite integrar una vasta y variada gama de dispositivos y aplicaciones [16]. En la siguiente sección se describen los Protocolos Inalámbricos, esenciales para los sistemas de comunicación modernos.

- **Wi-Fi.** Es uno de los protocolos que más se utilizan, dado a su facilidad de implementación y su gran capacidad de rendimiento. Esto permite crear conexiones estables y rápidas, ya que es un protocolo ideal en iniciativas que requieren la transmisión de datos en tiempo real [16].
- **Bluetooth.** Protocolo inalámbrico que permite conectar sensores, dispositivos móviles, equipos de monitoreo y máquinas, facilitando el intercambio de información [16].
- **Zigbee.** Protocolo inalámbrico que se utiliza en redes de sensores, dado su bajo consumo de energía y su capacidad de establecer grandes redes de malla ([16].
- **GSM.** Protocolo que permite la conexión de dispositivos IoT y redes móviles para monitorizar y controlar los dispositivos a distancia incluso sin redes Wi-Fi [16].

Los protocolos por cable constituyen tecnologías fundamentales de comunicación industrial. En este sentido, se presentan a continuación los principales protocolos aplicados en entornos por cable.

- **Modbus.** Protocolo que trabaja con la organización maestro-esclavo para la comunicación con múltiples nodos distribuidos. Se usa fundamentalmente en su versión sobre RS-485, lo que permite conexiones seguras a distancias largas y en condiciones electromagnéticas severas.
- **CAN (Controller Area Network).** Protocolo de comunicación serial permite una comunicación distribuida sin necesidad de un nodo maestro.
- **Ethernet.** Protocolo que permite la comunicación entre microcontroladores y la conexión de red a través de programas o envío de datos para acceso remoto [16].

Los Protocolos LPWAN se han consolidado como soluciones clave en aplicaciones IoT que requieren amplia cobertura y larga autonomía energética. En las siguientes líneas se describen sus principales variantes.

- **LoRa (Long Range).** Protocolo de comunicación sin hilos que permite transmitir datos a largas distancias de la manera menos costosa energéticamente. Esta característica lo hace idóneo para el control de dispositivos en superficies grandes como campos o explotaciones. Sus características hacen que sea capaz de comunicarse en los lugares donde no existe cobertura GSM [16].
- **Sigfox. Tecnología LPWAN** para transmitir pequeñas cantidades de datos a largas distancias y de manera muy energética. Opera en bandas de frecuencia sin licencia y usa modulación Ultra Narrow Band [16].
- **NB-IoT (Narrowband IoT).** Tecnología LPWAN delineada con los estándares 3GPP. Opera en bandas de frecuencia con licencias. Emplea canales de banda estrecha (200 kHz), con lo que permite una alta penetración de señal en el interior de los edificios y permite soportar muchos dispositivos en áreas muy pequeñas[16].

b. Estándares

Los estándares de Internet de las Cosas (IoT) establecidos para el sector de petróleo y gas son un aspecto fundamental que promueve la correcta, efectiva, segura y confiable recolección de las medidas operativas que se realizan mediante Redes de Sensores Inalámbricos. Estos lineamientos, por su parte, están fundamentados en normas internacionales que establecen los requisitos técnicos reguladores de la forma de operar, del correcto funcionamiento, de la interoperabilidad y de la resiliencia de los protocolos de comunicación, mostrando su estructura en una jerarquía de trabajo por capas.

- ***ZigBee PRO***. Es un estándar internacional que consiste en uno de los logros más recientes de la tecnología inalámbrica en el seno de la implementación de redes de sensores inalámbricos (WSN) fiables en la industria. Este marco normativo se utiliza sólo para dar una descripción concisa y en capas de la estructura del protocolo de red dentro de la primera capa (sensor) de la arquitectura IoT [20].
- ***Wireless HART***. Constituye un estándar internacional de redes de sensores inalámbricos (WSN), que es básico para gestionar plataformas y optimizar las operaciones en petróleo y gas. Este estándar proporciona los requisitos clave que permiten proporcionar una descripción organizada y jerárquica (por capas) de los protocolos de comunicación de la red de sensores [20].
- ***ISA100.11***. Es un estándar internacional básico de WSN considerado y mencionado como uno de los grupos de requisitos clave usados para la descripción de la estructura del protocolo de red. Los dispositivos inalámbricos utilizados en la industria se basan en estándares como ISA100.11a que proporcionan la interoperabilidad y la disponibilidad operacional de la red operativa en entornos duros [20]. La comparativa de los principales estándares se muestran en Tabla 1.

Tabla 1. Estándares IoT de industria petrolera.

Característica	Zigbee PRO	WirelessHART	ISA100.11a
Organización que lo define	Zigbee Alliance / Connectivity Standards Alliance (CSA)	HART Communication Foundation	International Society of Automation (ISA)
Tipo de estándar	Estándar de red WSN basado en IEEE 802.15.4	Estándar industrial WSN basado en IEEE 802.15.4	Estándar industrial WSN basado en IEEE 802.15.4
Capa del modelo IoT	Percepción / Red (sensores IoT)	Percepción / Red (instrumentación industrial)	Percepción / Red (instrumentación en entornos severos)
Topología	Malla, árbol, estrella	Malla autoreconfigurable	Malla flexible con alta confiabilidad
Frecuencia de operación	Sub-GHz / 2.4 GHz	2.4 GHz	2.4 GHz
Seguridad	Cifrado AES-128	AES-128 + autenticación	AES-128 + certificación y perfiles seguros
Interoperabilidad	Alta entre dispositivos Zigbee	Muy alta en instrumentación industrial certificada	Muy alta, diseñada para entornos severos

c. Normativa.

El régimen de comunicaciones aplicable a los sistemas industriales de Internet de las Cosas (IoT) se sustenta en el carácter estratégico de las telecomunicaciones y del espectro radioeléctrico, cuya administración, regulación y control corresponden a la Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL). En este contexto, los dispositivos que utilizan espectro y se conectan a redes públicas deben pasar por procesos de homologación y certificación. Esto es para asegurar que funcionen correctamente y para evitar interferencias dañinas [21]. En el sector hidrocarburífero, los Sujetos de Control están obligados a suministrar información operativa en tiempo real y a asegurar la compatibilidad de aplicaciones y protocolos de comunicación con el sistema SCADA de la entidad reguladora. De manera complementaria, la rectoría ejercida por el ministerio competente promueve la masificación de las Tecnologías de la Información y Comunicación (TIC), el IoT y las plataformas de visualización como herramientas para el desarrollo económico y social. Esto está en coherencia con la Política Nacional de Ciberseguridad (PNC) y el Plan Nacional de Desarrollo [22]. La seguridad de los sistemas IoT se vincula directamente con los derechos constitucionales al secreto e inviolabilidad de las comunicaciones y a la protección de los datos personales. Esto obliga a los prestadores de servicios a implementar medidas técnicas y de gestión robustas para garantizar la integridad de la

red y la confidencialidad de la información. Finalmente, la PNC prioriza la protección de la Infraestructura Crítica Digital (ICD) y de los servicios esenciales que dependen de la TO y las TIC. Esto se debe a que su vulneración o interrupción podría generar impactos severos en la continuidad de las actividades productivas y en la estabilidad de la sociedad [23]. En la Tabla 2 se muestra la Alineación del servicio de Conectividad IoT con la industria de hidrocarburos en el Ecuador del informe técnico No. IT-CRDS-GR-2022-046.

Tabla 2 Alineación del servicio de Conectividad IoT con la industria de hidrocarburos en el Ecuador [24].

Elemento normativo	Aporte a la implementación de IoT y aplicación en la industria del petróleo
Sección 3.1 – Base legal: Constitución de la República (arts. 226, 227, 261, 313 y 314)	Reconoce a las telecomunicaciones, el espectro y los hidrocarburos como sectores estratégicos, por lo que la conectividad IoT en petróleo se considera un servicio regulado y alineado a la política energética.
Sección 3.2 – Base legal: Ley Orgánica de Telecomunicaciones (arts. 3, 4, 35 y 36)	Establece principios como la neutralidad tecnológica y habilita el despliegue de infraestructura para transmitir datos, respaldando legalmente redes IoT para procesos petroleros.
Sección 3.3 – Base legal: Reglamento General y ROTH (arts. 18, 20, 37, 46, 47, 204 y 211)	Define títulos habilitantes, tarifas y responsabilidades, determinando quién puede ofrecer conectividad IoT y bajo qué condiciones administrativas y técnicas.
Sección 4.1 – Análisis del régimen de prestación de servicios LOT arts. 3 y 4	Ordena adecuar la normativa e incorporar el servicio de Conectividad IoT mediante una ficha técnica, estableciendo requisitos formales para proyectos industriales.
Sección 4.2 – Ámbito de la reforma, mandato de la Resolución 02-05-ARCOTEL-2022	Aporta una definición estándar de IoT y su ecosistema, clarificando roles entre operadores.
Sección 4.3–4.4 – Definición del servicio y ecosistema IoT (Recomendación UIT-T Y.2060)	Describe el uso de bandas libres UDBL y tecnologías LPWAN, facilitando redes de largo alcance en campos remotos y reduciendo dependencia de espectro licenciado.
Sección 4.4–4.5 – Referencias a tecnologías LPWAN (LoRaWAN, Sigfox) y Plan Nacional de Frecuencias (Res. 04-02-ARCOTEL-2021, bandas UDBL)	Formaliza el servicio de Conectividad IoT y define modalidades de habilitación (autorización/registro), dando claridad para operar o contratar conectividad IoT.
Sección 5.1.1 – Propuesta de reforma al ROTH (art. 13 y siguientes)	Establece el contenido mínimo del proyecto técnico IoT, exigiendo documentar arquitectura, enlaces, espectro y ubicación de nodos.
Sección 5.1.1 – Contenido mínimo del proyecto técnico	Define el alcance geográfico del servicio (nacional, regional o cantonal), permitiendo planificar redes IoT desde zonas puntuales hasta cobertura extendida.
Sección 5.1.1 – Tipo de título habilitante y área geográfica (art. 37 ROTH y análisis del informe)	Determina que el servicio IoT se otorgará como Autorización a empresas públicas y Registro de servicios a operadores privados, y que el área geográfica puede ser Nacional, Regional o Cantonal.
Sección 5.1.1 – Derechos y tarifas (art. 46 ROTH y remisión a normativa vigente)	Determina derechos y tarifas por habilitación y uso de frecuencias, influyendo en los costos cuando se requiere espectro licenciado o enlaces dedicados.

1.3.4 Algoritmos de Machine Learning

El Machine Learning es una rama de la inteligencia artificial que se enfoca en desarrollar algoritmos y modelos que permiten a las computadoras aprender de los datos. Estos sistemas hacen predicciones y resuelven problemas sin ser programados explícitamente para cada tarea. Se le define como la ciencia de crear software o modelos autónomos que aprenden de los datos y mejoran automáticamente con la experiencia. El proceso implica utilizar datos y algoritmos para entrenar a las computadoras de manera que imiten los procesos de aprendizaje humano. Esto incluye adquirir conocimiento, entender el proceso y aplicar ese conocimiento para el razonamiento y la toma de decisiones fiables [25].

a. Tipos de aprendizaje.

En machine learning gira en torno a los principales métodos de aprendizaje, agrupados sobre todo en cuatro tipos de algoritmos: Aprendizaje Supervisado, Aprendizaje No Supervisado, Aprendizaje Semisupervisado y Aprendizaje por Refuerzo. Estos algoritmos constituyen el bloque básico para que el modelo pueda procesar datos y encontrar patrones. Este enfoque permite construir soluciones prácticas en el mundo real en la medida en que las computadoras pueden aprender de los datos sin que tengan que ser programadas para ello [25].

- *Aprendizaje Supervisado.*

El Aprendizaje Supervisado es la metodología esencial del machine learning, en la que los modelos se entrenan a partir de datos etiquetados que vinculan cada instancia de entrada con un valor objetivo asociado. Este enfoque permite a los algoritmos aprender sistemáticamente la relación entre las características de la entrada y sus etiquetas, y computar en la generalización una clasificación o la predicción de los valores para los nuevos datos. En contraste con los otros tipos de aprendizaje sin guía explícita, el aprendizaje supervisado se puede definir como un proceso más parecido a seguir la instrucción de un formador, que garantiza una mayor precisión en la identificación de patrones y una mejor reducción de la incertidumbre en los procedimientos analíticos. Como consecuencia, se ha constituido como una de las técnicas más eficaces cuando

se dispone de conjuntos de datos etiquetados y, con frecuencia, se utiliza para problemas de clasificación y regresión debido a su capacidad para modelar relaciones complejas de manera estructurada.[25].

- ***Aprendizaje No Supervisado***

El Aprendizaje No Supervisado constituyéndose como una técnica de machine learning importante en la que el entrenamiento de los algoritmos no requiere datos etiquetados a priori. En este sentido, se centra en el hecho de que las máquinas extraerán patrones, formas latentes y relaciones internas de los datos; tiene una especial utilidad en ámbitos en los que etiquetar es complicado o requiere la participación de una persona que haga etiquetados. Por otra parte, en el aprendizaje no supervisado podemos encontrar características que posteriormente sirven para la clasificación o el análisis donde el aprendizaje supervisado tiene lugar. Las técnicas más representativas del Aprendizaje No Supervisado están constituidas por el clustering o agrupamiento y las reglas de asociación, que son técnicas que se basan en racionalizar la información según grupos más o menos coherentes y/o exposiciones de las relaciones frecuentes entre variables [25].

- ***Aprendizaje Semi supervisado***

El Aprendizaje Semi supervisado puede ser determinado, de manera más general, como una estrategia de aprendizaje intermedia situada entre el aprendizaje supervisado y el no supervisado que combina datos etiquetados, con un mayor número de ejemplos no etiquetados para mejorar el aprendizaje del modelo, siendo especialmente importante en los casos donde el proceso de etiquetado es caro, requiere amplios niveles de experiencia técnica o requiere tiempos largos asociados a especialistas del dominio. Dando un pequeño set de ejemplos con etiquetas conocidas, el algoritmo es capaz de inferir patrones a partir de datos no etiquetados y, así, mejorar su rendimiento de generalización. Las técnicas más empleadas en este tipo son el auto entrenamiento (self-training), el co-entrenamiento (co-training) o varios modelos generativos, que permiten un buen aprovechamiento de la estructura interna de los datos [25].

- ***Aprendizaje de Refuerzo***

El Aprendizaje por Refuerzo se concibe como una metodología avanzada de machine learning en la que un modelo es entrenado para tomar secuencias de decisiones dentro de entornos complejos y dinámicos. En este enfoque, el agente percibe e interpreta de manera continua las condiciones del entorno, y mediante un proceso iterativo de prueba y error identifica las acciones que conducen a resultados más favorables. El mecanismo central de aprendizaje se basa en un sistema de recompensas y penalizaciones: las acciones consideradas deseables generan retroalimentación positiva que refuerza su repetición, mientras que los comportamientos no adecuados reciben penalizaciones que ajustan la estrategia del agente en función del desempeño observado. Como consecuencia, el modelo desarrolla políticas que maximizan la recompensa acumulada a lo largo del tiempo, consolidando este enfoque como una herramienta clave para la optimización de procesos secuenciales y la toma autónoma de decisiones en escenarios inciertos [25].

- b. Modelos de aprendizaje***

El aprendizaje estadístico engloba una serie de herramientas clave para la modelización y la predicción, en la que se implementan enfoques supervisados y no supervisados a fin de dar sentido a datasets complejos.

- ***Random Forest***

Definido como un método de ensamble destinado a aumentar la precisión que se logra aplicando el bagging, el Random Forest añade la correlación sistemática de los árboles cuando se entrenan. Consiste así en generar muchos árboles de decisión teniendo como entrada ejemplares que provienen de muestras bootstrapped del conjunto de datos original, introduciendo de este modo variabilidad en la estructura de cada uno de los árboles y contribuyendo a la reducción del sobreajuste. Para conseguir la correlación, cada vez que se toman decisiones en cada nodo solo se consideran un conjunto aleatorio de predictores en el momento de llevar a cabo la división, evitando que distintos árboles utilicen siempre los mismos predictores. Promediando los resultados de una colección de árboles débilmente correlacionados, se produce una notable reducción de la varianza de la predicción final, contribuyendo a que produzca en gran

medida una mejora de la estabilidad y del desempeño del modelo [26]. Puede representarse del modo siguiente:

$$\hat{f}(x) = \frac{1}{B} \sum_{b=1}^B \hat{f}^b(x) \quad (1)$$

Dentro del esquema del Random Forest, el parámetro B hace referencia al número de árboles de decisión que constituyen el bosque, y la expresión $\hat{f}^b(x)$ representa la predicción generada por el b -ésimo árbol en particular del modelo. Cada uno de esos árboles se construye a partir de una muestra bootstrapped del conjunto de entrenamiento, lo que introduce variabilidad en la estructura de los árboles generados, que se ve acompañada con la aleatorización de un subconjunto de predictores en cada división, mecanismo crucial para garantizar que los árboles del modelo estén correlacionados; de este modo, el promedio de cada predicción individual permite disminuir el error considerablemente, logrando así que este procedimiento se convierta en una estrategia sólida y efectiva para las tareas de clasificación y regresión [26].

- ***Gradient Boosting***

El boosting se considera a la vez una técnica de ensemble que incorpora precisamente un modelo con el objetivo de mejorar la precisión de los árboles de decisión a través de un procedimiento de construcción secuencial, a diferencia de métodos como el bagging que son paralelos [i, 345]. Así, cada nuevo árbol se ajustará utilizando la información residual generada por el modelo ya construido de manera que el aprendizaje se realizará para corregir los errores cometidos en el modelo acumulado durante las iteraciones previas. La construcción del modelo es un procedimiento en el que se favorece un aprendizaje lento, lo que permite captar los patrones complejos en un número alto de ciclos de ajuste. Debido a que se produce un ajuste iterativo es necesario aplicar la correcta sintonización de los hiperparámetros, que son determinantes para regular la precisión y evitar el overfitting [26]. Se expresa de la manera que se detalla a continuación:

$$\hat{f}(x) = \sum_{b=1}^B \lambda \hat{f}^b(x) \quad (2)$$

En métodos de boosting, la expresión $\hat{f}(x)$ representa la predicción que hace el modelo para una observación de tipo x , la cual se obtiene al combinar, de forma secuencial, el aporte de todos los árboles construidos durante el entrenamiento. Cada uno de los componentes se denota por $\hat{f}^b(x)$, y corresponde a la predicción que realiza el b -ésimo árbol independiente que se genera y cuyo objetivo radica en modelar y corregir los residuos entre la predicción del árbol previamente ajustado. La combinación de los árboles está regulada por un parámetro de reducción λ (donde $\lambda \geq 0$) el cual controla qué tan influenciado queda el modelo acumulado, por cada árbol [26]. El algoritmo de Gradient Boosting pertenece a la familia de métodos de ensamble basados en árboles de decisión y cuenta con variantes avanzadas, entre las que destacan a continuación.

XGBoost. Incorpora la técnica Gradient Boosting Decision Tree (GBDT), la cual ensambla múltiples árboles de decisión, para crear un modelo conjunto robusto y de alta precisión predictiva. Su funcionamiento se basa en la actualización incremental de los pesos del modelo durante el entrenamiento, utilizando el gradiente de la función de pérdida para optimizar su rendimiento. Una característica clave es su estrategia de crecimiento de árbol nivel por nivel (level-wise), que, aunque intensiva en cómputo, mejora la estabilidad y la exactitud del modelo resultante [27].

LightGBM. Es un framework de gradient boosting desarrollado por Microsoft, conocido por su eficiencia y velocidad superior en la construcción de modelos de machine learning precisos. Incorpora un algoritmo basado en histogramas que discretiza los valores continuos en bins, lo cual resulta crucial para acelerar el entrenamiento y reducir el uso de memoria. El modelo es flexible, ya que soporta dos estrategias de crecimiento de árbol: level-wise y leaf-wise. La estrategia leaf-wise se centra en la división con la mayor ganancia, pero requiere una limitación de profundidad para evitar el sobreajuste [27].

CatBoost. Este modelo es distinguido por su capacidad sobresaliente para manejar características categóricas. Su diseño elimina la necesidad de preprocesamiento o normalización explícita de estas características, facilitando la captura de interacciones entre ellas. El modelo utiliza árboles de decisión oblivious que aseguran una estructura simétrica en cada nivel de profundidad, lo cual actúa como una forma de regularización y mejora la consistencia. Además, CatBoost implementa el mecanismo de ordered boosting para prevenir la fuga de información del objetivo (target leakage), calculando los valores de transformación solo a partir de datos vistos previamente [27]. La Tabla 3 muestra la comparación de modelos de entrenamiento Random Forest, XGBoost, LightGBM y CatBoost.

Tabla 3. Comparación de modelos de entrenamiento Random Forest, XGBoost, LightGBM y CatBoost [26], [27].

Aspecto	Random Forest	XGBoost	LightGBM	CatBoost
Método de ensamble	Bagging (bosque de árboles)	Gradient Boosting	Gradient Boosting (histogramas)	Gradient Boosting (árboles oblivious)
Base learner	Árboles CART independientes	Árboles CART + regularización	Árboles CART con bins/histogramas	Árboles CART oblivious (simétricos)
Pérdida típica	Impureza (Gini/Entropía) por árbol	Log-loss / multiclass (según objetivo)	Log-loss / cross-entropy	Log-loss / cross-entropy / multiclass
Salida del modelo	Voto mayoritario / promedio	Suma de scores + sigmoid/softmax	Suma de scores + sigmoid/softmax	Suma de scores + sigmoid/softmax
Tamaño del ensamble	n_estimators	num_boost_round / n_estimators	num_iterations + num_leaves	iterations
Control de complejidad	max_depth, min_samples_leaf	max_depth, min_child_weight, max_leaves	num_leaves, min_data_in_leaf	depth, l2_leaf_reg
Tasa de aprendizaje	No aplica (sin learning rate)	learning_rate	learning_rate	learning_rate
Muestreo de instancias	Bootstrap (bootstrap, max_samples)	subsample	bagging_fraction, bagging_freq	subsample + ordered boosting (permutaciones)
Muestreo de variables	max_features	colsample_*	feature_fraction, feature_fraction_bynode	rsm (subespacio aleatorio)
Regularización	Implícita (profundidad/hojas)	L1/L2 (alpha, lambda) + gamma	lambda_l1, lambda_l2, min_gain_to_split	l2_leaf_reg + penalizaciones internas
Crecimiento del árbol	Independiente por árbol (depth/best-first)	Mayormente level-wise	Principalmente leaf-wise (más agresivo)	Oblivious: misma secuencia de splits en todas las rutas
Desbalance de clases	class_weight / re-muestreo	scale_pos_weight	is_unbalance, scale_pos_weight	class_weights, auto_class_weights

c. Deep learning

Es un subcampo de machine learning que hace referencia a las arquitecturas de aprendizaje, basadas en el uso de arquitecturas neuronales formadas por múltiples capas que permiten modelar patrones de creciente complejidad desde datos crudos. Las arquitecturas de estos modelos son las convencionales en este campo, en donde existen una capa de entrada, múltiples capas y, finalmente, una capa de salida. Estos modelos toman también en consideración funciones de activación no lineales, para poder modelar relaciones complejas entre las variables [28]. Para lo cual tenemos los siguientes ejemplos de modelos Deep learning.

- ***Redes Neuronales Convolucionales (CNN)***

Las CNNs son arquitecturas de aprendizaje profundo altamente eficientes que se utilizan para extraer características ricas de los datos de entrada, como imágenes, a través de dos funciones críticas: convolución y pooling. El submuestreo (subsampling), o pooling, resume las características salientes en el espacio de características, reduciendo la dimensión de los datos [29]. La cantidad de parámetros entrenables (n_1) en una capa convolucional unidimensional calcula de la siguiente manera:

$$n_1 = (k * d + 1) * f \quad (3)$$

En donde k es el tamaño del núcleo (kernel size), d es la dimensión del espacio de características de la capa previa, f y es la dimensión del espacio de características de la capa actual.

- ***Redes de Memoria a Largo Corto Plazo (LSTM)***

Las LSTMs son una forma adaptada de Redes Neuronales Recurrentes (RNN), diseñadas para interpretar y pronosticar datos secuenciales, como series de tiempo numéricas o texto. Su arquitectura permite memorizar información de estados pasados en celdas de memoria especializadas, conocidas como compuertas (gates). Utilizando la información de estas compuertas, se calculan los valores predichos de la variable

objetivo [29]. El número de parámetros (n_1) en cada una de las cuatro compuertas de una red LSTM (input, forget, input modulation y output gate) se calcula así:

$$n_1 = (x + y) * x + x \quad (4)$$

El número total de parámetros en una capa LSTM es $4 * n_1$. En donde x denota el número de unidades LSTM (nodos) e y es la dimensión de entrada.

d. Criterios de Evaluación

La evaluación del modelo es la fase más importante dentro de la práctica del aprendizaje automático, cuyo significado es el de evaluar las puntuaciones y el funcionamiento del modelo entrenado aplicándolo a datos no vistos del conjunto test. De esta forma, se puede evaluar la capacidad del modelo para generalizar más allá del entrenamiento, su fiabilidad o utilidad en aplicaciones para las que se espera un funcionamiento correcto y reproducible.

- ***Matriz de confusión***

La matriz de confusión es una de las herramientas más importantes que se usan para evaluar el rendimiento de un modelo de clasificación al poder comparar de la forma más sistemática posible las predicciones generadas con las etiquetas reales del conjunto de datos. Estructurada en forma de tabla, dicha matriz permite agrupar los datos considerados en cuatro grupos diferentes, lo que ayuda a encontrar “aciertos”, “errores” o las pautas de clasificación más relevantes que ayudarán a poder evaluar con mayor detalle el funcionamiento del modelo [25]. En la Tabla 4 se muestra la estructura de la matriz de confusión de clasificación.

Tabla 4. Estructura de la matriz de confusión de clasificación.

MODEL PREDICTION	ACTUAL POSITIVE	ACTUAL NEGATIVE
Predicted positive	True Positive (TP)	False Positive (FP)
Predicted negative	False Negative (FN)	True Negative (TN)

- ***Verdaderos Positivos (TP)***

Corresponden a los casos en los que el modelo predice que una instancia concreta es positiva y de verdad esa instancia pertenece a la clase positiva. Son los casos en los que una clasificación ha sido correcta y se ha identificado un suceso positivo [25].

- ***Verdaderos Negativos (TN)***

Este concepto hace referencia a aquellas instancias que el modelo predice como negativas, coincidiendo con su etiqueta real. Por lo tanto, puede reflejar los aciertos en la detección de eventos negativos [25].

- ***Falsos Positivos (FP)***

Esto se produce cuando el modelo predice de forma errónea que un caso es positivo, a pesar de que su etiqueta real es negativa. Este error es conocido como Error Tipo I, el cual se correspondería con una falsa alarma dentro del proceso de clasificación [25].

- ***Falsos Negativos (FN)***

Esto sucede cuando el modelo predice incorrectamente que una instancia es negativa, y en realidad es positiva. Este error o tipo es conocido como Error Tipo II, el cual se correspondería con la no detección del modelo de un caso que pertenece a la clase positiva [25].

El tipo de clasificación multiclase se utiliza cuando a una instancia se le asignan una de tres o más categorías posibles teniendo como consecuencia que se generan límites de decisión más complejos. Este modelo resulta ampliamente utilizado en áreas donde el fallo en la correcta predicción de clases tiene un coste significativo [30]. Su dificultad radica, en gran medida, en la presencia de superposición entre categorías y en el frecuente desequilibrio de datos, factores que pueden deteriorar de forma significativa el rendimiento del modelo. Las evaluaciones de la clasificación multiclase se realizan con la matriz de confusión, cuya diagonal principal engloba el número de las predicciones correctas asociadas a los verdaderos positivos de cada una de las clases, permitiendo esto la obtención de las métricas de exactitud o precisión. Sin embargo, cuando la distribución de las clases está desequilibrada, se sigue

recomendando el uso de medidas más robustas, como el F1-score o la exhaustividad (recall), ya que resultan más adecuadas para inferir el rendimiento [31], [32]. En la Figura 3 se muestra la matriz de confusión de tipo clasificación multiclase.

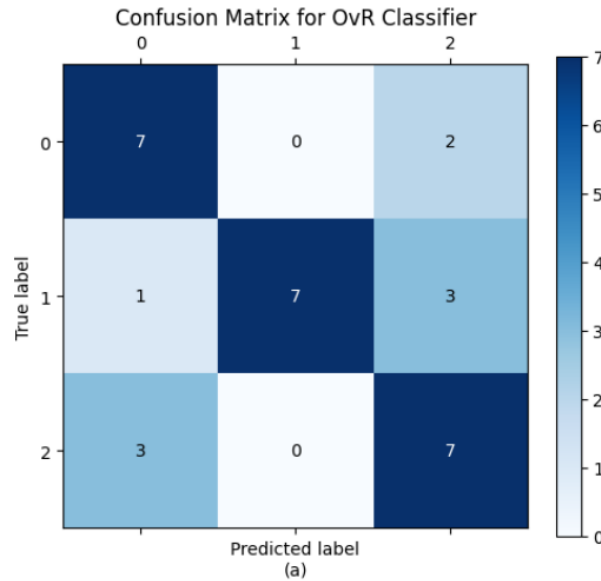


Figura 3. Matriz de confusión de tipo clasificación multiclase [30].

e. Métricas de Rendimiento para Clasificación

Las métricas de evaluación del rendimiento permiten cuantificar, mediante valores numéricos, distintas características del desempeño de un modelo de clasificación. Las mediciones en cuestión se obtienen de los valores que surgen de la matriz de confusión, y son medidas que nos proporcionan información suficiente acerca de la precisión, la capacidad de detección y el equilibrio entre aciertos y errores del modelo en condiciones reales de evaluación [25].

- **Accuracy**

La exactitud (accuracy) es el cociente entre el número de instancias clasificadas con acierto de las instancias totales evaluadas. Representa, por tanto, una métrica que permite tener una idea del comportamiento el modelo de clasificación, dado que recoge tanto los aciertos en la clasificación de positivos como los aciertos en la clasificación de negativos. Un valor alto de accuracy sugiere que el modelo de clasificación puede realizar con alta probabilidad las predicciones correctas para el conjunto de datos que se ha analizado [25].

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \quad (5)$$

- **Precisión**

Cuantifica la cantidad de instancias positivas detectadas (TP) entre todas aquellas que el clasificador ha etiquetado como positivas (TP + FP). Un valor elevado de precision puede interpretarse como una baja tasa de falsos positivos, ofreciendo menos posibilidades de error en las predicciones positivas emitidas [25].

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

- **Recall**

Evalúa la proporción de instancias realmente positivas detectadas por el modelo (TP) con respecto al total de las instancias potencialmente positivas (TP + FN). Un alto valor de recall puede deducirse como una buena capacidad del modelo para detectar la mayoría de los casos positivos dados en el dataset [25].

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

- **F1 Score**

Se refiere a la media entre la precisión y el recall, permitiendo ofrecer un valor justo del rendimiento del modelo, dado que se tiene en cuenta la capacidad de evitar falsos positivos junto con la de no tener falsos negativos. Esta métrica es de gran utilidad cuando los conjuntos de datos están desbalanceados, puesto que no deja que la evaluación se reme hacia la clase mayoritaria. Un elevado F1 Score puede interpretarse como que se ofrece una buena combinación entre precisión y recall [25].

$$FScore = \frac{Precision * Recall}{Precision + Recall} \quad (8)$$

- **Macro average**

Calcula el promedio aritmético simple de la métrica entre todas las clases, asignando el mismo peso a cada clase, independientemente del número de muestras que contenga. Es útil para evaluar si el modelo se comporta de forma equilibrada en clases frecuentes y poco frecuentes [25].

$$M_{macro} = \frac{1}{K} \sum_{i=1}^K M_i \quad (9)$$

En donde, K el número de clases y M_i la métrica de interés calculada para la clase i .

- **Weighted average**

Es el promedio ponderado de la métrica, donde el peso de cada clase es proporcional al número de muestras que posee. De este modo, las clases más frecuentes tienen mayor influencia en el valor final, proporcionando una medida más representativa del rendimiento global del modelo sobre el conjunto de datos desbalanceado [25].

$$M_w = \sum_{i=1}^K \frac{n_i}{N} M_i \quad (10)$$

Se define donde, n_i es número de muestras de la clase i y $N = \sum_{i=1}^K n_i$ el total de muestras y M_i es la métrica por clase.

1.3.5 Análisis y visualización de datos en tiempo real

En el mundo del análisis de datos industriales, las plataformas de visualización son fundamentales para darle sentido a los datos masivos generados por sistemas IoT, infraestructuras distribuidas y arquitecturas de monitoreo en tiempo real. Estas herramientas pueden convertir datos de series de tiempo, logs y métricas de rendimiento en visualizaciones que ayuden a entender cómo se comporta el sistema y tomar mejores decisiones.

a. Grafana

Grafana es una herramienta de visualización de datos enfocada en métricas y series temporales de sistemas informáticos, infraestructuras IoT o servicios distribuidos. Su arquitectura está basada en un modelo cliente-servidor con un motor gráfico, sistema de paneles personalizables y plugins para extender su compatibilidad con diversas fuentes de datos, desde bases de datos especializadas como InfluxDB, Prometheus o Elasticsearch. Los dashboards son el corazón de la herramienta, capaces de visualizar datos numéricos, históricos y patrones operativos en forma de gráficos interactivos, mediciones en tiempo real y paneles analíticos avanzados. El sistema de alertas automatizadas se configura con base en reglas, umbrales y chequeos constantes para informar anomalías operacionales por medio de Telegram, correo electrónico o webhooks [33]. En la Figura 4 se presenta la arquitectura general de Grafana.

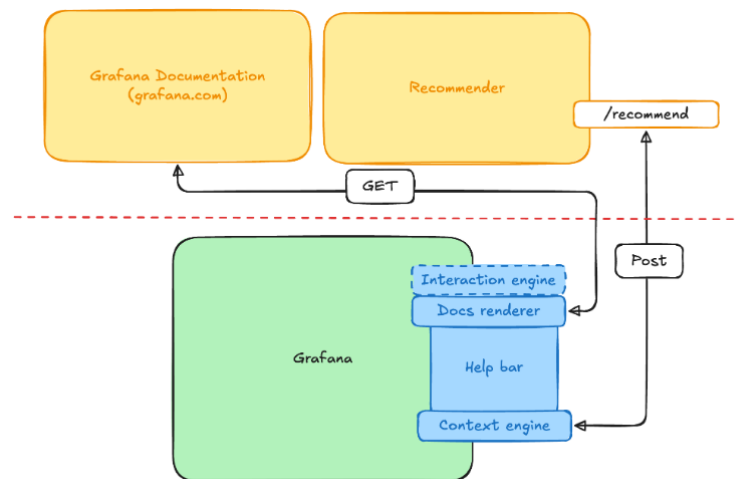


Figura 4. Arquitectura general de Grafana [33].

b. Kibana

Kibana es la interfaz de visualización del stack ELK para analizar, explorar y visualizar datos almacenados en Elasticsearch. Su objetivo es ofrecer formas interactivas de explorar registros, métricas de sistemas y eventos de aplicaciones distribuidas. A nivel funcional, Kibana incorpora herramientas de visualización enriquecidas, como gráficos de líneas, histogramas, mapas de calor y paneles de control, para descubrir tendencias, patrones y anomalías en tiempo real. El módulo de dashboarding permite crear paneles con múltiples consultas y filtros para realizar análisis comparativos y de

comportamiento en el tiempo. Gracias a su capacidad de analizar registros (logs) hace posible procesar millones de eventos de servidores, redes o dispositivos IoT, con búsquedas de alta precisión, filtrado por campos y correlación entre métricas operativas. En el diseño del stack ELK, Kibana es la capa visual que acompaña la ingesta de datos de Logstash y el almacenamiento indexado de Elasticsearch, garantizando un flujo para la interpretación en sistemas críticos [34]. En la Figura 5 se describe la arquitectura de Kibana.

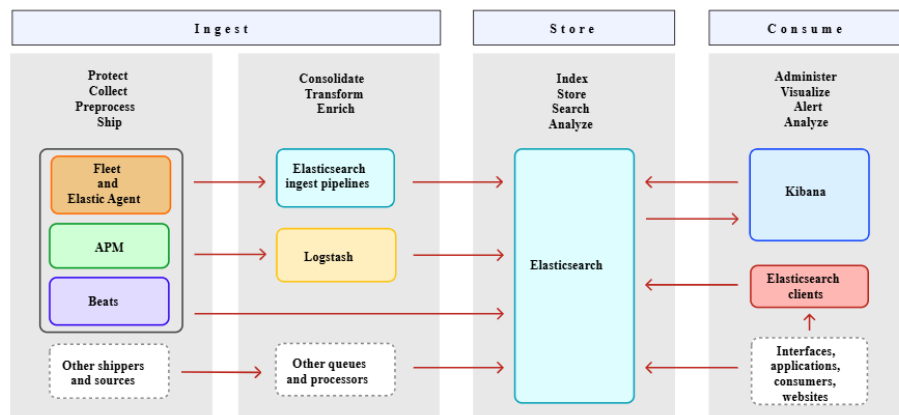


Figura 5. Arquitectura de Kibana [34].

c. *Power Bi*

Es una plataforma integral de inteligencia permite recopilar, transformar y visualizar datos de diversas bases de datos. Su arquitectura se basa en Power BI Desktop para modelado semántico, limpieza y creación de informes; Power BI Service en la nube para publicación, distribución y gobernanza; y Power BI Mobile para consumo en tiempo real. Además, la plataforma emplea un motor analítico in-memory que optimiza la compresión columnar y acelera la ejecución de consultas complejas, mejorando la eficiencia computacional en situaciones con grandes conjuntos de datos. Power BI permite crear modelos semánticos reutilizables, paneles interactivos y análisis descriptivos y predictivos para respaldar la toma de decisiones basada en datos [35]. La Figura 6 muestra la arquitectura de Power BI.

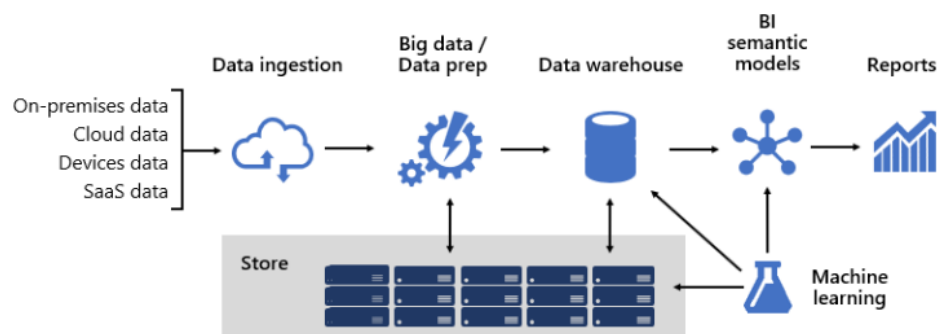


Figura 6. Arquitectura de Power BI [35].

Las plataformas Grafana, Kibana y Power BI son herramientas muy diferentes en cuanto a arquitectura, caso de uso y forma de manipular los datos, y por lo tanto tienen diferentes lugares en el ecosistema de monitoreo y visualización. En la siguiente Tabla 5 se resumen un análisis comparativo de las plataformas de visualización IoT.

Tabla 5. Análisis comparativo de las plataformas de visualización IoT [33], [34], [35].

Característica	Grafana	Kibana	Power BI
Tipo de plataforma	Plataforma de visualización y análisis de series temporales.	Interfaz de exploración y visualización para Elasticsearch.	Plataforma de <i>Business Intelligence</i> compuesta por conectores para análisis y visualización de datos.
Propósito principal	Dashboards, visualización avanzada, alertas y consultas sobre múltiples fuentes.	Análisis de logs, consultas sobre índices y dashboards del stack ELK.	Transformar datos de múltiples orígenes en informes interactivos, dashboards analíticos.
Modelo de datos	Depende de la fuente; no almacena datos.	Basado en índices de Elasticsearch.	Modelos semánticos creados en Power BI Desktop mediante relaciones.
Arquitectura	Cliente-servidor; motor de visualización + plugins + backends.	Integrada al stack ELK (Elasticsearch + Logstash + Kibana).	Compuesta por Power BI Desktop, Power BI Service y Power BI Mobile
Series temporales	Sí, mediante fuentes como InfluxDB, Prometheus o Loki.	Limitado a lo indexado en Elasticsearch.	Compatible a través de modelos importados o DirectQuery
Visualización	Gráficos avanzados, paneles dinámicos, múltiples plugins.	Histogramas, mapas de calor, gráficos Lens, dashboards ELK.	Informes interactivos, dashboards personalizados, segmentadores
Método de recolección de datos	No recolecta; consume datos de otras fuentes.	No recolecta; depende de Elasticsearch y Logstash.	DirectQuery y APIs y servicios en la nube.
Integraciones	Más de 100 fuentes: InfluxDB, Prometheus, MySQL, Loki, etc.	Integración nativa únicamente con Elasticsearch.	SQL Server, Azure, SharePoint, Oracle, APIs REST

1.3.6 Indicadores de desgaste mecánico y eficiencia

Las fallas mecánicas en equipos rotativos, incluyendo las Bombas Sumergibles Eléctricas (ESP), tienen relación principalmente con las partes móviles del ensamblaje de fondo de pozo. Estas averías pueden ser fugas, piezas atascadas o dobladas, roturas o piezas sueltas [36]. Un montaje típico consta de una bomba centrífuga multietapa, un motor sumergible, una sección de sello/protector, un cable de alimentación y equipo de superficie [37].

a. Parámetros de Funcionamiento que Influyen en el Desgaste

El desgaste y las fallas de las ESP rara vez son el resultado de una sola causa, sino de la combinación de varias causas que contribuyen [37].

- *Factores Operacionales y Eléctricos*

Temperatura Alta. El sobrecalentamiento del motor es una de las principales causas de falla del motor [38]. El calor o los cambios extremos de temperatura deterioran el aislamiento eléctrico y provocan fallos en las piezas [37].

Vibración. La vibración excesiva causa fatiga mecánica en las piezas, desalineación del eje y daño en los rodamientos.

Sobrecarga (Overload). Se produce cuando el motor consume demasiada corriente y se sobrecalienta, dañándose. La sobrecarga también puede ser causada por el desgaste extremo [37].

Subcarga (Underload). A menudo señala condiciones de bombeo agotado (pumped-off) o bloqueo por gas (gas-locked) [38].

Ciclos repetitivos. Los ciclos constantes de encendido y apagado (Startup/Shutdown) agotan y desgastan mecánicamente [37].

b. Indicadores Mecánicos de Desgaste Directos o Indirectos

Los sistemas de monitoreo de fondo de pozo capturan datos críticos para el análisis y predicción de fallas. En la Tabla 6 se muestra los Indicadores Mecánicos de Desgaste directos e indirectos [37].

Tabla 6. Indicadores Mecánicos de Desgaste directos e indirectos [38].

Parámetro	Aplicación/Indicación
Vibración	Indica desgaste mecánico de los cojinetes y frecuencias que causan vibración excesiva (Resonancia).
Temperatura del Motor	Muestra el aumento de la temperatura de funcionamiento; un aumento indica sobrecalentamiento, que es una causa principal de falla del motor, a menudo debido al aumento de la fricción (desgaste).
Presión de Descarga	Se utiliza para determinar la cabeza (Head) y la eficiencia de la bomba.
Corriente del Motor	Los cambios indican variaciones en la densidad del fluido y el consumo de energía. Un aumento repentino puede ser señal de sobrecarga o de problemas inminentes.
Presión de Entrada	Estática y dinámica, indica condiciones de flujo.

c. Métodos contemporáneos para detectar desgaste usando datos (IoT + ML)

La predicción de fallas en las ESP necesita datos de alta frecuencia de los equipos de monitoreo de fondo de pozo y superficiales [8].

- Recopilación de datos***

Los Sistemas de Monitoreo de Fondo de Pozo capturan y envían datos críticos como presión de entrada y descarga, temperaturas del motor y de entrada, corriente, voltaje y vibración. Estos datos son la base para los modelos predictivos de datos [8].

- Métodos de aprendizaje automático (ML) para la detección de desgaste.***

El aprendizaje automático (ML) y la inteligencia artificial (AI) se han convertido en enfoques viables para el diagnóstico predictivo de fallas. Los modelos capturan patrones sutiles y tendencias que anticipan las fallas [8].

Ingeniería de características temporales. Esta técnica es fundamental para registrar la forma en que van cambiando las señales de deterioro. Se calculan estadísticas temporales sobre ventanas deslizantes (5-10 min), tales como medias móviles (MA), pendientes de primer orden (slopes) y desviaciones estándar rodantes (rolling standard

deviations). Las variables más predictivas de falla son la temperatura del motor, la pendiente rodante de la presión de entrada y la variación en la vibración [8].

Modelos de aprendizaje. Algoritmos como los de machine learning se emplean para clasificar el estado de funcionamiento de la ESP en: Estable, Inestable y Falla [8].

Preprocesamiento de datos. Bosque de Aislamiento Se usa para detectar y eliminar valores atípicos y mejorar la calidad de los datos.

Análisis de Componentes Principales. Se utiliza para la reducción de dimensionalidad y la de correlación de características, convirtiendo las entradas correlacionadas en componentes ortogonales, lo que mejora la eficiencia computacional y la interpretabilidad [8].

1.3.7 Parámetros de funcionamiento óptimo

El correcto funcionamiento de las bombas electro sumergibles resulta esencial para mantener el caudal de producción y alargar su vida útil, que suele ser más corta de lo previsto por las condiciones extremas del subsuelo: corrosión, altas temperaturas y presiones [39].

a. Características eléctricas del motor.

Los parámetros eléctricos son indicadores de seguridad del motor y estabilidad de funcionamiento, pues revelan la demanda de potencia, las condiciones de carga y la integridad del aislamiento. El monitoreo tradicional con gráficos de amperímetro es poco práctico para detectar fallas incipientes, ya que el cambio en el consumo de energía dependía de las variaciones de voltaje, flujo y densidad del fluido[40].

b. Parámetros hidráulicos

Los parámetros hidráulicos son indicadores para diagnosticar el comportamiento de la bomba y su interacción directa con el pozo, ya que son los que muestran la eficiencia con la que transforma la energía cinética en presión para levantar fluidos [40].

c. *Parámetros mecánicos/térmicos*

La condición térmica y mecánica es determinante para la vida de la ESP, ya que las altas temperaturas y vibraciones son las principales causas de fallas [40]. La temperatura del motor es el principal enemigo y el que más acorta la vida de la ESP. Las altas temperaturas acortan la vida útil y son la principal causa de falla del aislamiento eléctrico [8].

La siguiente Tabla 7 resume Parámetros de Funcionamiento de la Bomba Electrosumergible (ESP), sus rangos de operación o umbrales críticos, unidades originales.

Tabla 7. Parámetros de Funcionamiento de la Bomba Electrosumergible (ESP)[40].

Variable	Rango de Operación	Unidad	Definición
VARIABLES ELÉCTRICAS			
Corriente del motor	20 – 130	A (amperes)	Corriente consumida por el motor ESP.
Voltaje del motor	200 – 420	V (volts)	Voltaje suministrado al motor ESP.
Frecuencia de operación	30 – 70	Hz (hertz)	Frecuencia de operación del variador de velocidad.
Potencia consumida	10 – 150	kW (kilowatts)	Potencia activa consumida por el sistema ESP.
Desbalance de corriente	0 – 5 (normal) / >10 (fallas probables)	% (porcentaje)	Porcentaje de desbalance entre fases.
VARIABLES HIDRÁULICAS			
Presión de succión	5 – 30	bar (bar)	Presión en la entrada de la bomba ESP.
Presión de descarga	40 – 120	bar (bar)	Presión a la salida de la bomba.
Diferencial de presión	20 – 80	bar (bar)	Diferencia entre la presión de descarga y succión.
Caudal bombeado	50 – 300	m ³ /h (metros cúbicos por hora)	Caudal producido por la bomba ESP.
VARIABLES MECÁNICAS / TÉRMICAS			
Vibración RMS	0.5 – 3.0	mm/s (milímetros por segundo)	Vibración radial del conjunto motor-bomba.
Temperatura del motor	60 – 125 (normal) / >140 (falla severa)	°C (grados Celsius)	Temperatura del motor ESP.

1.3.8 Mantenimiento y ciclo de vida

El mantenimiento predictivo se ha convertido en la principal manera de mejorar la eficiencia operativa y disminuir los gastos de mantenimiento en los sistemas ESP,

mediante algoritmos de IA y sensores IoT para el monitoreo y diagnóstico en tiempo real del estado de las bombas [41], [42].

a. Mantenimiento predictivo y detección de fallas

Para hacer un diagnóstico acertado de falla en las ESP, se deben revisar los datos desde que se instaló la bomba hasta que falló. Las mejoras en los sistemas ESP han hecho más accesibles los sensores y sistemas de datos para un diagnóstico temprano de fallas [42].

En la Tabla 8 se ilustra el ciclo operacional y los niveles de mantenimiento, en el cual se categorizó el estado funcional de la ESP en cuatro categorías según el rendimiento de la bomba.

Tabla 8. El ciclo operacional y los niveles de mantenimiento [43].

Estado Operacional	Descripción del Rendimiento	Mantenimiento Recomendado
Normal	Todos los parámetros operacionales están dentro del rango aceptable, indicando un rendimiento óptimo.	No se detectan anomalías.
Aviso 1	Disminución inicial del rendimiento debido a desgaste, cambios de viscosidad o bloqueos menores.	Monitoreo mejorado y mantenimiento preventivo.
Aviso 2	El rendimiento disminuido se ha estabilizado, indicando problemas persistentes pero que no empeoran.	Mantenimiento más proactivo.
Falla	Desviaciones irregulares y significativas, indicando problemas críticos.	Mantenimiento inmediato para prevenir la falla del sistema.

b. Retos de la implementación del mantenimiento predictivo

A pesar de los beneficios, la implementación de sistemas de mantenimiento predictivo presenta varios desafíos:

Costos elevados. Los altos costos se refieren a la modernización de la infraestructura, la adquisición de sensores IoT y la implementación de servidores para el procesamiento de datos en tiempo real [41].

Complejidad del algoritmo. Es arduo adaptar los algoritmos de IA a las condiciones específicas de funcionamiento. La inconsistencia de los datos o la existencia de vacíos por fallas técnicas de los sensores puede degradar la predicción y generar señales de falsos positivos o falsos negativos [41].

Datos variables e incompletos. La capacidad de los algoritmos con datos variables e incompletos no ha sido suficientemente estudiada, lo que es crítico para los sistemas en tiempo real [41].

Personal capacitado. Es necesario formar al personal técnico para manejar los sistemas de monitoreo avanzados, interpretar las predicciones y reaccionar ante fallos potenciales [41].

1.3.9 Hardware y software

Con la finalidad de realizar una adecuada selección de los dispositivos requeridos para el desarrollo del sistema IoT, se identificaron las principales características técnicas de cada elemento involucrado en la adquisición, transmisión, almacenamiento y procesamiento de datos.

Para el entorno de procesamiento se empleó un computador de procesador único, el cual se detalla en la Tabla 9.

Tabla 9. Especificaciones técnicas del equipo portátil [44].

Característica	Especificación
	
Sistema operativo	Microsoft Windows 11 Pro
Versión del sistema	10.0.26200, compilación 26200
Fabricante del sistema operativo	Microsoft Corporation
Modelo del equipo	HP Laptop 15-da0xxx
Tipo de sistema	PC basado en x64
Procesador	Intel(R) Core(TM) i7-8550U CPU @ 1.80 GHz (4 núcleos, 8 hilos)

Para la parte de software se recurrió al uso de tablas comparativas para sintetizar y organizar la información técnica de las herramientas empleadas en el desarrollo del sistema propuesto. Estas tablas permitieron contrastar aspectos como compatibilidad,

requerimientos de hardware e integración con bases de datos y librerías de análisis de datos.

a. Entornos de Entrenamiento Cloud

El proceso de entrenamiento de modelos de machine learning demanda recursos computacionales de alto rendimiento, los cuales no siempre se encuentran disponibles de manera local o mediante infraestructura física dedicada. A continuación, se presentan las plataformas en la nube más relevantes utilizadas para estos fines.

- ***Google Colab***

Google Colab es un entorno de computación alojado en la nube que proporciona un sistema basado en Jupyter Notebook “requiring no setup and offering free access to computing resources, including GPUs and TPUs”, lo que permite ejecutar cargas de trabajo de aprendizaje automático sin configuración local. La plataforma opera como un servicio orientado a investigación y prototipado rápido, integrando bibliotecas estándar como TensorFlow, PyTorch y scikit-learn, que pueden ejecutarse directamente desde el navegador. Su infraestructura gratuita está sujeta a límites de uso, duración de sesión y disponibilidad dinámica de aceleradores según demanda global del servicio. Colab no incluye bases de datos nativas, pero permite interacción directa con Google Drive y servicios externos para la ingestión de datos. Debido a la disponibilidad de aceleración por GPU/TPU, es apto para experimentación con modelos avanzados, aunque no está diseñado como plataforma de producción industrial [45].

- ***Kaggle Notebooks***

Kaggle Notebooks es un entorno de análisis reproducible basado en la nube que permite ejecutar código en Python mediante una interfaz interactiva enfocada en ciencia de datos y aprendizaje automático. Su diseño prioriza la reproducibilidad y la colaboración, facilitando “reproducible and collaborative analysis” e integración nativa con los datasets públicos de la plataforma Kaggle, lo que favorece el acceso inmediato a repositorios de datos curados. El servicio es gratuito y puede habilitar GPU bajo ciertas condiciones, aunque sujeto a restricciones de tiempo y recursos

compartidos. Los notebooks admiten ejecución de frameworks de ML y DL, participación en competencias y documentación sistemática de experimentos. Aunque no constituye una solución industrial para despliegue a gran escala, es ampliamente utilizado para prototipado, benchmarking y experimentación de modelos supervisados y no supervisados [46].

- ***Amazon SageMaker***

Amazon SageMaker es un servicio completamente gestionado de aprendizaje automático que permite construir, entrenar e implementar modelos mediante infraestructura administrada dentro del ecosistema AWS. La plataforma soporta entrenamiento distribuido, integración con frameworks como TensorFlow, PyTorch y MXNet y acceso unificado al almacenamiento mediante arquitecturas de datos tipo lakehouse conectadas a Amazon S3, Redshift y Athena. SageMaker opera bajo un modelo de “pago por uso”, aunque AWS ofrece niveles gratuitos limitados para experimentación inicial. Además, incorpora herramientas de preparación de datos, pipelines de MLOps, gobernanza de activos y capacidades de inferencia en tiempo real o batch, convirtiéndose en una solución apta para aplicaciones industriales y escalables [47]. La Tabla 10 muestra la comparativa de entornos de entrenamiento detallados.

Tabla 10. Comparativa de entornos de entrenamiento Cloud para Machine Learnig.

Criterio	Google Colab	Kaggle Notebooks	Amazon SageMaker
Tipo de plataforma	Entorno alojado de Jupyter Notebook con acceso gratuito a aceleradores (GPU/TPU).	Plataforma de análisis reproducible con notebooks Python/R integrados en la infraestructura de Kaggle.	Servicio completamente gestionado de ML para construir, entrenar y desplegar modelos a escala.
Costo	Gratuito con límites	Gratuito	Pago por uso
Capacidades de cómputo	GPU/TPU gratuitas con límites de sesión.	GPU disponibles según restricciones.	Clusters distribuidos, instancias dedicadas, GPU/CPU de alto rendimiento.
Soporte para frameworks	TensorFlow, PyTorch, scikit-learn, XGBoost, etc.	TensorFlow, PyTorch, scikit-learn, LightGBM, etc.	TensorFlow, PyTorch, MXNet, scikit-learn, HuggingFace, entre otros.
Gestión de datos y datasets	Sin bases internas.	Acceso directo a miles de datasets públicos dentro de Kaggle.	Integración con Redshift, Athena y Data Lake.
Limitaciones	Límites estrictos de uso, tiempos de sesión, y recursos compartidos.	Dependencia de disponibilidad de GPU y tiempo de sesión limitado.	Costos elevados en entrenamiento intensivo; requiere configuración inicial.

b. Plataformas de desarrollo orientadas al control y gestión de información

Un sistema que permita el control de información es vital para crear arquitecturas backend modernas, especialmente en casos que exigen interoperabilidad, escalabilidad y procesamiento eficiente de datos. Estas herramientas dan la posibilidad de especificar la lógica de negocio, exponer servicios a través de interfaces estándar y asegurar el flujo confiable entre sistemas distribuidos.

- ***FastAPI***

FastAPI es un framework de alto rendimiento para elaborar APIs REST en Python, aprovechando type hints y la validación de datos con Pydantic para minimizar errores en tiempo de ejecución y mejorar los flujos de trabajo. Se basa en ASGI (Asynchronous Server Gateway Interface) y servidores como Uvicorn o Hypercorn, que permiten operaciones concurrentes con latencias por debajo de 1 ms en condiciones controladas. A diferencia de otras herramientas más tradicionales, incorpora de manera nativa la generación de documentación interactiva basada en OpenAPI y Swagger UI, mejorando la trazabilidad e interoperabilidad de los servicios. Su arquitectura y la compatibilidad con bibliotecas externas para seguridad, almacenamiento y despliegue han hecho de FastAPI uno de los frameworks más eficientes y escalables en el ecosistema de Python [48].

- ***Flask***

Es un microframework de Python de tipo minimalista que solo ofrece las herramientas necesarias para crear aplicaciones web y APIs, dejando que el usuario añada más componentes según las necesidades de la aplicación. Su estructura es un núcleo adaptable basado en Werkzeug para el manejo de peticiones HTTP y Jinja2 para el renderizado de plantillas, lo que permite separar la lógica de la presentación. Si bien no incluye por defecto utilidades sofisticadas, permite agregar extensiones para autenticación, bases de datos, validación o manejo de sesiones, lo que ha contribuido a su popularidad en proyectos académicos e industriales. A diferencia de frameworks más estructurados, Flask es simple y fácil de aprender, lo que permite crear microservicios rápidamente sin perder modularidad en el diseño [48].

- **CherryPy**

Es un framework orientado a objetos para crear aplicaciones web en Python que tiene la característica de poder funcionar como un servidor HTTP en sí mismo, capaz de servir peticiones concurrentemente. Está construido sobre un motor interno multihilo y un enrutador que permite exponer como recursos web los métodos de una clase, sin tener que crear estructuras complejas ni depender de librerías pesadas. Pero a pesar de ser tan simple, soporta sesiones, error handling, control de caché y herramientas de configuración extensible, lo que asegura su estabilidad en aplicaciones pequeñas y medianas [49].

En la Tabla 11 se presenta un análisis comparativo de plataformas de desarrollo orientadas al control y gestión de información.

Tabla 11. Análisis comparativo de plataformas de desarrollo orientadas al control y gestión de información [48], [49], [50].

Criterio	FastAPI	Flask	CherryPy
Tipo de framework	Framework para APIs REST basado en ASGI	Microframework minimalista para aplicaciones web y APIs	Framework orientado a objetos y servidor HTTP embebido
Arquitectura base	Asincrónica sobre ASGI; compatible con Uvicorn/Hypercorn	Síncrona basada en WSGI mediante Werkzeug	Multihilo con servidor HTTP
Modelo de concurrencia	Soporte nativo para async/await	La concurrencia depende del servidor externo	Multiproceso y multihilo
Rendimiento	Alta velocidad, comparables a Node.js y Go usando ASGI	Desempeño moderado, dependiente de extensiones y servidor WSGI	Rápido y ligero para cargas medias
Componentes internos	Pydantic para validación, Starlette como núcleo de enrutamiento	Werkzeug para HTTP y Jinja2	Servidor HTTP y gestor de sesiones y sistema de configuración
Documentación automática	Sí, generación automática con OpenAPI y Swagger UI	requiere extensiones como Flask-RESTx	No disponible nativamente
Manejo de datos y validación	Validación automática basada en modelos Pydantic	Validación manual o mediante extensiones externas	Validación básica, configurable por el usuario
Facilidad de uso	Estructurado, con tipado y autogeneración de rutas y esquemas	Muy simple, curva de aprendizaje reducida	Minimalista, requiere más configuración manual
Formatos soportados	JSON	JSON, HTML, XML	JSON
Servidor embebido	Requiere ASGI server	Requiere ASGI server	Sí, servidor HTTP nativo incorporado
Licencia	MIT	BSD-3-Clause	BSD

c. Bases de datos de series temporales

Son sistemas de análisis avanzado, monitorización industrial e infraestructuras IoT por su capacidad para manejar grandes volúmenes de datos ordenados en el tiempo. Entre las soluciones más populares se encuentran InfluxDB, TimescaleDB y Prometheus TSDB, cada una con arquitecturas y modelos de funcionamiento adaptados a diferentes necesidades de ingestión, análisis y almacenamiento.

- ***InfluxDB***

Es una base de datos escrita para ingestión de alto rendimiento y análisis de datos marcados por tiempo, la pieza central en aplicaciones IoT, telemetría industrial y monitoreo en tiempo real, como se define en su documentación. Su arquitectura usa un motor de almacenamiento optimizado en los formatos TSM y TSI, que usan compresión, segmentación e índices para manipular millones de puntos por segundo sin degradación de rendimiento. Los datos se estructuran en mediciones, tags y campos, permitiendo consultas rápidas y filtrado eficiente sobre grandes conjuntos de datos. Además, InfluxDB proporciona dos lenguajes de consulta: InfluxQL, similar a SQL, y Flux, para análisis avanzado y procesamiento temporal distribuido. La ingestión se realiza a través del Line Protocol y APIs nativas, garantizando la interoperabilidad con sensores, servicios web y plataformas industriales [51], [52].

- ***TimescaleDB***

Construida como una extensión nativa de PostgreSQL, combinando la potencia relacional de SQL con optimizaciones para datos temporales, según su documentación. Su arquitectura se apoya en hypertables y chunks, particionando datos en bloques temporales y espaciales para optimizar la ingestión, indexación y ejecución de consultas. Este diseño soporta cargas analíticas y transaccionales al mismo tiempo y es totalmente compatible con SQL estándar y el ecosistema de PostgreSQL. Además, TimescaleDB viene con compresión de series temporales >90 % y continuous aggregates que materializan cálculos regulares para acelerar las consultas históricas. Los conectores nativos se integran con sistemas industriales, plataformas IoT y servicios en la nube para una consistencia y disponibilidad continuas [53], [54].

- **Prometheus TSDB**

El motor de almacenamiento de Prometheus, Prometheus TSDB, es una base de datos de series temporales optimizada para métricas numéricas de alta frecuencia y creada bajo la Fundación de Computación Nativa de la Nube CNCF. Su operación implica la formación de bloques de dos horas con muestras comprimidas, un WAL (Write-Ahead Log) para recuperación y procesos de compacto para consolidar datos históricos. El modelo de datos se organiza en etiquetas (labels) tipo clave-valor, permitiendo identificar y filtrar rápidamente miles de series en paralelo. La recolección se lleva a cabo con un modelo pull en el que Prometheus extrae métricas de endpoints HTTP instrumentados, garantizando consistencia temporal y control total sobre la frecuencia de muestreo [55], [56].

En la Tabla 12 se presenta un análisis comparativo de plataformas de desarrollo orientadas al control y gestión de información.

Tabla 12. análisis comparativo de bases de datos de series temporales.

Característica técnica	InfluxDB	TimescaleDB	Prometheus TSDB
Tipo de sistema	Base de datos de series temporales (TSDB) especializada.	Extensión de PostgreSQL diseñada para series temporales.	TSDB integrada en Prometheus para métricas de monitoreo.
Motor de almacenamiento	Motor TSM/TSI optimizado para compresión y escritura secuencial.	Motor PostgreSQL con particionamiento automático mediante <i>hypertables</i> y <i>chunks</i> .	Motor propio basado en segmentos, bloques de 2 horas y WAL.
Modelo de datos	Measurement, Tags, Fields y Timestamp.	Tablas SQL extendidas como <i>hypertables</i> con columnas temporales.	Series identificadas por <i>labels</i> + timestamp.
Método de ingestión	Line Protocol, API HTTP nativa, Telegraf.	SQL estándar, COPY, ingestión continua, ingestión paralela.	Prometheus scrapea endpoints HTTP.
Compresión / optimización	Compresión por bloques TSM y índices TSI	Compresión nativa de <i>chunks</i> con algoritmos específicos para tiempo.	Compresión de muestras dentro de bloques
Lenguaje de consulta	Flux y/o InfluxQL.	SQL completo + funciones temporales.	PromQL, lenguaje especializado en métricas.
Retención de datos	Retención configurables por bucket.	Retención larga mediante <i>chunks</i> + compresión.	Requiere soluciones externas (Thanos, Cortex).
Arquitectura general	Servidor InfluxDB + API + Engine TSM/TSI + Opcional InfluxDB Cloud.	PostgreSQL + Timescale Extension + Hypertables + Continuous Aggregates.	Prometheus Server + TSDB + WAL + Compaction + Exporters.

d. Docker

Es una plataforma de código abierto para desarrollar, empaquetar, distribuir y ejecutar aplicaciones en contenedores. Docker encapsula una aplicación con todas sus dependencias en un contenedor aislado para que se pueda ejecutar en cualquier lugar sin depender de lo que esté instalado en el sistema host. Cada contenedor tiene su propio sistema de archivos, bibliotecas, configuración y procesos, lo que asegura que la aplicación se ejecute de manera consistente sin importar dónde se despliegue. Docker y la contenedorización virtualizan a nivel de sistema operativo: los contenedores comparten el kernel del sistema anfitrión y utilizan mecanismos del kernel para aislar procesos, red, sistema de archivos y limitar recursos, permitiendo que muchos contenedores se ejecuten simultáneamente en un solo host de manera eficiente y con poco overhead, lo que permite la densidad y la portabilidad [57], [58]. En la Tabla 13 se muestra las características de instalación de Docker Desktop.

Tabla 13. Características de instalación de Docker Desktop.

Parámetro	Especificación
Producto	Docker Desktop for Windows
Versión	4.44.2
Desarrollador	Docker Inc.
Tipo de software	Plataforma de contenedores para desarrollo y despliegue de aplicaciones
Sistema operativo anfitrión	Microsoft Windows 11 Pro, 64 bits
Arquitectura soportada	x86_64 sobre WSL2 / Hyper-V
Tamaño de instalación	2,66 GB
Tamaño de ejecución	8 GB de memoria RAM
Componentes principales	Docker Engine, Docker CLI, Docker Compose v2, interfaz gráfica Docker Desktop

- ***Docker Engine Architecture.***

La arquitectura de Docker Engine está diseñada como un modelo modular para ejecutar aplicaciones en contenedores de manera eficiente, con componentes que trabajan juntos para administrar imágenes, redes, volúmenes y ciclo de vida. Su arquitectura se compone de tres partes principales: el Docker Daemon como proceso principal del sistema, la API REST que expone las funcionalidades internas y el Docker CLI como herramienta de línea de comandos para los usuarios. Esta arquitectura sigue un modelo cliente-servidor que permite la comunicación y

automatización en sistemas locales y distribuidos [57]. En la Figura 7 se observa la arquitectura de Docker.

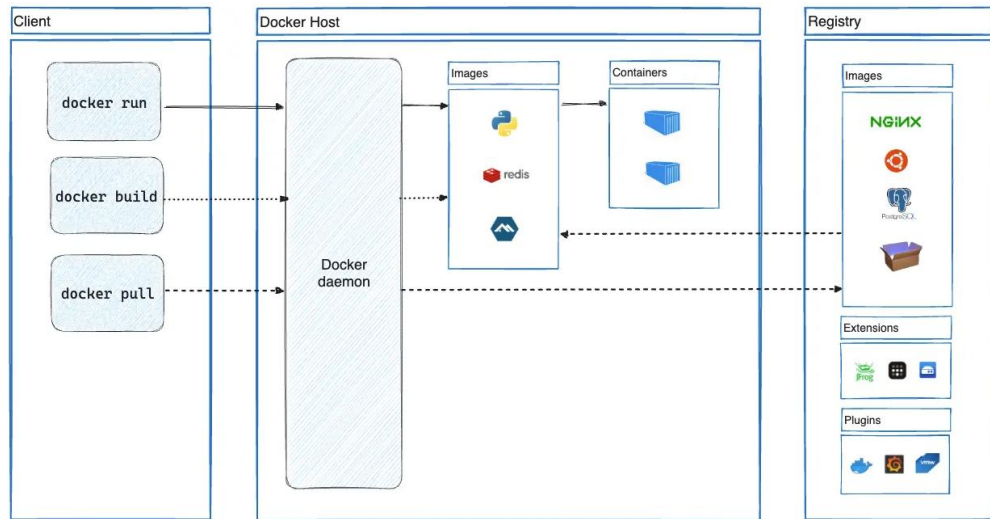


Figura 7. Arquitectura de Docker [57].

- ***Docker Daemon, API y CLI***

El Docker Daemon (`dockerd`) es el corazón de Docker Engine y es quien se encarga de controlar los contenedores, las imágenes, los drivers de red y los volúmenes. Este daemon hace accesible sus funcionalidades a través de una API REST compatible con el protocolo HTTP. Esta separación permite que aplicaciones externas se comuniquen con el daemon sin necesidad de acceso al host, mejorando la escalabilidad y la automatización. Además, la API proporciona endpoints para crear, inspeccionar, monitorear y destruir contenedores, lo que permite una gestión precisa del entorno de ejecución [57].

- ***Containerd y runc***

El containerd es el runtime de nivel medio que usa Docker Engine para administrar el ciclo de vida de los contenedores: crearlos, ejecutarlos, suspenderlos, tomar snapshots y distribuirlos. Este servicio, gestionado por la Cloud Native Computing Foundation (CNCF), proporciona una arquitectura de peso ligero y extensible a través de plugins para soportar distintos sistemas de almacenamiento y controladores de red. En la capa inferior se encuentra `runc`, un runtime compatible con OCI para lanzar contenedores a partir de bundles que especifican el sistema de archivos y la configuración del proceso.

Mientras containerd orquesta tareas complejas, runc simplemente ejecuta los procesos aislados en el kernel del host. La combinación de ambos garantiza que Docker se ajuste a los estándares abiertos de contenedorización, evitando bloqueos tecnológicos y permitiendo la interoperabilidad entre plataformas [57].

- ***Redes en Docker***

La red en Docker se configura a través de drivers de red enchufables que definen diferentes formas en que los contenedores pueden conectarse entre sí, dependiendo de sus necesidades de aislamiento, rendimiento o alcance. Entre los drivers por defecto están bridge, host, overlay, macvlan, ipvlan y none, que se pueden especificar en el momento de la creación de la red. El modo bridge es la red por defecto y crea un puente virtual que permite la comunicación entre contenedores en el mismo host, dándoles una interfaz de red con dirección IP en una subred interna. Para casos distribuidos o multi-host, Docker proporciona la red overlay, que permite conectar contenedores/servicios en diferentes hosts en la misma red lógica. Esta red se puede utilizar tanto para servicios orquestados como para contenedores independientes, siempre y cuando los hosts estén en un clúster de Docker. En cuanto a la asignación de IPs y puertos, Docker asigna subredes IPv4 a cada red que crea; cuando se inicia un contenedor, éste recibe una IP de esa red. Para hacer visibles al exterior los servicios hay que publicar puertos con la directiva publish, mapeando un puerto del host a uno del contenedor [57].

- ***Docker Compose en IoT + Machine Learning***

Docker Compose es una herramienta de configuración ligera para definir y ejecutar múltiples contenedores a través de archivos declarativos (.yaml). Su objetivo es orquestar servicios que deben ejecutarse juntos, definiendo dependencias, configuraciones de red y políticas de persistencia sin la sobrecarga de sistemas de orquestación como Kubernetes. Los servicios en Docker Compose son como unidades lógicas que encapsulan contenedores con roles específicos, y puedes definir la imagen, las variables de entorno, las dependencias, los mapeos de puertos y las políticas de reinicio en el archivo docker-compose.yaml. Compose crea redes virtuales internas que permiten que los servicios se comuniquen directamente entre sí utilizando la resolución DNS integrada en lugar de tener que codificar direcciones IP. Además, la

herramienta gestiona volúmenes persistentes para almacenar datos importantes fuera del ciclo de vida del contenedor, garantizando su supervivencia en bases de datos, modelos de aprendizaje automático o registros. En un ambiente IoT con Machine Learning, Docker Compose facilita que cada parte se despliegue de manera orquestada y replicable, para adquisición de datos, procesamiento en tiempo real, almacenamiento en series temporales y visualización analítica con compatibilidad entre versiones y dependencias [57].

1.3.10 Enfoque metodológico para Machine Learning.

La implementación de soluciones que se fundamentan en Machine Learning precisa la aplicación de metodología estructurada que guíen la ejecución de todo el proceso, desde la comprensión del problema hasta la validación final y la puesta en producción del modelo.

- ***CRISP-DM (Cross Industry Standard Process for Data Mining)***

Es modelo metodológico que se puede considerar independiente de la disciplina de minería de datos, orientado a estructurar proyectos de minería de datos a través de un procedimiento claro, repetible y dirigido; por lo tanto, su utilización permite estructurar sistemáticamente la recopilación de datos, el análisis y el tratamiento de grandes volúmenes de información, lo que representa un elemento clave en las empresas actuales donde todo depende de determinar métricas precisas que guíen la toma de decisiones [59]. En este sentido, el modelo establece que existe un ciclo de vida de la minería de datos que consiste en seis fases que son iterativas, afianzadas desde la comprensión del negocio hasta la implementación, lo cual permite el intercambio de información entre los participantes de un proyecto y consigue la trazabilidad de los resultados ahí obtenidos. Se compone de seis fases, las cuales son iterativas y su secuencia no es rígida. A continuación, se presenta la composición y las actividades de cada fase [60]. En la Figura 8 se muestra las fases de trabajo de la metodología CRISP-DM.

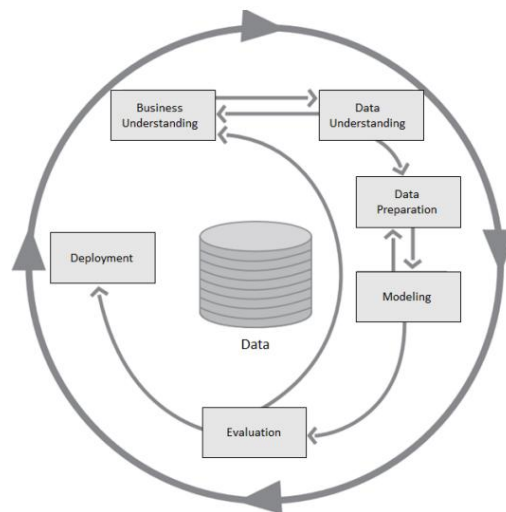


Figura 8. Fases de trabajo de la metodología CRISP-DM [60].

Comprensión del Negocio (Business Understanding). Se centra en comprender los objetivos y requisitos empresariales del proyecto, se obtiene el conocimiento de negocio para ser una definición clara del problema de minería de datos y se establecen unos primeros planes. Se evalúa la situación y la forma de evaluar el éxito de la minería de datos[60].

Comprensión de los Datos (Data Understanding). Comienza con la recopilación inicial de datos para obtener familiaridad con ellos. Su objetivo es identificar problemas de calidad de los datos y obtener primeras ideas que permitan formular hipótesis sobre información oculta [60].

Preparación de Datos (Data Preparation). Incluye la preparación de los datos con el propósito de construir el conjunto de datos final que se utilizará en la fase de modelado. Por tanto, incluye la selección, limpieza y transformación de tablas, registros y atributos; es con frecuencia la fase más larga de realizar [60].

Modelado (Modeling). Se llevan a cabo la selección de las distintas técnicas de modelado y la calibración de sus parámetros en busca de valores óptimos. La técnica para emplear dependerá de la problemática de negocio y de los datos disponibles [60].

Evaluación (Evaluation). Se lleva a cabo una evaluación en profundidad del modelo construido y se revisan los pasos dados con la intención de poder contrastar que el modelo esté haciendo llegar adecuadamente los objetivos empresariales planteados.

Uno de los objetivos clave será el de determinar si se ha tenido en cuenta suficientemente algún problema de negocio relevante. Una vez que culmina esta fase es necesario la toma de decisiones correspondiente respecto a lo que se hará con los resultados de la minería de datos [60].

Despliegue (Deployment). Consiste en empaquetar el modelo de forma que el cliente u organización custodien de estos resultados y el conocimiento que se ha podido construir gracias a la aplicación de las técnicas de minería de datos. Las actividades pueden ser tan diversas como generar un informe sencillo hasta la construcción de un proceso de minería de datos, pero las topologías son tan diversas como las tareas pasadas [60].

1.4 Objetivos

1.4.1 Objetivo general

Desarrollar un sistema IoT con análisis predictivo para el monitoreo de la salud de los sistemas de bombeo de campos petroleros.

1.4.2 Objetivos específicos

- Estructurar una base de datos de sensores IoT de sistemas de bombeo de campos petroleros que permita identificar parámetros críticos vinculados a la salud operativa de los equipos.
- Aplicar un modelo de análisis predictivo basado en machine learning integrando los datos históricos de los sensores para determinar probabilidades de fallo en los sistemas de bombeo.
- Construir una plataforma de visualización que genere alertas sobre necesidades de mantenimiento y detección temprana de fallos, presentando indicadores del estado de salud de los motores de bombeo.

CAPÍTULO II. METODOLOGÍA

2.1 Materiales

En el desarrollo de la presente investigación se utilizaron recursos de hardware y software orientados a al diseño e implementación de una solución de monitoreo y análisis predictivo aplicada a bombas de levantamiento artificial tipo ESP. Como plataforma de cómputo se empleó un computador portátil HP Laptop 15-da0xxx, equipado con un procesador Intel® Core™ i7-8550U y sistema operativo Microsoft Windows 11 Pro, sobre el cual se desplegó el entorno de virtualización y contenedores para la ejecución de los servicios del sistema. En cuanto al software, se utilizó Google Colab como entorno principal de experimentación. Se emplearon librerías como pandas y NumPy para el tratamiento de datos; scikit-learn, XGBoost, CatBoost y LightGBM para el entrenamiento de modelos de clasificación multiclase; y joblib para la serialización de los modelos. Para el despliegue se implementó una API REST con FastAPI y Uvicorn. Esta fue contenedorizada mediante Docker Desktop e integrada a un stack de monitoreo basado en InfluxDB como base de datos de series de tiempo y Grafana para la visualización de tableros e indicadores de salud de las bombas. Además, se complementó con Google Colab como entorno de apoyo para el análisis exploratorio y la validación de los modelos.

2.2 Métodos

2.2.1 Modalidad de la investigación

En esta sección se describieron los tipos de investigación empleados durante el desarrollo de la investigación, junto con el proceso que permitió llevar a cabo cada una de sus etapas. Se realizó una investigación de tipo aplicada, ya que se enfocó en el diseño y desarrollo de un sistema IoT con análisis predictivo para el monitoreo de la salud de sistemas de bombeo en campos petroleros. En su implementación se aplicaron conocimientos adquiridos a lo largo de la carrera universitaria, como programación, procesamiento de señales, análisis de datos, sistemas embebidos, redes de comunicación, algoritmos de machine learning y arquitecturas IoT.

En otro ámbito, se efectuó una investigación de tipo bibliográfico, que implicó una revisión y análisis crítico de la literatura existente sobre sistemas IoT aplicados a la industria petrolera, técnicas de análisis predictivo para mantenimiento industrial, arquitecturas de monitoreo remoto, algoritmos de machine learning para predicción de fallas, protocolos de comunicación para ambientes industriales y casos de estudio de implementaciones exitosas en campos petroleros.

Finalmente, esta investigación fue de tipo experimental, ya que implicó la implementación y validación de un sistema funcional mediante simulaciones controladas utilizando datos reales de variables operacionales de motores y bombas petroleras. Se realizaron experimentos específicos para evaluar la precisión de los algoritmos predictivos, la eficiencia de la arquitectura IoT propuesta y la robustez del sistema en la detección temprana de anomalías y predicción de fallas en sistemas de bombeo.

2.2.2 Población y muestra

En la presente investigación no se consideró necesaria la definición de una población ni la selección de una muestra, ya que el estudio no se orientó al análisis de sujetos humanos, grupos sociales o fenómenos demográficos. El enfoque del trabajo fue de carácter tecnológico y experimental, centrado en el diseño, desarrollo e implementación de un sistema IoT con análisis predictivo para el monitoreo de la salud de sistemas de bombeo en campos petroleros. Por tanto, la validación del sistema se realizó mediante pruebas controladas con datos operacionales, sin requerir la participación de una muestra representativa de una población.

2.2.3 Recolección de información

Con el objetivo de desarrollar un sistema IoT con análisis predictivo para el monitoreo de la salud de sistemas de bombeo en campos petroleros, se definió una estrategia metodológica que detalló los elementos clave del proceso de recolección de información. Dicha planificación consideró los fines del estudio, los dispositivos tecnológicos involucrados, los métodos de obtención de datos, los recursos necesarios, el cronograma de trabajo y el entorno de desarrollo. A continuación, se presenta la

descripción estructurada de cada uno de estos criterios fundamentales para la ejecución del proyecto en la Tabla 14.

Tabla 14. Recolección de información.

Criterio	Descripción
¿Para qué?	Obtener información técnica y estructurada que permita construir una base de datos con parámetros críticos de operación, entrenar modelos predictivos de fallos y validar una plataforma de visualización para el monitoreo de sistemas de bombeo.
¿De qué personas u objetos?	De datos industriales de sensores, obtenidos a partir de un dataset.
¿Cómo?	Mediante la recopilación de base de datos, simulaciones controladas, análisis de variables críticas mediante herramientas de software y machine learning.
¿Con qué?	Empleando software de análisis de datos, documentación técnica y herramientas de visualización.
¿Cuándo?	Agosto 2025 – enero 2026.
¿Dónde?	En la Facultad de Ingeniería en Sistemas, Electrónica e Industrial (FISEI) de la Universidad Técnica de Ambato, bajo condiciones controladas de simulación.

2.2.4 Procesamiento y análisis de datos

Los datos recopilados se procesaron mediante las siguientes etapas:

- Codificación técnica y estructuración de la base de datos: Se organizaron los datos históricos y simulados obtenidos de registros técnicos y pruebas de laboratorio. Se identificaron variables clave, estructurando la base de datos para facilitar su análisis y el entrenamiento de modelos predictivos.
- Selección de características y entrenamiento del modelo: Se aplicaron técnicas de análisis exploratorio y selección de características para determinar las variables más relevantes en la predicción de fallos. Posteriormente, se entrenaron modelos de machine learning como Random Forest o SVM, evaluando su rendimiento mediante métricas como precisión, recall y F1-Score.
- Evaluación del sistema de visualización y generación de alertas: Se analizó el comportamiento de la plataforma de visualización desarrollada, verificando la actualización dinámica de indicadores, la precisión en la generación de alertas y la capacidad de respuesta ante condiciones críticas simuladas.

- Síntesis técnica y validación del sistema: Se elaboró un informe técnico que resumió los resultados obtenidos en cada etapa, destacando la efectividad del sistema IoT propuesto. Se identificaron fortalezas, limitaciones y recomendaciones para su implementación en entornos reales de operación petrolera.

CAPÍTULO III. RESULTADOS Y DISCUSIÓN

En el desarrollo de la investigación se aplicó la metodología CRISP-DM, debido a su enfoque estructurado para proyectos de minería de datos; sus fases se sintetizan en la Figura 8, Fases de trabajo de la metodología CRISP-DM [61], con el propósito de estructurar de manera sistemática el diseño del sistema IoT con análisis predictivo para el monitoreo de la salud de los sistemas de bombeo en campos petroleros. En la etapa de comprensión del negocio se definieron los objetivos operativos del mantenimiento predictivo y se identificaron las variables críticas asociadas al desempeño de las bombas electrosumergibles (ESP). A continuación, durante la comprensión de los datos, se examinaron las bases de información disponibles, evaluando su estructura, consistencia y disponibilidad temporal. Posteriormente, la preparación de los datos permitió depurar la información mediante procesos de limpieza, normalización y generación de características, consolidando una base única apta para el entrenamiento de los modelos. En la fase de modelado se entrenaron algoritmos de Machine Learning supervisado, ajustando hiperparámetros y validando su rendimiento mediante métricas especializadas para series temporales. Seguidamente, en la fase de evaluación, se verificó el grado de cumplimiento de los objetivos mediante análisis de precisión, estabilidad y capacidad de predicción anticipada. Finalmente, en la etapa de despliegue, las predicciones generadas se integraron en una arquitectura IoT a través de un clúster de Kubernetes conectado a una API, InfluxDB y Grafana, como se ilustra en la Figura 9, que muestra la arquitectura de despliegue del sistema. Esta integración permitió la visualización y el monitoreo continuo del estado operativo de los equipos, consolidando el resultado principal del sistema propuesto.

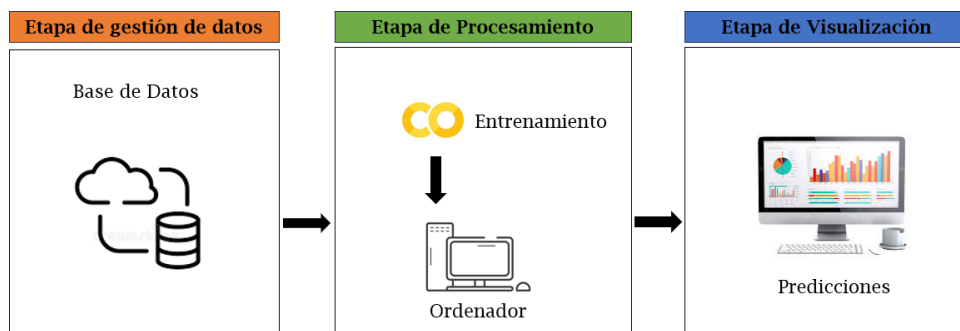


Figura 9. Estructura del sistema IoT.

3.1 Desarrollo de la metodología CRISP-DM

3.1.1 Comprensión de negocio

En el contexto de la industria petrolera ecuatoriana y, en general, de América Latina, los sistemas de levantamiento artificial mediante bombas de cavidad progresiva y, especialmente, bombas eléctricas sumergibles (ESP) constituían un componente crítico para sostener la producción en yacimientos maduros y en fases avanzadas de declinación. Estos equipos operaban en ambientes altamente agresivos, con elevados contenidos de agua, gas y sólidos, lo que incrementaba la probabilidad de fallas prematuras y paros no programados. A ello se sumaba que muchos campos se ubicaban en zonas remotas de la Amazonía y de la costa, donde el acceso a personal especializado era limitado y los costos asociados a intervenciones de pozo, logística y pérdida de producción resultaban particularmente elevados. En este escenario, la gestión eficiente de la salud operativa de las ESP se convertía en un factor estratégico para garantizar la continuidad del negocio y la competitividad de las operaciones petroleras en la región.

No obstante, en gran parte de estas operaciones la supervisión de las ESP seguía siendo predominantemente reactiva, apoyada en registros manuales, informes periódicos y sistemas de control que no se encontraban plenamente integrados con plataformas de análisis avanzado. Esta situación provocaba que problemas de naturaleza eléctrica, hidráulica o mecánica se detectaran de forma tardía, favoreciendo fallas repentinas del motor, el desgaste acelerado del sistema de bombeo y condiciones de operación fuera de los rangos recomendados por el fabricante. Además, el uso limitado de tecnologías IoT impedía que variables críticas como corriente, voltaje, frecuencia del variador, presiones de succión y descarga, caudal, vibración y temperatura se capturaran y procesaran de manera continua y centralizada, lo que dificultaba la automatización del monitoreo y la construcción de indicadores tempranos de riesgo de falla. En consecuencia, el problema de negocio que abordó esta investigación se formuló como la ausencia de un sistema integrado de monitoreo y análisis predictivo que permitiera anticipar y gestionar de forma proactiva las fallas de las bombas eléctricas

sumergibles, reduciendo los paros no programados y los costos operativos en campos petroleros remotos.

Frente a esta problemática, la presente investigación planteó el desarrollo de un sistema IoT con capacidades de análisis predictivo, basado en técnicas de machine learning, orientado a la supervisión de la salud de bombas eléctricas sumergibles en campos petroleros. La propuesta integró la adquisición y transmisión remota de variables operacionales de las ESP con modelos de aprendizaje supervisado fundamentados principalmente en algoritmos de ensamble basados en árboles de decisión, cuya caracterización se describió en la Tabla 3 del Capítulo 1. Se optó por utilizar los modelos Random Forest, XGBoost, CatBoost y LightGBM, entrenados para clasificar el estado de la bomba y tipificar la falla según su naturaleza eléctrica, hidráulica o mecánica/térmica, así como para estimar la probabilidad de ocurrencia y la severidad de fallas en un horizonte de tiempo definido. El objetivo de negocio se centró en disponer de una herramienta de soporte a la decisión para mantenimiento predictivo que contribuyera a disminuir la frecuencia y duración de paros no programados, optimizar los planes de intervención y favorecer una operación más segura, eficiente y sostenible de las ESP en el contexto petrolero ecuatoriano y latinoamericano.

3.1.2 Comprensión de datos

En el Capítulo 1 se definieron y dimensionaron los principales parámetros eléctricos, hidráulicos, mecánicos y térmicos asociados al funcionamiento de las bombas electrosumergibles, los cuales se sintetizan en la Tabla 5, “Parámetros de Funcionamiento de la Bomba Electrosumergible (ESP)” [41]. Estos parámetros constituyeron la base conceptual para la creación del conjunto de datos utilizado en esta investigación, ya que representaron las variables clave del funcionamiento del sistema de bombeo. Sobre esta fundamentación, el sistema propuesto fue sometido a pruebas controladas en los laboratorios de la Universidad Técnica de Ambato, con el fin de validar su desempeño bajo condiciones representativas de operación industrial. De este modo, la comprensión de los datos se articuló directamente con el contexto teórico y experimental establecido en los capítulos precedentes.

a. Recopilación de datos

La recopilación de la información se inició con una búsqueda sistemática de datasets en repositorios especializados. En primera instancia, la exploración se realizó en Google Académico, empleando combinaciones de palabras clave orientadas a “*predictive maintenance*”, “*time series*”, “*sensores industriales*” y variables eléctricas, hidráulicas, mecánicas y térmicas asociadas a sistemas de bombeo. A partir de esta búsqueda preliminar se identificaron como fuentes principales los repositorios MendeleyData, Kaggle y el UCI Machine Learning Repository, seleccionados por su reconocimiento académico, disponibilidad de metadatos estructurados y acceso abierto a conjuntos de datos en formatos compatibles (*CSV*, *XLSX*). Estas plataformas permitieron localizar y analizar datasets de referencia, que sirvieron como insumo para definir el esquema de variables, los rangos de operación y los criterios de calidad de los datos requeridos para el entrenamiento y validación de los modelos de análisis predictivo que se presentaron en el Capítulo 3 de esta investigación.

b. selección de Variables

Dado que ninguna de las bases de datos identificadas contenía de forma aislada la totalidad de variables eléctricas, hidráulicas y mecánicas/térmicas requeridas para el análisis integral del sistema de bombeo, se determinó que su utilización individual resultaba insuficiente para caracterizar con precisión las condiciones de operación de las bombas. En consecuencia, se optó por combinar los datasets provenientes de MendeleyData, Kaggle y UCI en un solo repositorio unificado, conservando aquellas variables con mayor relevancia física y operacional. Esta integración permitió aumentar la resolución y diversidad de los escenarios de funcionamiento representados, mejorando la capacidad de los modelos para aprender patrones de falla y normalidad. Así, se logró una base de datos mejorada y coherente, que fue considerada adecuada para las siguientes etapas de preparación, modelado y evaluación en la metodología CRISP-DM. En la Tabla 15 se muestra la distribución de registros por tipo de variables y fuente de datos.

Tabla 15. Distribución de registros por tipo de variables y fuente de datos.

Categorías de variables	Fuentes	Número de registros asociados por una sola bomba
Eléctricas	Kaggle (Pump Sensor Data)	150 000
	UCI (Hydraulic System)	30 000
	Total	180 000
Hidráulicas	Kaggle (Pump Sensor Data)	130 000
	UCI (Hydraulic System)	30 000
	Total	160 000
Mecánicas / Térmicas	MendeleyData (ESP vibración)	50 000
	UCI (Hydraulic System)	63 287
	Total	113 287
Número de bombas por dataset	Kaggle: 30 bombas; MendeleyData: 28 bombas; UCI: 26 bombas	
Total		453 287

c. Proceso de ingeniería de datos

A partir de los datasets seleccionados se llevó a cabo un proceso de ingeniería de datos orientado a obtener un conjunto de información coherente con el comportamiento de bombas electrosumergibles empleadas en la extracción petrolera. Este proceso permitió integrar y transformar los registros procedentes de Kaggle, MendeleyData y UCI hasta conformar un único repositorio, estructurado para representar el funcionamiento de 25 bombas, cada una asociada a aproximadamente 105.120 registros, alcanzando un total cercano a 2.628.000 observaciones. De forma resumida, las principales etapas fueron:

- Selección de variables relevantes asociadas a bombas ESP tomando como referencia la Tabla 5. Parámetros de funcionamiento de la bomba electrosumergible (ESP).
- Normalización semántica de columnas y unidades de medida, unificando nombres de campos y escalas físicas entre las distintas fuentes para garantizar compatibilidad y trazabilidad.
- Limpieza y aseguramiento de calidad de los datos, incluyendo detección y tratamiento de valores atípicos, eliminación de registros duplicados, manejo de datos faltantes y homogenización de la frecuencia de muestreo temporal.

- Generación de etiquetas unificadas bajo un esquema de clasificación multiclase, diferenciando operación normal y diversos tipos de falla eléctrica, hidráulica y mecánica/térmica.
- Balanceo de clases mediante técnicas de submuestreo y SMOTE, incrementando de forma sintética las clases minoritarias de falla para evitar sesgos en el entrenamiento de los modelos.
- Particionamiento lógico por bomba, distribuyendo el conjunto depurado y balanceado entre 25 bombas con un número similar de registros por activo.

d. Exploración de datos

En función del proceso de integración y depuración descrito, el dataset quedó estructurado en torno a una categoría de referencia asociada al estado normal de operación y tres categorías principales de variables: eléctricas, hidráulicas y mecánicas/térmicas. Aproximadamente el 56,659 % de los registros correspondió a la condición Normal, mientras que cerca del 20,948 % se asoció a variables eléctricas (corriente, voltaje, frecuencia, potencia, desbalance armónico), el 16,114 % a variables hidráulicas (presiones de succión y descarga, diferencial de presión, caudal) y el 6,279 % restante a variables mecánicas y térmicas (vibración y temperatura del motor). En la Figura 10 se muestra la distribución de registros por variable.

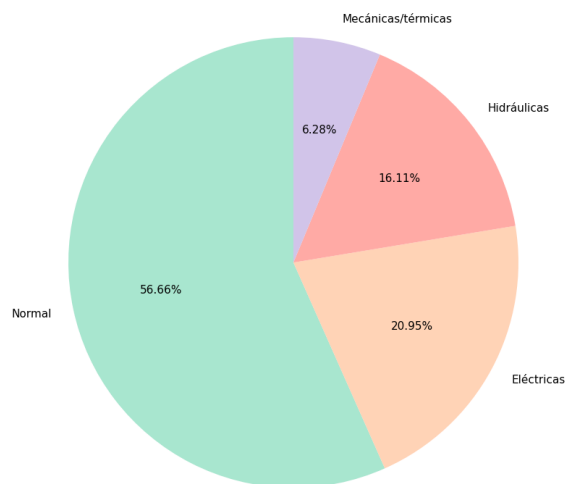


Figura 10. Distribución de registros por variable.

El dataset *DataSet_BombaESP.csv* se conformó a partir de la integración y depuración de variables eléctricas, hidráulicas y mecánicas/térmicas de bombas electrosumergibles, estandarizando tipos de dato, rangos de operación y fuentes de origen. En la Tabla 16 se muestra la estructura del dataset.

Tabla 16. Estructura del DataSet_BombaESP.csv

Nombre de la variable	Tipo de variable	Rango de operación (normal)	Descripción de la variable	Fuente de origen (dataset)
bomba_id	string	Bomba 1 – bomba 25	Id de la bomba	Kaggle, UCI
tiempo_operacion_h	string	2023-01-01 – 2023-12-31	Tiempo de operación	Kaggle, UCI
Normal	float	N/A	Funcionamiento Óptimo	MendeleyData, UCI y Kaggle
corriente_motor_a	float	20 – 130 A	Corriente del motor	Kaggle, UCI
voltaje_motor_v	float	200 – 420 V	Voltaje del motor	Kaggle, UCI
frecuencia_hz	float	30 – 70 Hz	Frecuencia de operación	Kaggle
potencia_kw	float	10 – 150 kW	Potencia consumida	Kaggle
arm_desbalance_pct	float	0 – 5 %	Desbalance de corriente	Kaggle
presion_succion_bar	float	5 – 30 bar	Presión de succión	Kaggle, UCI
presion_descarga_bar	float	40 – 120 bar	Presión de descarga	Kaggle, UCI
delta_presion_bar	float	20 – 80 bar	Diferencial de presión	Kaggle, UCI
caudal_m3_h	float	50 – 300 m³/h	Caudal bombeado	Kaggle, UCI
vibracion_rms_mm_s	float	0,5 – 3,0 mm/s RMS	Vibración RMS	MendeleyData, UCI
temperatura_motor_c	float	60 – 125 °C	Temperatura del motor	UCI

El conjunto final representó 25 bombas, cada una con 105 120 registros, alcanzando un total de 2 628 000 observaciones, distribuidas en tres categorías de variables: eléctricas, hidráulicas y mecánicas/térmicas. La Figura 11 describe la consolidación de los registros para el archivo *DataSet_BombaESP.csv*, provenientes de los tres datasets fuente (Kaggle, MendeleyData y UCI), en el cual los datos originales fueron sometidos a depuración, eliminación de outliers y homogenización de variables.

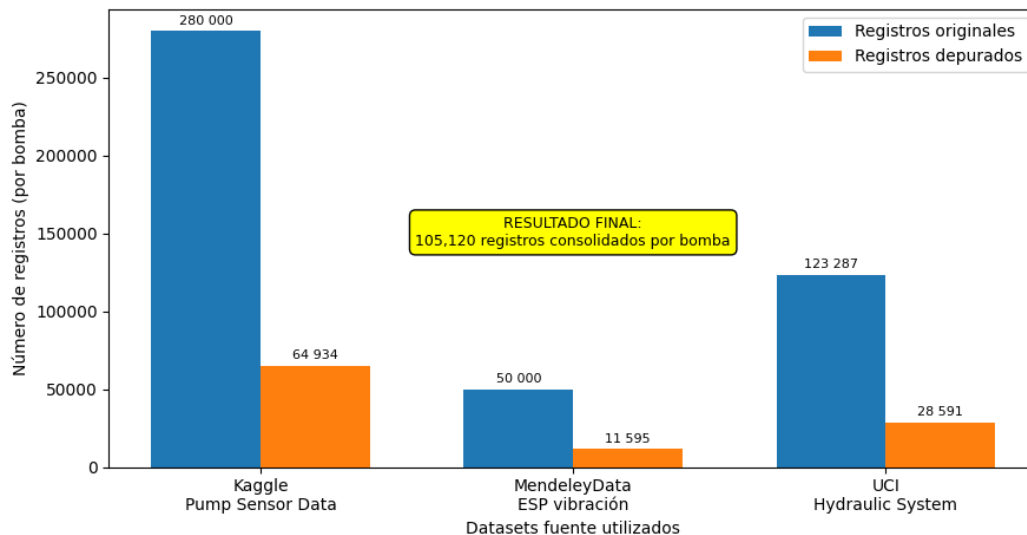


Figura 11. Distribución de registros por variable.

3.1.3 Preparación de datos

En esta etapa se realizó la preparación de los datos, que incluyó el preprocesamiento, limpieza, etiquetado y organización correcta de los registros del archivo DataSet_BombaESP.csv para las 25 bombas, ajustando formatos temporales, rangos de operación y categorías de falla.

a. *Etiquetado.*

En la etapa de etiquetado de datos se utilizó Google Colab como entorno principal para cargar el archivo DataSet_BombaESP.csv y asignar de manera sistemática las clases de falla a cada registro. Una vez importado el dataset, se definieron las etiquetas de salida del modelo (Y) y se procedió a codificar, mediante valores numéricos, tanto el sensor asociado a la falla como el tipo de subcomponente afectado, la severidad del evento, el estado global de la bomba y la probabilidad de falla en las próximas 24 horas. La selección de las variables se basó en la Tabla 7, Parámetros de Funcionamiento de la Bomba Electrosumergible (ESP) [41]. En la Tabla 17 se muestra la codificación unificada de las etiquetas de salida.

Tabla 17. Codificación unificada de las etiquetas de salida.

Etiqueta	Código	Significado
variable_falla	0	Sin falla
variable_falla	1	corriente_motor_a
variable_falla	2	voltaje_motor_v
variable_falla	3	frecuencia_hz
variable_falla	4	potencia_kw
variable_falla	5	arm_desbalance_pct
variable_falla	6	presion_succion_bar
variable_falla	7	presion_descarga_bar
variable_falla	8	delta_presion_bar
variable_falla	9	caudal_m3_h
variable_falla	10	vibracion_rms_mm_s
variable_falla	11	temperatura_motor_c
subcomponente_falla	0	Normal
subcomponente_falla	1	Falla Eléctrica
subcomponente_falla	2	Falla Hidráulica
subcomponente_falla	3	Falla Mecánica / Térmica
severidad_falla	0	Normal
severidad_falla	1	Alerta temprana
severidad_falla	2	Alarma
severidad_falla	3	Falla crítica
estado_bomba	0	Normal
estado_bomba	1	Falla
falla_predicha_24h	0	No fallará en 24 h
falla_predicha_24h	1	Fallará en 24 h

b. Unificación y tipado de campos

Se efectuó la unificación y tipado de los campos del archivo *DataSet_BombaESP.csv*, asegurando la consistencia semántica y de formato entre todas las variables. En la Figura 12 se detalla el Script de unificación de identificadores y variables numéricas.

```
df = pd.read_csv("DataSet_BombaESP.csv")
# Id y tiempo
df["bomba_id"] = df["bomba_id"].astype(str)
df["tiempo_operacion_h"] = pd.to_datetime(df["tiempo_operacion_h"])
# Variables numéricas (X)
feature_cols = [
    "corriente_motor_a", "voltaje_motor_v", "frecuencia_hz", "potencia_kw",
    "arm_desbalance_pct", "presion_succion_bar", "presion_descarga_bar",
    "delta_presion_bar", "caudal_m3_h", "vibracion_rms_mm_s",
    "temperatura_motor_c"
]
df[feature_cols] = df[feature_cols].astype("float32")

# Etiquetas (Y)
label_cols = [
    "variable_falla", "subcomponente_falla",
    "severidad_falla", "estado_bomba", "falla_predicha_24h"
]
df[label_cols] = df[label_cols].astype("int32")
```

Figura 12. Script de unificación de identificadores y variables numéricas.

Para ello, se definió a *bomba_id* y *tiempo_operacion_h* como identificadores categóricos/temporales, mientras que las variables eléctricas, hidráulicas y

mecánicas/térmicas se normalizaron al tipo float, y las etiquetas de salida se codificaron como enteros. Este proceso permitió disponer de una estructura de datos homogénea y lista para su depuración.

c. Limpieza de duplicados, NaN, rangos imposibles y outliers

Se ejecutó un proceso sistemático de limpieza del *dataset*, eliminando registros duplicados, corrigiendo valores perdidos y descartando observaciones fuera de los rangos físicos definidos para cada variable. Adicionalmente, se aplicó un recorte de *outliers* a los límites operativos de las bombas, de modo que los datos conservaran coherencia con las condiciones reales de funcionamiento y se redujera el impacto de valores extremos en el entrenamiento de los modelos. En la Figura 13 se puede observar el script de limpieza de duplicados, valores faltantes y outliers.

```
# Quitar duplicados
df = df.drop_duplicates()
# Imputar NaN con la mediana de cada variable
for col in feature_cols:
    df[col].fillna(df[col].median(), inplace=True)
# Filtrar rangos físicamente imposibles
df = df[
    ("corriente_motor_a": (0, 500), "voltaje_motor_v": (0, 5000), "frecuencia_hz": (0, 200),
    "potencia_kw": (0, 1000), "arm_desbalance_pct": (0, 100), "presion_succion_bar": (0, 200),
    "presion_descarga_bar": (0, 300), "delta_presion_bar": (0, 300), "caudal_m3_h": (0, 1000),
    "vibracion_rms_mm_s": (0, 50), "temperatura_motor_c": (0, 300),
    )
]
# Recortar outliers a rangos operativos normales
rangos_operativos = {
    "corriente_motor_a": (20, 130), "voltaje_motor_v": (200, 420), "frecuencia_hz": (30, 70),
    "potencia_kw": (10, 150), "arm_desbalance_pct": (0, 5), "presion_succion_bar": (5, 30),
    "presion_descarga_bar": (40, 120), "delta_presion_bar": (20, 80), "caudal_m3_h": (50, 300),
    "vibracion_rms_mm_s": (0.5, 3.0), "temperatura_motor_c": (60, 125),
}
```

Figura 13. Script de limpieza de duplicados, valores faltantes y outliers.

d. Balanceo de clases con SMOTE

Con el objetivo de evitar sesgos del modelo hacia la condición normal de operación, se realizó el balanceo de clases sobre el conjunto de entrenamiento. En la Figura 14 se observa este método, para ello se utilizó la técnica SMOTE, tomando como referencia la etiqueta estado_bomba (normal/falla) y generando ejemplos sintéticos en las clases minoritarias. De este modo, se obtuvo un conjunto de entrenamiento equilibrado en términos de ocurrencia de fallas, manteniendo la coherencia de las demás etiquetas asociadas a cada registro.

```

from imblearn.over_sampling import SMOTE
# X_train -> variables de entrada del conjunto de entrenamiento
# y_train_estado -> columna 'estado_bomba' del conjunto de entrenamiento
smote = SMOTE(random_state=42)
X_train_bal, y_train_estado_bal = smote.fit_resample(
    X_train, y_train_estado
)

```

Figura 14. Aplicación de SMOTE para el rebalanceo de clases.

e. *Dataset Consolidado*

El archivo *DataSet_BombaESP_Completo.csv* correspondió al dataset unificado y depurado resultante del proceso de ingeniería de datos, conformado por 2 628 000 muestras y 18 variables, que incluyeron tanto las variables operativas de la bomba como las etiquetas de falla generadas a partir de las fuentes Kaggle, MendeleyData y UCI, como se observa en la Figura 15. Sobre este conjunto se aplicaron de forma secuencial la estandarización de tipos de dato, la corrección de valores faltantes, la eliminación de registros inconsistentes y el recorte de *outliers* de acuerdo con los rangos físicos de operación de las bombas electrosumergibles.

```

import pandas as pd

# =====
# 1. Cargar el dataset final
# =====
df = pd.read_csv("DataSet_BombaESP_Completo.csv")

print("==== RESUMEN DEL DATASET DataSet_BombaESP.csv ====")
print(f"Número de muestras (filas): {df.shape[0]}")
print(f"Número de variables (columnas): {df.shape[1]}\n")

==== RESUMEN DEL DATASET DataSet_BombaESP.csv ====
Número de muestras (filas): 2628000
Número de variables (columnas): 18

```

Figura 15. Dataset consolidado de bombas ESP.

Para cuantificar de manera más precisa el comportamiento de la bomba electrosumergible y la ocurrencia de fallas, se analizaron la cantidad absoluta y el porcentaje de registros asociados a cada clase de la etiqueta variable *_falla*. La Figura 16 presenta la distribución de los tipos de falla mediante un gráfico de barras, en el cual se observa el predominio de la condición normal frente a las distintas modalidades de falla eléctrica, hidráulica y mecánica/térmica, lo que permitió dimensionar el desbalance inherente del problema antes del modelado predictivo. En la Tabla 18 se detalla la distribución de registros por tipo de falla.

Tabla 18. Distribución de registros por tipo de falla.

Tipo de falla	Cantidad de registros	Porcentaje (%)	Descripción
Normal	1 488 999	56.659	Operación normal de la bomba, sin anomalías detectadas en las variables.
corriente_motor_a	148 009	5.632	Falla asociada a desviaciones anómalas en la corriente del motor.
voltaje_motor_v	129 245	4.918	Falla debida a variaciones o caídas de voltaje en la alimentación.
frecuencia_hz	90 850	3.457	Falla relacionada con frecuencia de operación fuera del rango establecido.
potencia_kw	74 609	2.839	Falla vinculada a consumos de potencia inconsistentes con la carga.
arm_desbalance_pct	107 801	4.102	Falla por desbalance de corriente entre fases del motor.
presion_succion_bar	99 233	3.776	Falla asociada a presiones de succión anómalas.
presion_descarga_bar	67 382	2.564	Falla debida a presiones de descarga fuera del rango operativo.
delta_presion_bar	139 047	5.291	Falla vinculada a cambios anómalos en el diferencial de presión.
caudal_m3_h	117 813	4.483	Falla asociada a variaciones no deseadas en el caudal bombeado.
vibracion_rms_mm_s	60 891	2.317	Falla por incrementos anómalos en la vibración RMS del conjunto.
temperatura_motor_c	104 121	3.962	Falla relacionada con sobre temperaturas en el motor ESP.

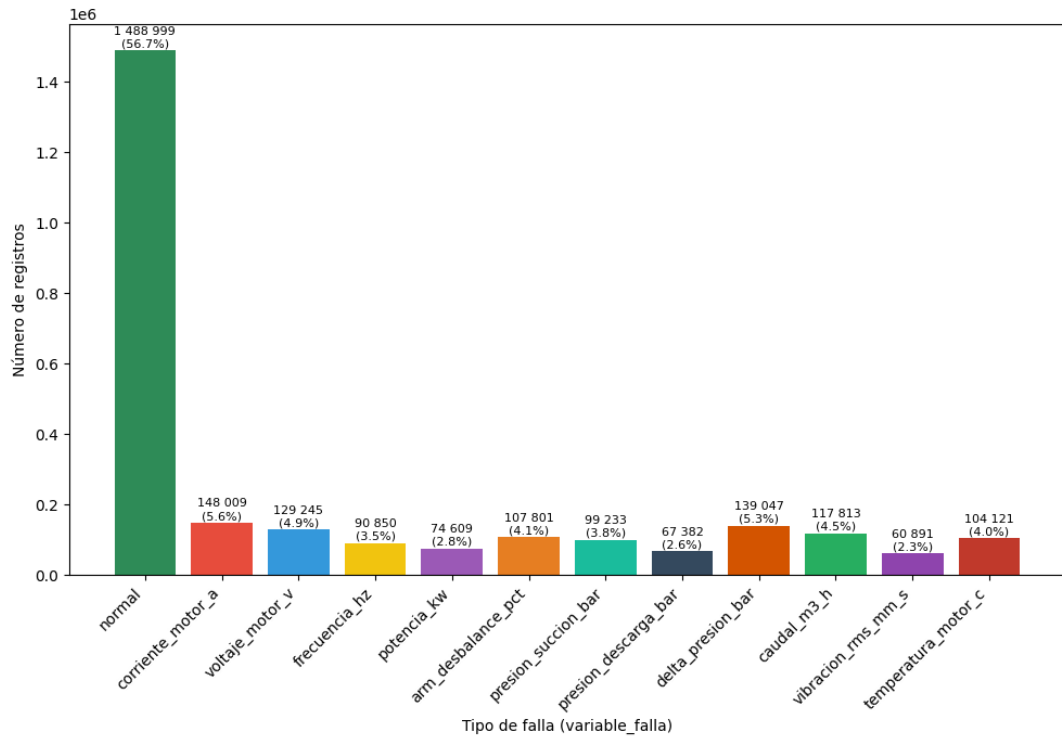


Figura 16. Distribución de registros por tipo de falla.

f. División de Datos

El conjunto de datos `DataSet_BombaESP_Completo.csv` fue particionado en la plataforma Google Colab en tres subconjuntos disjuntos: 70 % para entrenamiento, 15 % para validación y 15 % para pruebas, como se ilustra en la Figura 17. La partición del dataset en entrenamiento, validación y prueba se realizó para evaluar el modelo de forma rigurosa y sin sesgos. El conjunto de entrenamiento se empleó para ajustar el modelo; el de validación, para optimizar la configuración y controlar el sobreajuste; y el de prueba, para estimar de manera objetiva el rendimiento y la capacidad de generalización con datos no vistos, aproximando su comportamiento en condiciones reales.

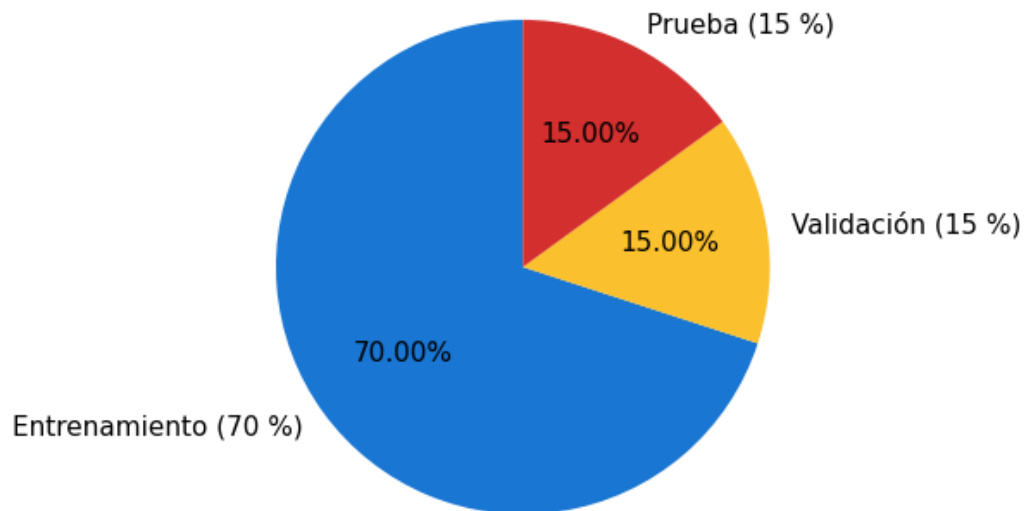


Figura 17. División del dataset para el entrenamiento.

3.1.4 Modelado

Para el proceso de modelado, el entrenamiento de los clasificadores se realizó en el entorno de cómputo en la nube Google Colab en versión PRO, utilizando Python como lenguaje principal y librerías especializadas para ciencia de datos y aprendizaje automático (*pandas*, *NumPy*, *scikit-learn*, *XGBoost*, *LightGBM* y *CatBoost*). El dataset final consolidado se almacenó en archivos con formato (*.csv*), los cuales fueron importados en Colab para aplicar el preprocesamiento y la partición en conjuntos de

entrenamiento, validación y prueba (70 %, 15 % y 15 %), siguiendo el flujo general de creación del dataset, entrenamiento y evaluación que se muestra en la Figura 18.

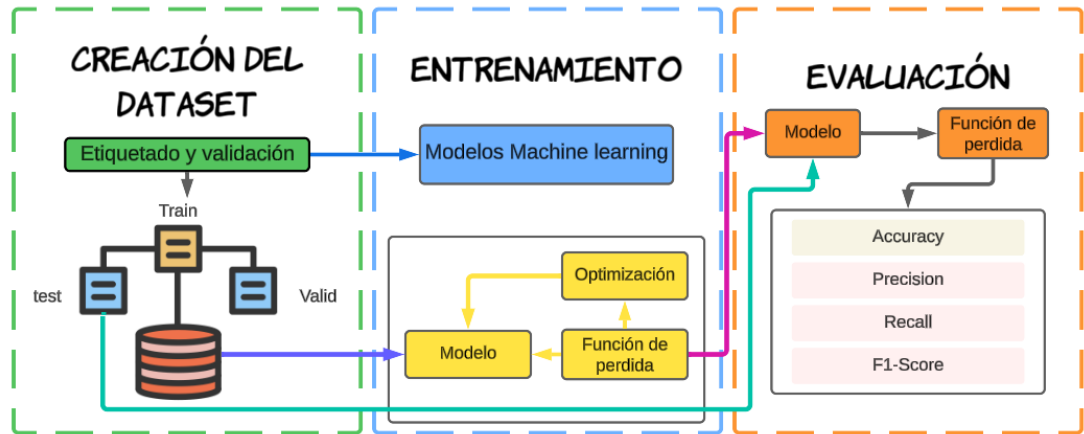


Figura 18. Proceso de entrenamiento de modelos machine learning.

a. Entrenamiento.

En el proceso de entrenamiento se definieron como variables de entrada los parámetros eléctricos, hidráulicos, mecánicos y térmicos de las bombas, que se presentan en la Tabla 16, mientras que como variables de salida se utilizaron las etiquetas de estado y tipo de falla, que se detallan en la Tabla 17 de etiquetado. A partir de una configuración inicial, se ejecutó un proceso sistemático de ajuste de los hiperparámetros más relevantes, que se explican en la Tabla 19, con el propósito de mejorar la capacidad de generalización y mitigar el sobreajuste. En este contexto se entrenaron los modelos Random Forest, XGBoost, LightGBM y CatBoost, que se describen en la Tabla 3. Adicionalmente, se habilitaron rutinas de registro y visualización de métricas de desempeño durante el entrenamiento, lo que permitió monitorear la evolución de indicadores como Accuracy, Precision, Recall y F1-Score sobre los subconjuntos de entrenamiento y validación. Una vez entrenados los modelos de clasificación multisalida, se procedió a su evaluación utilizando las métricas previamente definidas, lo que permitió obtener una visión global y comparativa del desempeño de cada modelo para cada una de las etiquetas de salida consideradas.

Tabla 19. Hiperparámetros configurados y su función en cada modelo.

Función del hiperparámetro	Random Forest (rf)	CatBoost (cat)	LightGBM (lgbm)	XGBoost (xgb)
Cantidad de árboles / iteraciones del ensamble	n_estimators = 100	iterations = 300	n_estimators = 300	n_estimators = 200
Profundidad máxima permitida en los árboles	max_depth = 15	depth = 8	max_depth = -1 (sin límite explícito)	max_depth = 8
Mínimo de muestras en cada hoja de decisión	min_samples_leaf = 5	No aplica	No aplica	No aplica
Tasa de aprendizaje del proceso de boosting	No aplica	learning_rate = 0.1	learning_rate = 0.05	learning_rate = 0.1
Porcentaje de muestras usadas por cada árbol	No aplica	No aplica	subsample = 0.8	subsample = 0.8
Porcentaje de variables usadas por cada árbol	No aplica	No aplica	colsample_bytree = 0.8	colsample_bytree = 0.8
Función objetivo / tipo de problema multiclase	No aplica	loss_function = "MultiClass"	objective = "multiclass"	objective = "multi:softmax"
Número total de clases a predecir	No aplica	No aplica	num_class = int(n_clases max)	num_class = int(n_clases max)
Método interno de construcción de árboles	No aplica	No aplica	No aplica	tree_method = "hist"
Uso de paralelismo en CPU	n_jobs = -1	No aplica	n_jobs = -1	n_jobs = -1
Control de salida / impresión de métricas	No aplica	verbose = False	No aplica	No aplica
Semilla para reproducibilidad de resultados	random_state = 42	random_seed = 42	random_state = 42	random_state = 42

Para cada etiqueta se analizaron de manera individual los *classification reports* y las matrices de confusión generadas, complementando el análisis con las gráficas correspondientes que mostraron el comportamiento del modelo en la predicción de las diferentes clases de falla. Adicionalmente, se registraron el tiempo de entrenamiento, el tiempo de inferencia por etiqueta y el tamaño final de cada modelo, el cual fue almacenado en formato “.pkl” para su futura implementación en el entorno de despliegue. En la Tabla 20 se sintetizan estos resultados, presentando para cada modelo las métricas globales de rendimiento.

Tabla 20. Análisis Comparativo de los Modelos Entrenados.

Modelo	Etiqueta de salida	Accuracy	Precision (w)	Recall (w)	F1-Score (w)	Tiempo de entrenamiento	Tiempo de inferencia por etiqueta	Tamaño del modelo
Random Forest	variable_falla	0.82	0.84	0.82	0.82	17 min	3.2054 s	109.61 MB
	subcomponente_falla	0.83	0.84	0.83	0.83	47.03 s	1.4942 s	
	severidad_falla	0.82	0.80	0.82	0.80		1.8436 s	
	estado_bomba	0.83	0.85	0.83	0.83		1.7602 s	
	falla_predicha_24h	0.91	0.91	0.91	0.91		1.6798 s	
CatBoost	variable_falla	0.77	0.82	0.77	0.78	29 min	1.1585 s	17.19 MB
	subcomponente_falla	0.78	0.82	0.78	0.78	58.17 s	0.4891 s	
	severidad_falla	0.78	0.83	0.78	0.78		0.9140 s	
	estado_bomba	0.79	0.83	0.79	0.79		0.3494 s	
	falla_predicha_24h	0.85	0.87	0.85	0.86		0.3618 s	
LightGBM	variable_falla	0.75	0.80	0.75	0.76	20 min	0.5691 s	60.45 MB
	subcomponente_falla	0.76	0.80	0.76	0.76	30.26 s	0.8425 s	
	severidad_falla	0.76	0.81	0.76	0.76		1.2564 s	
	estado_bomba	0.77	0.81	0.77	0.77		0.9464 s	
	falla_predicha_24h	0.83	0.85	0.83	0.84		0.8495 s	
XGBoost	variable_falla	0.78	0.83	0.78	0.79	18 min	4.2079 s	45.29 MB
	subcomponente_falla	0.79	0.83	0.79	0.79	20.40 s	5.6040 s	
	severidad_falla	0.79	0.84	0.79	0.79		16.0217 s	
	estado_bomba	0.80	0.84	0.80	0.80		15.2443 s	
	falla_predicha_24h	0.86	0.88	0.86	0.87		16.4571 s	

3.1.5 Evaluación

Durante la fase de comparación de modelos, el algoritmo Random Forest se consolidó como la mejor alternativa en términos de desempeño predictivo, tras el proceso de verificación y validación realizado sobre los resultados, que se sintetizan en la Tabla 20. Dicho modelo obtuvo los valores más elevados de accuracy y F1-Score ponderado en las cinco etiquetas de salida consideradas (*variable_falla*, *subcomponente_falla*, *severidad_falla*, *estado_bomba* y *falla_predicha_24h*), superando de manera consistente a los modelos CatBoost, LightGBM y XGBoost.

En la Tabla 21 se presentan las métricas de desempeño del modelo Random Forest para la etiqueta *variable_falla* sobre el conjunto de prueba. La clase 0, que concentró el mayor número de instancias del conjunto de prueba, obtuvo un F1-Score de 0,85, mientras que varias clases asociadas a condiciones de falla (1, 2, 5, 8, 9 y 11) registraron valores de recall superiores a 0.94, lo cual indicó una alta capacidad del modelo para detectar correctamente la mayoría de los eventos de falla, aun a costa de un incremento moderado de falsos positivos en algunas de ellas. En contraste, las clases 3, 4, 7 y 10 presentaron valores de recall cercanos al 0.50, a pesar de mantener precisiones elevadas (≥ 0.85), lo que sugirió que, si bien las predicciones positivas para dichas clases fueron mayoritariamente correctas, el modelo tendió a subestimar su frecuencia real.

Tabla 21. Métricas para la etiqueta *variable_falla*.

Clase	Precisión	Recall	F1-Score	Soporte (n° de muestras)
0 = Normal	0.88	0.83	0.85	253 403
1 = Corriente del motor	0.78	1.00	0.87	25 189
2 = Voltaje del motor	0.70	0.97	0.81	21 804
3 = Frecuencia del VSD	0.88	0.50	0.64	15 481
4 = Potencia eléctrica consumida	0.87	0.50	0.63	12 809
5 = Desbalance armónico/de corriente	0.75	0.97	0.84	18 079
6 = Presión de succión	0.85	0.57	0.68	17 027
7 = Presión de descarga	0.86	0.51	0.64	11 419
8 = ΔP (diferencial de presión)	0.75	0.99	0.86	23 566
9 = Caudal de producción	0.75	0.98	0.85	20 044
10 = Vibración mecánica RMS	0.85	0.50	0.63	10 323
11 = Temperatura del motor	0.67	0.94	0.78	17 616

En la Tabla 22 se presenta el desempeño de la etiqueta *subcomponente_falla*. La clase 0 alcanzó un F1-Score de 0,85, mientras que las clases 1 y 2 registraron valores de (0,80) y (0,83), respectivamente, destacándose esta última por un recall de (0,92), indicativo de una alta capacidad de detección de fallas hidráulicas. Aunque la clase 3 obtuvo un F1-Score ligeramente inferior (0,74), su desempeño se consideró aceptable dado el menor número de muestras disponibles para esta categoría.

Tabla 22. Métricas para la etiqueta *subcomponente_falla*.

Clase	Precisión	Recall	F1-Score	Soporte (n° de muestras)
0 = Normal	0.90	0.80	0.85	253 403
1 = Falla Eléctrica	0.77	0.84	0.80	93 362
2 = Falla Hidráulica	0.76	0.92	0.83	72 056
3 = Falla Mecánica / Térmica	0.69	0.80	0.74	27 939

Por otro lado, para la etiqueta *severidad_falla*, en la Tabla 23 se observa que la clase 3 (falla crítica) destacó, con un F1-Score de 0,94 y un recall de (0,95), lo que evidenció una detección prácticamente completa de las condiciones críticas. La clase 2 (alarma) obtuvo un F1-Score de (0,68), mientras que la clase 1 (alerta temprana) presentó un F1-Score menor (0,38), lo que reflejó una capacidad reducida del modelo para identificar este nivel incipiente de severidad.

Tabla 23. Métricas para la etiqueta *severidad_falla*.

Clase (descripción)	Precisión	Recall	F1-Score	Soporte (n° de muestras)
0 = Normal	0.82	0.88	0.85	253 403
1 = Alerta temprana	0.65	0.26	0.38	48 357
2 = Alarma	0.69	0.67	0.68	48 336
3 = Falla crítica	0.89	0.95	0.94	96 664

Además, en la Tabla 24 se observa que, para la etiqueta *estado_bomba*, la clase 0 (normal) presentó un F1-Score de (0,83), con alta precisión (0,93) y un recall de 0,75, mientras que la clase 1 (falla) obtuvo también un F1-Score de (0,83), con un recall de (0,93), lo que indicó una elevada capacidad para identificar adecuadamente los estados de falla de la bomba electrosumergible.

Tabla 24. Métricas para la etiqueta *estado_bomba*.

Clase	Precisión	Recall	F1-Score	Soporte (n° de muestras)
0 = Normal	0.93	0.75	0.83	253 403
1 = Falla	0.74	0.93	0.83	193 357

Por último, para la etiqueta *falla_predicha_24h*, en la Tabla 25 se observa que la clase 0 (no fallará en 24 h) obtuvo un F1-Score de (0,93), mientras que la clase 1 (fallará en 24 h) registró un F1-Score de (0,86), con un recall de (0,86), lo que indicó una buena capacidad del modelo para anticipar eventos de falla dentro del horizonte de 24 horas, aspecto clave para la planificación de acciones de mantenimiento preventivo.

Tabla 25. Métricas para la etiqueta *falla_predicha_24h*.

Clase	Precisión	Recall	F1-Score	Soporte (n° de muestras)
0 = No fallará en 24h	0.93	0.93	0.93	301 760
1 = Fallará en 24h	0.85	0.86	0.86	145 000

A continuación, en la Tabla 26 se presentan las métricas globales obtenidas por el modelo Random Forest para cada una de las etiquetas de salida, considerando indicadores de desempeño como accuracy, precisión, recall y F1-Score en sus

versiones macro y ponderada. Estos resultados permitieron evaluar de manera comparativa la capacidad del modelo para discriminar la variable de falla dominante, el subcomponente afectado, la severidad del evento, el estado operacional de la bomba y la probabilidad de ocurrencia de una falla en las próximas 24 horas.

Tabla 26. Métricas globales de validación del modelo entrenado.

Métrica	variable_falla	subcomponente_falla	severidad_falla	estado_bomba	falla_predicha_24h
Accuracy	0.82	0.83	0.82	0.83	0.91
Precisión macro	0.80	0.78	0.76	0.84	0.89
Recall macro	0.77	0.84	0.70	0.84	0.89
F1-Score macro	0.76	0.81	0.71	0.83	0.89
Precisión ponderada	0.84	0.84	0.80	0.85	0.91
Recall ponderado	0.82	0.83	0.82	0.83	0.91
F1-Score ponderado	0.82	0.83	0.80	0.83	0.91
Tiempo de inferencia (s)	3.4848	1.8785	1.7987	1.4765	1.4839

En la Figura 19 se presenta la matriz de confusión para la etiqueta *variable_falla*, mientras que la Figura 20 corresponde a la etiqueta *subcomponente_falla*, la Figura 21 se observa la matriz asociada a *severidad_falla*, la Figura 22 muestra los resultados para *estado_bomba* y la Figura 23 ilustra la matriz de *falla_predicha_24h*. El análisis de estas matrices permitió interpretar, de forma desagregada por clase, cómo el modelo Random Forest aprendió a separar las categorías durante el entrenamiento, identificando patrones de acierto y de error que no se evidencian únicamente con métricas globales. En todos los casos se observó que la diagonal principal concentró la mayor parte de los conteos, lo que indicó un número elevado de verdaderos positivos y confirmó la capacidad del modelo Random Forest para distinguir correctamente entre las clases normales y los distintos tipos de falla. Por su parte, los elementos fuera de la diagonal reflejaron los falsos positivos y falsos negativos, evidenciando, por ejemplo, ciertas confusiones entre modos de falla eléctricos, hidráulicos y mecánicos, así como entre niveles de severidad baja y estados normales o de alarma.

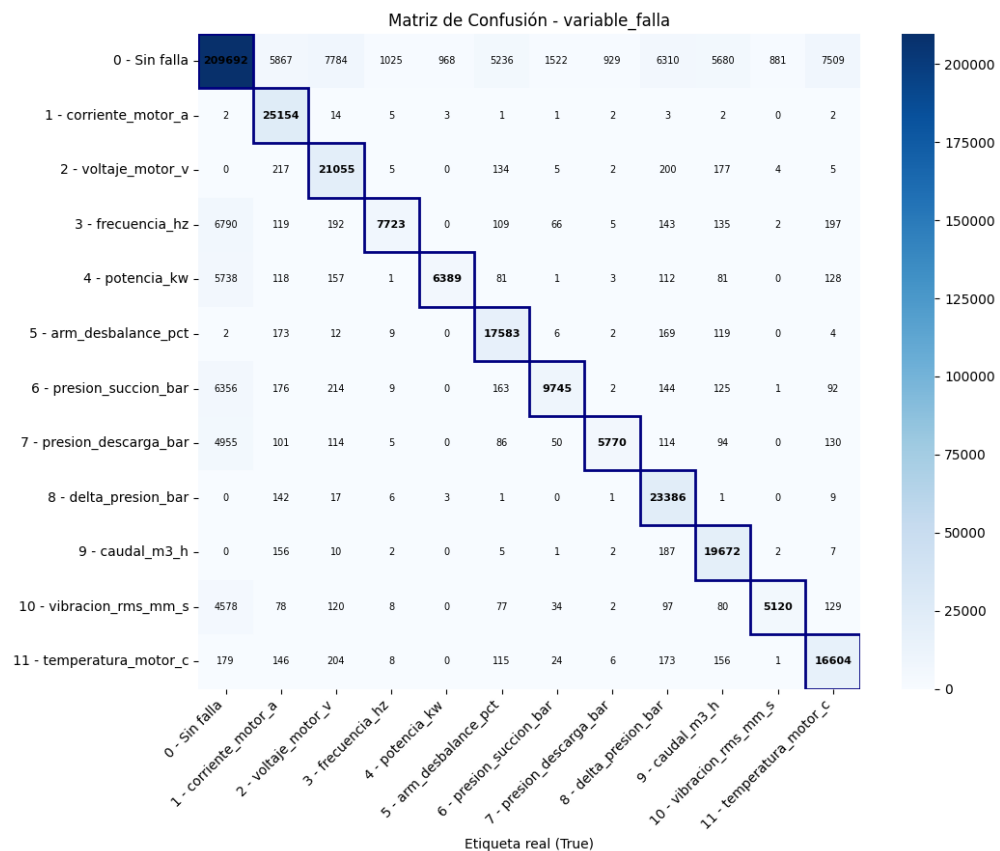


Figura 19. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida variable_falla.

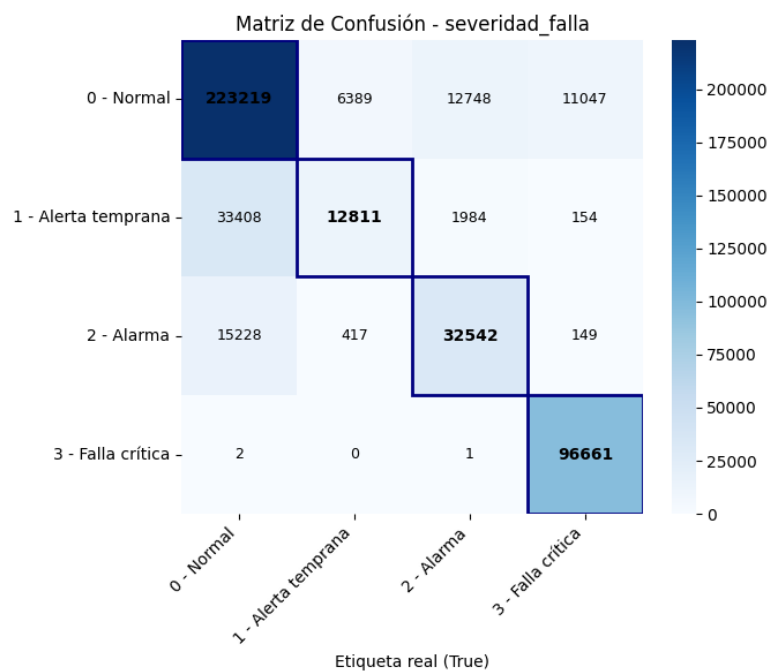


Figura 20. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida subcomponente_falla.

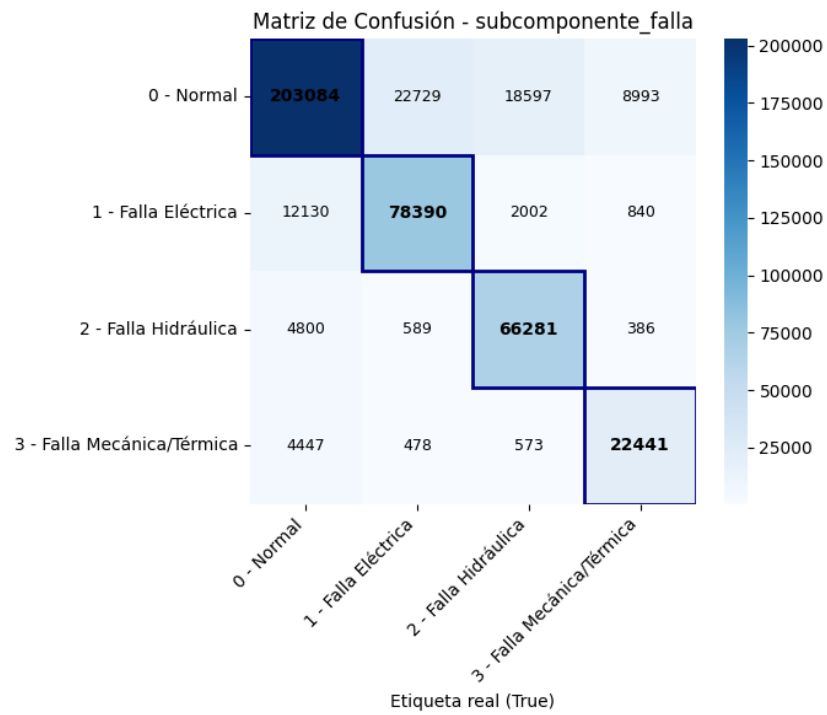


Figura 21. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida `severidad_falla`.

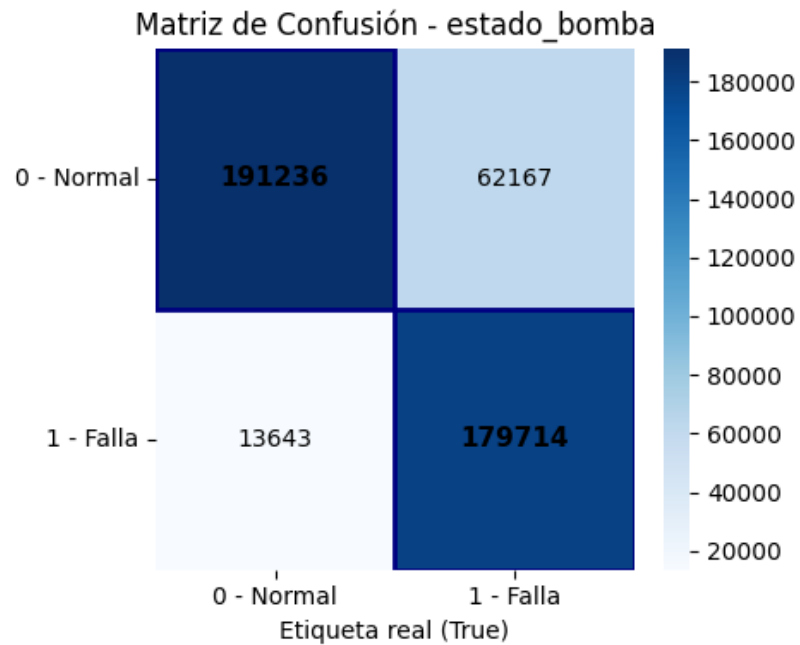


Figura 22. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida `estado_bomba`.

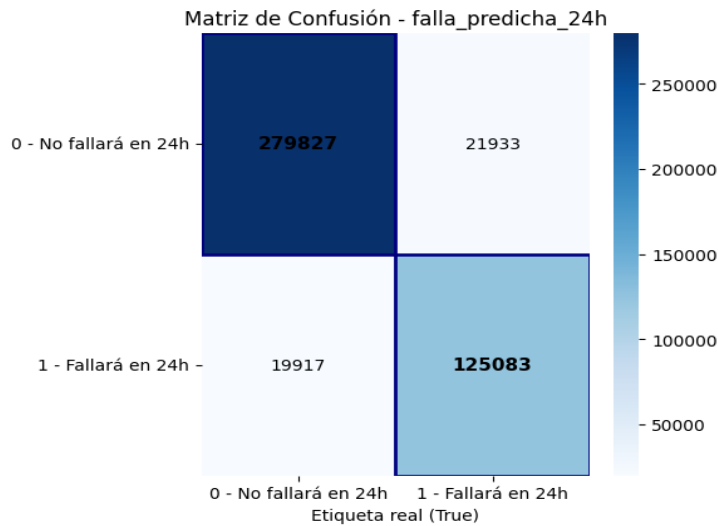


Figura 23. Matriz de confusión del modelo Random Forest respecto a la etiqueta de salida `falla_predicha_24h`.

Con el fin de complementar el análisis basado en matrices de confusión, se generaron curvas Precision–Recall y F1–Recall para cada una de las etiquetas de salida del modelo Random Forest. Las curvas Precision–Recall por clase bajo un enfoque *one-vs-rest*, utilizando como variable de decisión las probabilidades estimadas por el clasificador sobre el conjunto de prueba. Estas curvas permitieron visualizar el compromiso entre precisión y sensibilidad al variar el umbral de decisión, evidenciando que, para la mayoría de las clases asociadas a condiciones de falla y severidad elevada, fue posible mantener niveles altos de precisión incluso para valores de recall elevados, lo que resultó crítico en aplicaciones de monitoreo de integridad de equipos. En la Figura 24 se presenta el conjunto de curvas Precision–Recall por clase para `variable_falla`, cuyo análisis permitió comparar qué tipo de falla (eléctrica, hidráulica, mecánica/térmica o normal) el modelo separó con mayor claridad; se observó que las clases con curvas más próximas a valores altos de precisión y recall evidenciaron una mejor discriminación, mientras que aquellas que descendieron rápidamente al incrementarse el recall sugirieron mayor presencia de falsos positivos o solapamiento entre patrones. Para subcomponente_falla, en la Figura 25 se ilustran las curvas correspondientes y, al tratarse de una etiqueta más granular, se identificó qué subcomponentes presentaron patrones suficientemente distintivos y cuáles tendieron a confundirse, lo que sustentó posibles ajustes posteriores como enriquecimiento de variables, balanceo adicional o calibración de hiperparámetros. En

cuanto a severidad_falla, la Figura 26 recoge los resultados obtenidos, donde las curvas permitieron evaluar la separación entre niveles cercanos y se evidenció que los errores se concentraron principalmente entre categorías adyacentes (por ejemplo, alerta temprana frente a normal, o alarma frente a crítica), facilitando la elección de umbrales más conservadores para severidades críticas. Por su parte, la Figura 27 corresponde a falla_predicha_24h y, al ser una tarea binaria orientada al pronóstico, el análisis permitió valorar el compromiso entre anticipar fallas (mayor recall) y evitar alarmas innecesarias (mayor precision), complementándolo con la relación F1–Recall para seleccionar un umbral operativo acorde al costo relativo de falsos positivos y falsos negativos en mantenimiento. Finalmente, para estado_bomba, la Figura 28 resume el comportamiento del modelo bajo distintos umbrales, reflejando su capacidad para discriminar el estado general (normal/falla) y permitiendo justificar la adopción de un criterio más estricto o sensible según el objetivo del sistema, priorizando alerta temprana o la reducción de falsas alarmas.

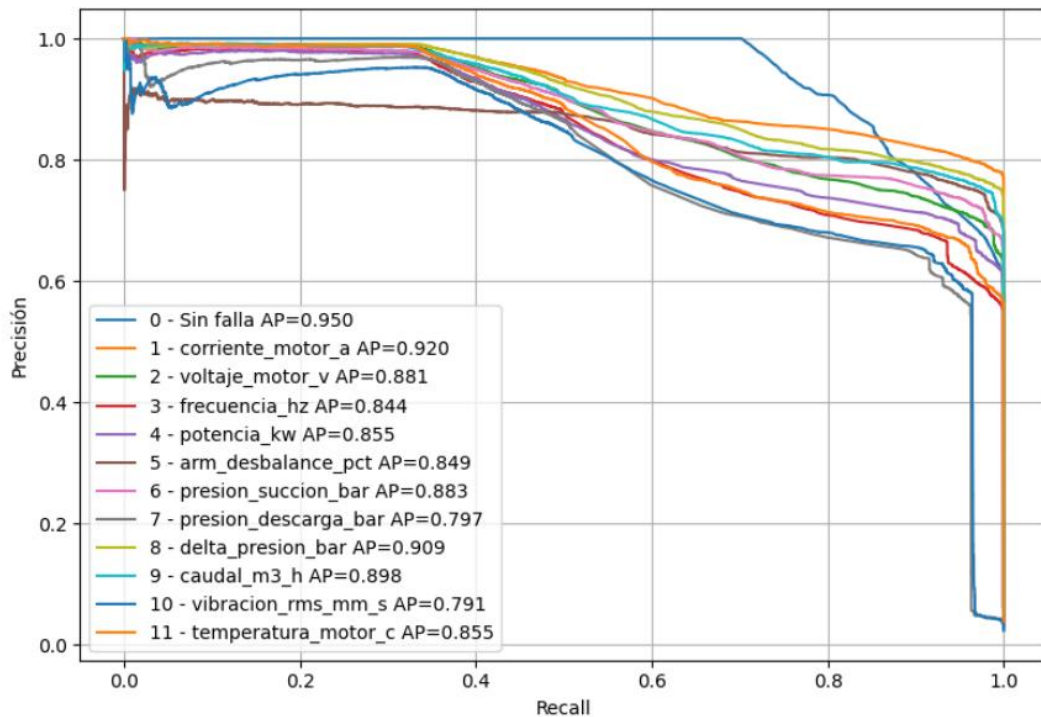


Figura 24. Curvas de confianza entre Precision–Recall variable falla.

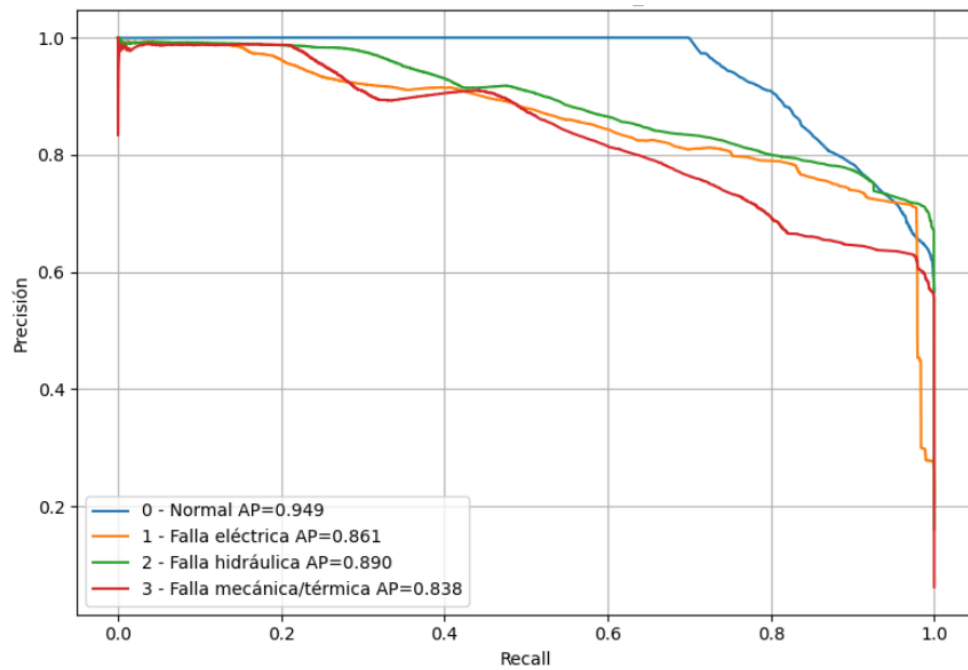


Figura 25. Curvas de confianza entre Precision–Recall subcomponente falla.

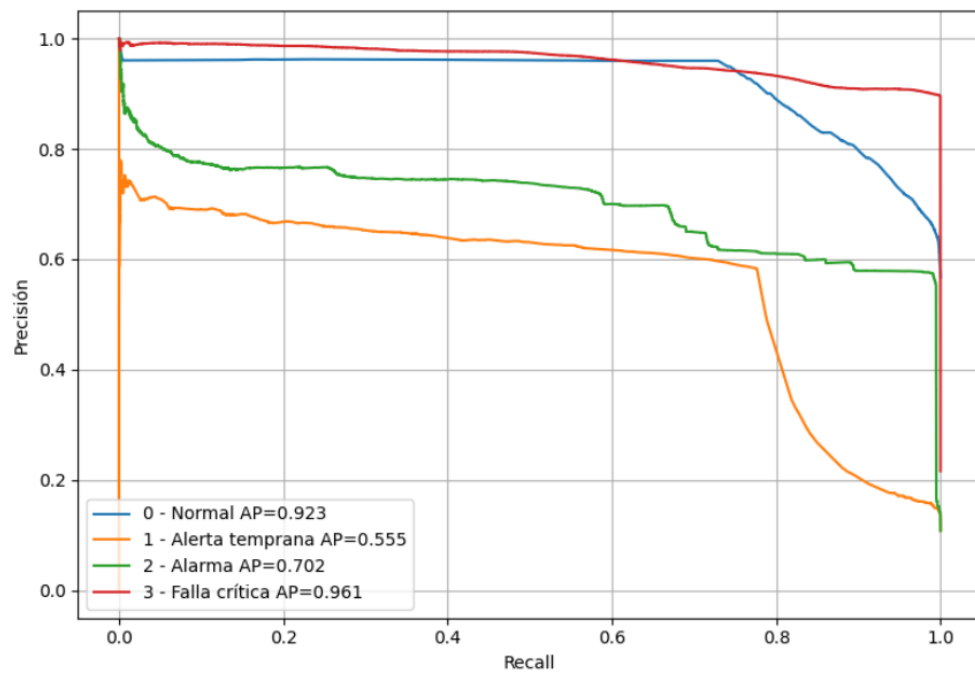


Figura 26. Curvas de confianza entre Precision–Recall severidad falla.

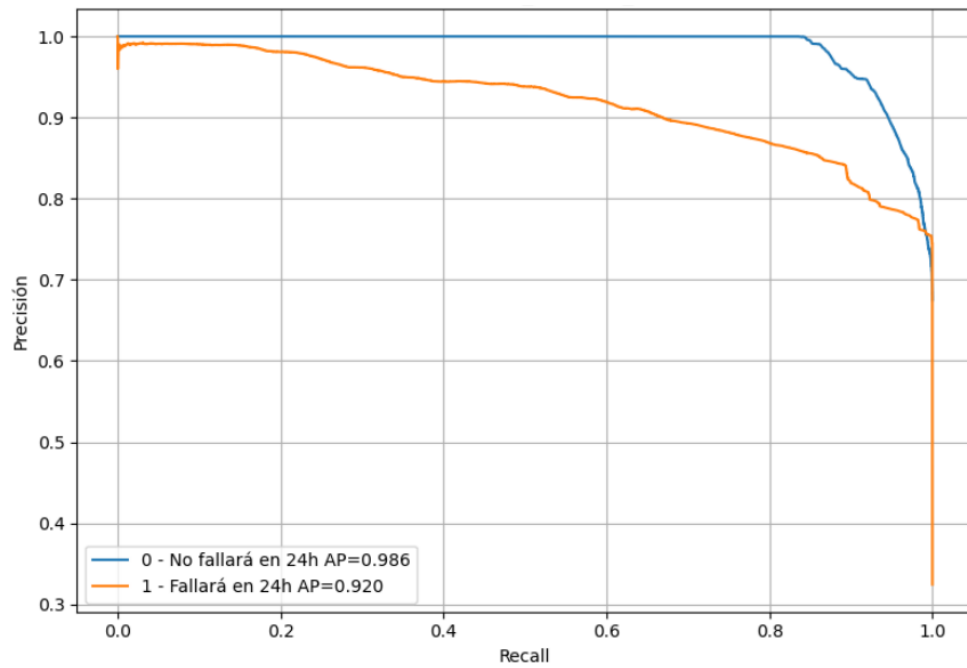


Figura 27. Curvas de confianza entre Precision–Recall de falla en 24 horas.

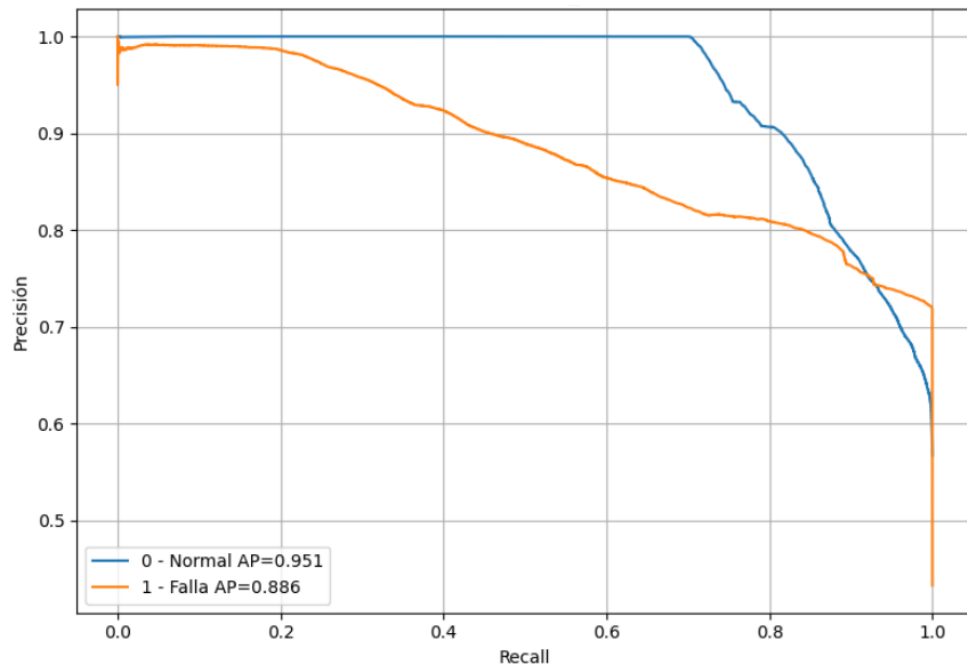


Figura 28. Curvas de confianza entre Precision–Recall del estado de la bomba.

3.1.6 Despliegue

Una vez finalizada la etapa de evaluación, en la que se seleccionó la configuración definitiva del modelo de clasificación multietiqueta para el diagnóstico de las bombas electrosumergibles, se procedió a exportar el modelo entrenado a un archivo serializado (.pkl), dando por concluida la fase de desarrollo y validación offline del modelo predictivo. A partir de este componente, se llevaron a cabo las preparaciones necesarias para su integración en la arquitectura IoT propuesta para el monitoreo del estado de las bombas ESP. Para ello, se definió una estructura modular de despliegue en la que el modelo se encapsuló dentro de una API de inferencia y se integró con los servicios de almacenamiento de series de tiempo (InfluxDB) y visualización (Grafana). En la Figura 29 se ilustra el flujo de utilización del modelo dentro del módulo de procesamiento, el cual constituyó la base del sistema de diagnóstico y predicción implementado.



Figura 29. Proceso de entrenamiento del modelo Random Forest.

a. Implementación de la API de inferencia

Para habilitar el proceso de inferencia sobre el modelo de clasificación multietiqueta, se implementó una API utilizando el framework FastAPI en el archivo main.py. En primer lugar, se cargó el modelo entrenado desde el archivo serializado modelo_bombasESP_RF.pkl mediante la librería joblib, y se leyó el dataset histórico que se reservó para las pruebas, DataSet_BombaESP_Test.csv, el cual contenía los registros de operación de 25 bombas electrosumergibles. Sobre este conjunto de datos

se ejecutaron filtros básicos para conservar únicamente las bombas con identificadores entre 1 y 25, ordenar los registros por timestamp y bomba_id, y descartar filas con valores faltantes en las variables de sensores, garantizando así la coherencia de la información utilizada durante la inferencia.

A continuación, se definió la aplicación FastAPI con un endpoint de prueba ("/") orientado a verificar el estado del servicio y un endpoint principal de predicción ("/predict_25"). El comportamiento de este último se apoyó en una función auxiliar que recorrió secuencialmente los instantes de tiempo únicos presentes en el CSV y, para cada marca temporal, extrajo un lote de lecturas correspondiente a las 25 bombas. Sobre dicho lote se generaron variables temporales derivadas (año, mes, día, hora, minuto, día de la semana, día del año y número de semana) y se construyó la matriz de características X respetando exactamente el orden de columnas utilizado durante el entrenamiento del modelo, validando previamente que todas las columnas requeridas estuvieran presentes en el DataFrame.

Una vez preparada la matriz de entrada, la API invocó el modelo para obtener las predicciones de las cinco etiquetas de salida. Para cada bomba se conformó una estructura de respuesta que incluyó: el identificador bomba_id, el timestamp original, un bloque con las variables de sensores en formato numérico, otro bloque con las predicciones del modelo (*variable_falla*, *subcomponente_falla*, *severidad_falla*, *estado_bomba*, *falla_predicha_24h*) y, adicionalmente, las etiquetas reales provenientes del dataset como referencia. Finalmente, el endpoint "/predict_25" devolvió un objeto JSON que agrupó los resultados para las 25 bombas, junto con la marca temporal de la petición y un manejo básico de errores a través de excepciones HTTP, lo que permitió simular un flujo de inferencia en línea a partir de un CSV histórico de operación. En la Figura 30 se presenta el diagrama de flujo del servicio de inferencia de la API, donde se detalla el proceso de carga del modelo, lectura del dataset histórico y generación de predicciones por bomba.

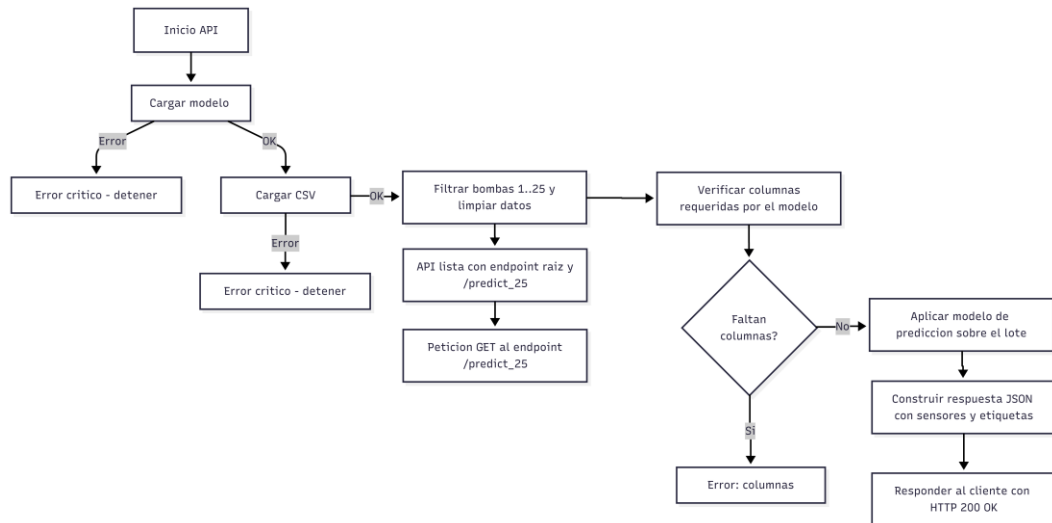


Figura 30. Estructura del servicio de inferencia de la API.

En complemento a la API de inferencia, se implementó un cliente externo encargado de consumir periódicamente el endpoint `/predict_25` e integrar las predicciones con la base de datos de series de tiempo y el sistema de alertas. Este cliente, desarrollado en el script `cliente_esp_influx.py`, se configuró para ejecutarse dentro del mismo entorno contenerizado, utilizando variables de entorno para resolver las direcciones internas de los servicios (*API_URL* apuntando al contenedor *python-app* e *INFLUX_URL* al contenedor *influxdb*), así como el token, la organización y el *bucket* de InfluxDB. Al iniciar, el cliente estableció la conexión con InfluxDB mediante la librería oficial *influxdb_client*, creó la interfaz de escritura síncrona (*write_api*) y entró en un bucle de verificación hasta que la API de predicción respondiera correctamente, garantizando que las consultas solo se realizaran cuando el servicio de inferencia estuviera operativo.

Una vez establecida la comunicación, el cliente ejecutó un bucle principal con un intervalo configurable, en el que solicitó una nueva predicción al *endpoint* `/predict_25`. La respuesta JSON, que contenía el estado de las 25 bombas para un instante de tiempo, se procesó para construir puntos de medición en InfluxDB: por cada bomba se generó un Point con la medición *esp_monitoreo*, un tag *bomba_id* y campos asociados tanto a las variables de sensores como a las etiquetas de salida del modelo, diferenciando entre predicciones (**_pred*) y etiquetas reales (**_real*). En la Figura 31 se muestra el diagrama de flujo del cliente de monitoreo ESP, encargado de consultar periódicamente la API y registrar las predicciones en InfluxDB.

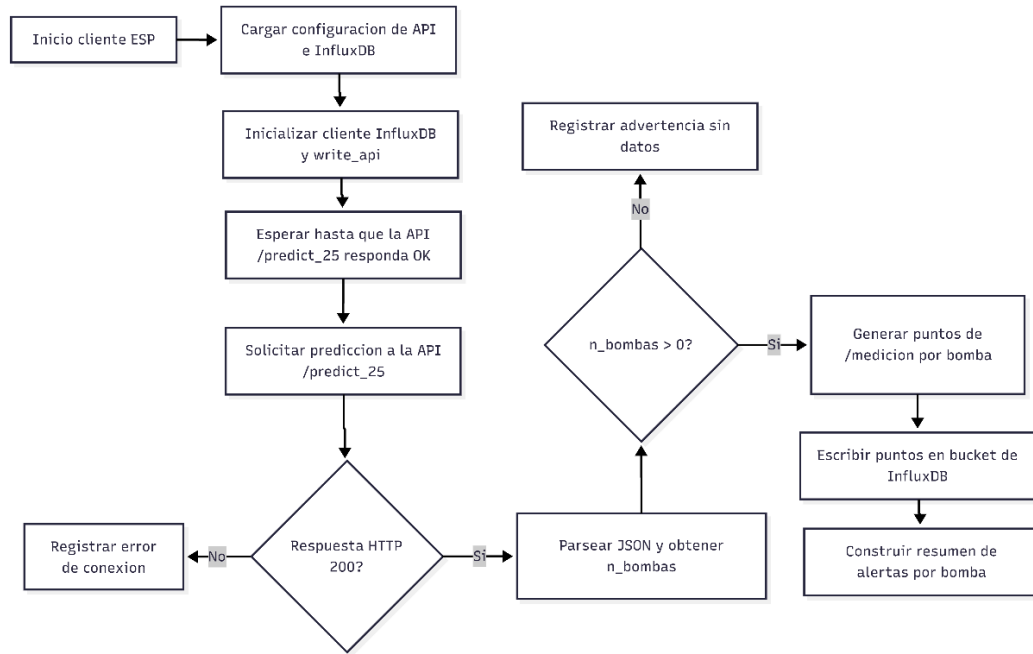


Figura 31. Arquitectura del cliente y registro en InfluxDB.

b. Estructura del directorio para la contenerización

Para la fase de despliegue se estructuró el directorio raíz IoT_ESP como contenedor lógico de todos los componentes necesarios para la contenerización de la solución. En este directorio se definieron las carpetas influxdb y grafana, destinadas a almacenar de forma persistente los datos, la configuración interna y los volúmenes asociados a la base de datos de series de tiempo y a la plataforma de visualización, respectivamente, evitando la pérdida de información al recrear los contenedores. El directorio python-app agrupó todo el código de aplicación, el modelo y los artefactos de construcción de la imagen Docker, concentrando en un único módulo la lógica de inferencia, el cliente de comunicación y las dependencias del entorno de ejecución.

Dentro de python-app se incluyó el script main.py, en el que se implementó la API de inferencia, la carga del modelo serializado y la simulación de lecturas históricas a partir del CSV. El archivo cliente_esp_influx.py se utilizó como cliente de integración, responsable de consumir periódicamente la API, transformar las respuestas en puntos de medición y escribirlos en InfluxDB, además de generar y enviar notificaciones por correo electrónico. El archivo Dockerfile definió la construcción de la imagen del servicio Python, especificando la imagen base, la instalación de las librerías requeridas y la copia de los scripts y del modelo al contenedor. El fichero requirements.txt

consolidó las dependencias de Python (FastAPI, InfluxDB Client, Requests, entre otras), asegurando que el entorno de ejecución dentro del contenedor replicara el utilizado durante el desarrollo. El archivo `DataSet_BombaESP.csv` almacenó el histórico de operación de las bombas, empleado para la simulación de telemetría en la fase de pruebas, mientras que `modelo_bombasESP_RF.pkl` correspondió al modelo Random Forest multietiqueta previamente entrenado y exportado para su uso exclusivo en inferencia.

En la raíz del directorio `IoT_ESP` se ubicó el archivo `docker-compose.yml`, encargado de orquestar los servicios `python-app`, `influxdb` y `grafana`, definiendo imágenes, puertos, redes internas y volúmenes persistentes. Finalmente, el archivo `.env` centralizó las variables de entorno sensibles y parámetros de configuración (URL de servicios, tokens, organización y bucket de InfluxDB), facilitando la gestión del despliegue en distintos entornos sin modificar el código fuente. La organización descrita se resume gráficamente en la Figura 32, donde se representa la estructura del directorio `IoT_ESP` y la relación entre sus carpetas y archivos principales.

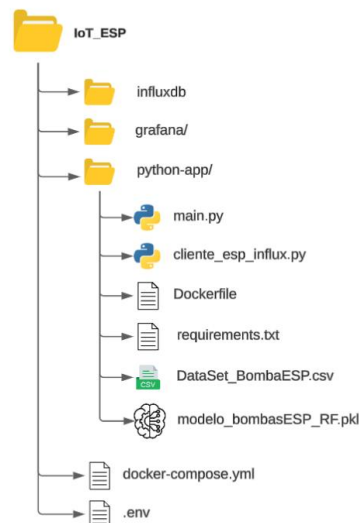


Figura 32. Estructura del directorio `IoT_ESP` para la contenerización del sistema monitoreo predictivo.

c. Despliegue de servicios con Docker Compose

Para iniciar el proceso de despliegue se accedió, desde la consola del sistema, al directorio de trabajo donde se localizó la carpeta del proyecto `IoT_ESP`, la cual contenía el archivo `docker-compose.yml`, el fichero `(.env)` y las subcarpetas `python-`

app, influxdb y grafana. En la Figura 33 se muestra la ubicación que se estableció como ruta base para la ejecución de los comandos de Docker, de modo que el orquestador pudiera resolver correctamente las rutas relativas de los servicios, montar los volúmenes definidos y acceder a todos los artefactos necesarios para la construcción y puesta en marcha de los contenedores.

```
PS C:\IoT_ESP> ls

Directorio: C:\IoT_ESP

Mode                LastWriteTime         Length Name
----                -
d-----          30/11/2025    23:35          grafana
d-----          30/11/2025    22:53        influxdb
d-----          30/11/2025    22:52      python-app
-a-----         1/12/2025     10:38         797 .env
-a-----         1/12/2025     10:38        928 docker-compose.yml

PS C:\IoT_ESP>
```

Figura 33. Selección de la ruta previa al despliegue.

Posteriormente, se procedió a la construcción de las imágenes de los servicios definidos en el archivo *docker-compose.yml*, con el objetivo de generar los contenedores requeridos para la API de inferencia, InfluxDB y Grafana. Para ello, desde el directorio *IoT_ESP* se ejecutó el comando *docker compose build*, mediante el cual Docker leyó las instrucciones especificadas en el Dockerfile del servicio *python-app*, descargó las imágenes base necesarias, instaló las dependencias listadas en *requirements.txt* y empaquetó el código de la API, el cliente de integración y el modelo serializado en una imagen reutilizable, tal como se observa en la Figura 34.

```
#6 [1/5] FROM docker.io/library/python:3.11-slim@sha256:e4676722fba839e2e5cdb844a52262b43e90e56dbd55b7ad953ee3615ad7534f
#6 resolve docker.io/library/python:3.11-slim@sha256:e4676722fba839e2e5cdb844a52262b43e90e56dbd55b7ad953ee3615ad7534f 0.0s done
#6 DONE 0.0s

#7 [2/5] WORKDIR /app
#7 CACHED

#8 [3/5] COPY requirements.txt .
#8 CACHED

#9 [4/5] RUN pip install --no-cache-dir -r requirements.txt
#9 CACHED

#10 [5/5] COPY rf_model.pkl PdM_telemetry.csv stream_rf_sintetico.py ./
#10 CACHED

#11 exporting to image
#11 exporting layers done
#11 exporting manifest sha256:6de1c387f9f6a80cce372ae260763921b6fc9b316e84408386d7566572d49f8f done
#11 exporting config sha256:8ef0c1e292be86b22801bf6f89773260284d8812edf7d13173d4739f6868237b done
#11 exporting attestation manifest sha256:c9eccc38cd2d8d015cca56ef5f78c4dfe36ae097af3dc54b3ed5b55df8b5d15b
#11 exporting manifest sha256:c9eccc38cd2d8d015cca56ef5f78c4dfe36ae097af3dc54b3ed5b55df8b5d15b 0.1s done
#11 exporting manifest list sha256:lad04ad644ab826f0b6be8b665bcb3965dccc9721137dc549f68074fde37cfcea 0.0s done
#11 naming to docker.io/library/iot-python-app:latest done
#11 unpacking to docker.io/library/iot-python-app:latest 0.0s done
#11 DONE 0.2s

#12 resolving provenance for metadata file
#12 DONE 0.0s
[*] Building 1/1
✔iot-python-app Built 0.0s
PS C:\IoT>
```

Figura 34. Construcción de los contenedores.

Para la puesta en marcha de la arquitectura contenerizada, se procedió a levantar los servicios definidos en el archivo *docker-compose.yml* mediante la ejecución del comando *docker compose up -d* desde el directorio IoT_ESP. Esta instrucción inició en segundo plano (*-d*) los contenedores correspondientes a la API de inferencia, a la base de datos InfluxDB y a la plataforma Grafana, creando la red interna definida en la orquestación y asociando los volúmenes persistentes configurados para cada servicio. De este modo, el entorno de ejecución quedó habilitado de forma automática a partir de un único comando, tal como se observa en la Figura 35.

```
PS C:\IoT_ESP> docker compose up -d
time="2025-12-10T06:09:21-05:00" level=warning msg="C:\\IoT_ESP\\docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion"
[+] Running 4/4
✔ Container influxdb Running 0.0s
✔ Container python-app Running 0.0s
✔ Container grafana Running 0.0s
✔ Container cliente-esp Running 0.0s
PS C:\IoT_ESP>
```

Figura 35. Levantamiento de los contenedores del sistema IoT_ESP.

Una vez levantados los servicios, se verificó el estado de la interfaz gráfica de Docker Desktop, en la sección Containers. En la Figura 36 se observa el sistema agrupando los contenedores cliente-esp, grafana, python-app e influxdb, todos en estado activo (*running*). En la parte superior de la pantalla se reportó un uso agregado de CPU de aproximadamente 1,14 % sobre 8 núcleos disponibles (800 %), y un consumo total de memoria de alrededor de 2,47 GB sobre 11,37 GB instalados en el equipo, lo que indicó que la carga del sistema de monitoreo se mantuvo dentro de márgenes aceptables de recurso. Adicionalmente, para cada contenedor se mostró el consumo individual de memoria, los puertos mapeados hacia el host y el tráfico de red, lo que permitió confirmar que la API de inferencia (*python-app*), la base de datos InfluxDB, la plataforma Grafana y el contenedor cliente-esp se encontraban operativos.

Container CPU usage 1

1.14% / 800% (8 CPUs available)

Container memory usage 1

2.47GB / 11.37GB

Show charts

Q Search

Only show running containers

	Name	Container ID	Image	Port(s)	Memory usage...	Memory (%)	Disk read/write	Network I/O	PIDS	Actions
▼	<div><div></div><div>iot_esp</div><div></div></div>	-	-	-	2.47GB / 46.56Gi	21.25%	8.49MB / 0B	613.28KB / 929.5	8, 24, 39, 31	<div><div></div><div></div><div></div></div>
	<div><div></div><div>cliente-esp</div><div></div></div>	651951eea6ff	iot_esp-clie		40.65MB / 11.64Gi	0.34%	0B / 0B	41.7KB / 26.1KB	8	<div><div></div><div></div><div></div></div>
	<div><div></div><div>grafana</div><div></div></div>	ec637b705f31	grafana/gr 3000:3000		98.22MB / 11.64Gi	0.82%	6.51MB / 0B	382KB / 762KB	24	<div><div></div><div></div><div></div></div>
	<div><div></div><div>python-app</div><div></div></div>	fdc003e32ea1	iot_esp-pyt 8000:8000		2.26GB / 11.64Gi	19.44%	77.8KB / 0B	9.58KB / 51.4KB	39	<div><div></div><div></div><div></div></div>
	<div><div></div><div>influxdb</div><div></div></div>	9eb2174f5957	influxdb:2. 8086:8086		77.15MB / 11.64Gi	0.65%	1.9MB / 0B	180KB / 90KB	31	<div><div></div><div></div><div></div></div>

Figura 36. Interfaz de Docker Desktop con los contenedores activos.

d. Configuración del entorno de datos en InfluxDB

Para la configuración del entorno de datos se puso en operación el contenedor de InfluxDB 2.7 definido en el archivo *docker-compose.yml* y, una vez iniciado, se accedió a la interfaz web del servicio a través de <http://localhost:8086>. Mediante el asistente inicial se creó el usuario administrativo y se registró la organización OrgESP junto con el *bucket* de trabajo *bombas_esp*, que actuó como repositorio de todas las series de tiempo generadas por el sistema de monitoreo. En esta misma etapa se generó un token de autenticación, el cual se incorporó en el archivo (*.env*) y fue referenciado por el cliente *cliente_esp_influx.py* a través de las variables INFLUX_URL, INFLUX_TOKEN, INFLUX_ORG e INFLUX_BUCKET, garantizando así una conexión segura y consistente entre el servicio de inferencia y la base de datos de series de tiempo desplegada en Docker.

A nivel de modelado de los datos, se definió la medición *esp_monitoreo* como punto central de almacenamiento, utilizando el identificador de bomba *bomba_id* como etiqueta (*tag*) y registrando como campos (*fields*) tanto las variables de proceso (*corriente, voltaje, presiones, caudal, vibración y temperatura*) como las salidas del modelo de clasificación *multietiqueta*. Cada punto se almacenó utilizando el tiempo real del sistema (*datetime.utcnow()*), de manera que las gráficas y paneles posteriores en Grafana se correspondieran con el instante efectivo de ejecución de la inferencia y no únicamente con el *timestamp* histórico del dataset.

En la Figura 37 se realizan consultas exploratorias desde el módulo Data Explorer de InfluxDB, seleccionando la organización *OrgESP* y el *bucket* *bombas_esp*. Se filtró la medición *esp_monitoreo* y se inspeccionaron las series asociadas a diferentes valores de *bomba_id*, comprobando que, por cada solicitud del cliente, se registraban exactamente veinticinco puntos (*uno por bomba*) con la totalidad de los campos de sensores y etiquetas. Estas consultas se visualizaron tanto en formato tabular como en forma de gráfica de líneas, contrastando los valores de las variables físicas con las salidas de clasificación para distintos instantes de tiempo. De esta manera se confirmó que el script *cliente_esp_influx.py* escribía de forma consistente en la base de datos y que la estructura diseñada para el almacenamiento de las series de tiempo cumplía con los requerimientos del sistema de monitoreo predictivo.



Figura 37. Interfaz de configuración inicial de InfluxDB.

e. Configuración del entorno de visualización en Grafana

Para habilitar el entorno de visualización del sistema IoT, se desplegó una instancia de Grafana como contenedor independiente dentro del mismo *docker-compose.yml* utilizado para la API y para InfluxDB. Una vez levantados los servicios con *docker compose up -d*, se accedió a la interfaz web de Grafana mediante *http://localhost:3000*, donde se completó el asistente inicial de configuración, cambiando las credenciales por defecto y registrando un usuario administrativo para la gestión de paneles y fuentes de datos. Desde el módulo *Data sources* se añadió InfluxDB como origen de información, especificando la URL interna del contenedor (*http://influxdb:8086*), la organización OrgESP, el bucket *bombas_esp* y el token generado previamente en InfluxDB. En la Figura 38 se muestra la configuración de la fuente de datos InfluxDB en Grafana.

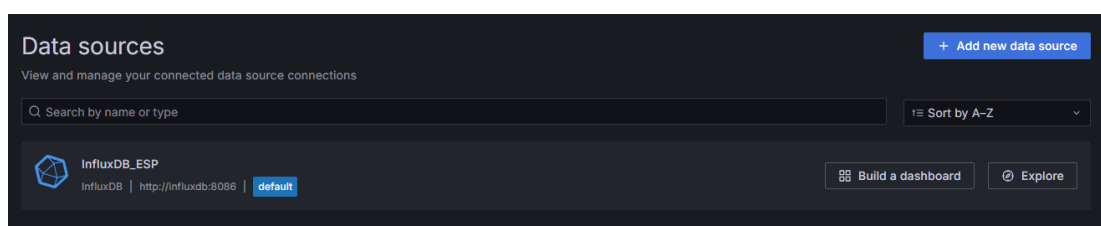


Figura 38. Configuración de la fuente de datos InfluxDB en Grafana.

Sobre esta fuente de datos se construyeron distintos dashboards orientados al monitoreo operativo de las bombas electrosumergibles. En primer lugar, se definieron variables de plantilla para seleccionar dinámicamente el identificador de bomba y el intervalo de tiempo de análisis, lo que permitió reutilizar los mismos paneles para cualquiera de las 25 bombas del sistema. A continuación, se crearon paneles de tipo time series para visualizar la evolución de las variables de proceso, utilizando unidades de medida y límites de visualización coherentes con los rangos de operación definidos en el diseño del sistema. De manera complementaria, se añadieron paneles de tipo stat y table para mostrar las salidas del modelo, aplicando codificación por colores y umbrales sobre los niveles de severidad, de forma que el usuario pudiera identificar de manera inmediata estados críticos y condiciones de riesgo. En la Figura 39 se observa la pantalla principal de diseño del dashboard en Grafana.

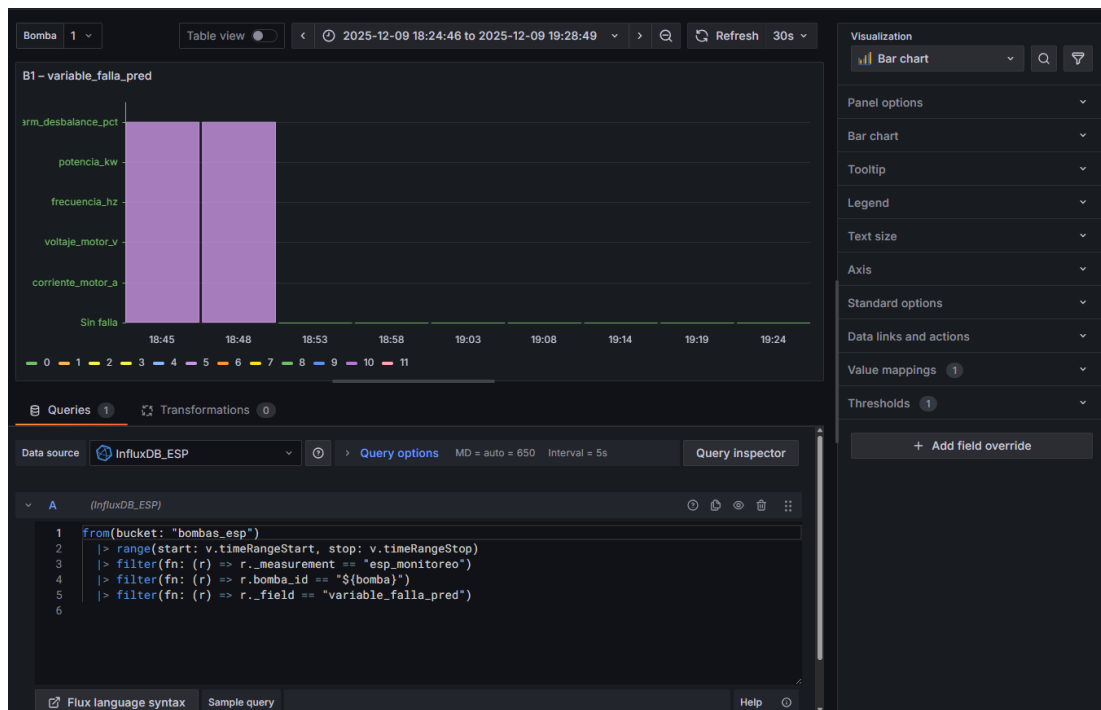


Figura 39. Diseño de dashboard en Grafana.

Finalmente, se integró el mecanismo de alertas por correo electrónico directamente desde Grafana, con el fin de complementar el sistema de notificaciones implementado en el cliente Python. Para ello, se configuró un *contact point* de tipo Email utilizando un servidor SMTP de Gmail, definiendo la dirección del remitente, el puerto seguro y la contraseña de aplicación asociada, y se vinculó este canal a diferentes reglas de

alerta. En la Figura 40 se evidencia la configuración del mecanismo de alertas por correo electrónico en Grafana.

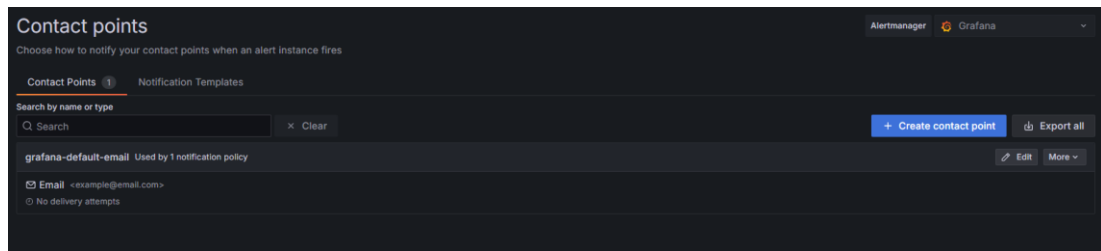


Figura 40. Configuración del canal de notificación por correo electrónico en Grafana.

3.2 Pruebas de funcionamiento del sistema

Durante las pruebas de funcionamiento, el panel de Grafana correspondiente a la Bomba 1 mostró el comportamiento del sistema en un escenario de operación estable, como se aprecia en la Figura 41.

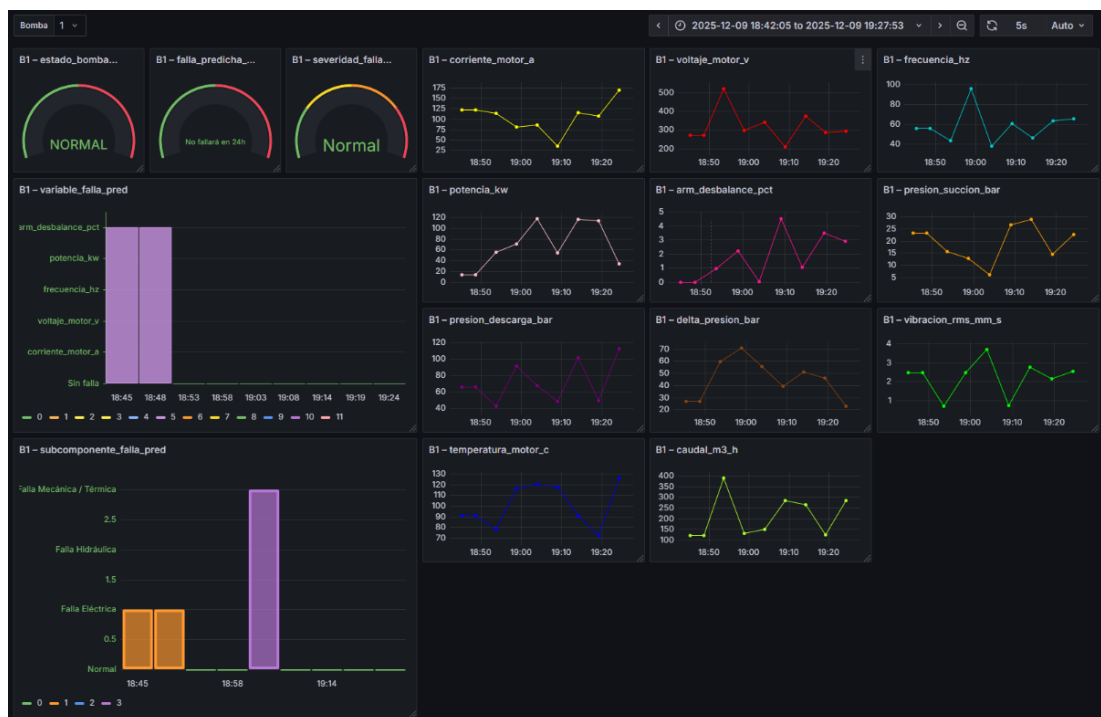


Figura 41. Panel de Grafana para la Bomba 1.

En la última lectura registrada, los indicadores *estado_bomba_pred* y *severidad_falla_pred* se visualizaron en condición NORMAL, mientras que el indicador *falla_predicha_24h_pred* reportó explícitamente que la bomba no fallará en las próximas 24 horas, evidenciando que el modelo de clasificación no detectó

anomalías relevantes en ese intervalo. De forma simultánea, se observó la recepción continua de las variables eléctricas, hidráulicas, mecánicas y térmicas (*corriente, voltaje, presión de succión y descarga, caudal, vibración, temperatura, frecuencia y desbalance*), graficadas en series de tiempo individuales, lo que permitió confirmar que la API de inferencia, la base de datos InfluxDB y el entorno de visualización estaban intercambiando datos de manera consistente durante las pruebas realizadas con un total de 3 575 registros.

En la Bomba 11 se observó un escenario de condición anómala detectada por el modelo. En la Figura 42 se visualiza el indicador *estado_bomba_pred* en estado FALLA, mientras que el gauge de *severidad_falla_pred* se posiciona en nivel de Alarma, evidenciando que el sistema clasificó la situación como un evento de riesgo relevante en el estado actual de la bomba.



Figura 42. Panel de Grafana para la Bomba 11.

No obstante, el indicador *falla_predicha_24h_pred* se mantuvo en la condición de “No fallará en 24h”, lo que reflejó que, aunque el modelo diagnosticó una falla presente, no estimó una progresión hacia una falla total en el horizonte temporal de 24 horas. En los paneles de *variable_falla_pred* y *subcomponente_falla_pred* se identificaron variables específicas y subcomponentes asociados a la anomalía, mientras que las

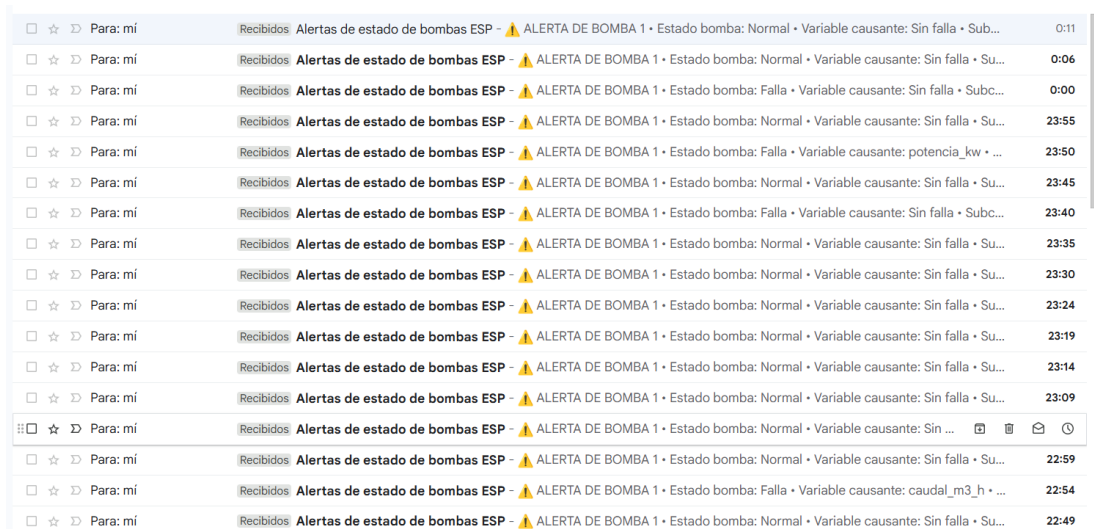
series de tiempo de voltaje, corriente, presiones, caudal, vibración y temperatura mostraron oscilaciones y desviaciones respecto a los patrones observados en condiciones normales, coherentes con el diagnóstico emitido para esta bomba.

En la Figura 43 de la Bomba 6 se representa el escenario de mayor criticidad identificado durante las pruebas. El indicador *estado_bomba_pred* se mantuvo en condición de FALLA y el gauge de *severidad_falla_pred* se posicionó en Falla crítica, mientras que el indicador *falla_predicha_24h_pred* mostró explícitamente la condición de “Fallará en 24h”, señalando un riesgo inminente de pérdida de servicio. En el panel de *variable_falla_pred* se observó que las variables asociadas a la alimentación eléctrica, especialmente la frecuencia del variador y el voltaje del motor, fueron identificadas como principales causantes de la anomalía, lo cual se corroboró en el panel de *subcomponente_falla_pred*, donde el sistema clasificó el evento dentro de la categoría de Falla Eléctrica. Las series de tiempo de corriente, voltaje, frecuencia, potencia y desbalance mostraron incrementos y oscilaciones significativas respecto a los niveles esperados, coherentes con una condición de sobreexigencia eléctrica que, de mantenerse, podría conducir a una falla funcional de la bomba en el horizonte de 24 horas previsto por el modelo.



Figura 43. Panel de Grafana para la Bomba 6.

Para comprobar el correcto funcionamiento del subsistema de notificaciones, se ejecutó el cliente *cliente_esp_influx.py* con el envío de alertas habilitado y se monitorizó la bandeja de entrada de la cuenta de Gmail configurada como destinataria. Tal como se observa en la Figura 44, se recibieron de forma periódica múltiples correos con el asunto “Alertas de estado de bombas ESP – A ALERTA DE BOMBA X”, donde X correspondió al identificador de la bomba evaluada en cada iteración del bucle. Cada mensaje resumió el estado operativo (Normal o Falla), la variable causante, el subcomponente afectado, el nivel de severidad y la predicción de falla a 24 horas, generados a partir de la respuesta de la API. La llegada continua y ordenada de estas notificaciones confirmó que el cliente logró consumir correctamente el *endpoint* de inferencia, construir el cuerpo del correo con la información de todas las bombas y transmitirlo mediante SMTP seguro hacia el servidor de Gmail, cerrando así el ciclo completo de monitoreo y alerta del sistema IoT.



Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Sub...	0:11
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	0:06
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Falla • Variable causante: Sin falla • Subc...	0:00
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:55
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Falla • Variable causante: potencia_kw • ...	23:50
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:45
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Falla • Variable causante: Sin falla • Subc...	23:40
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:35
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:30
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:24
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:19
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:14
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	23:09
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin ...	
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	22:59
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Falla • Variable causante: caudal_m3_h • ...	22:54
Iconos	Para: mí	Recibidos	Alertas de estado de bombas ESP - ALERTA DE BOMBA 1 • Estado bomba: Normal • Variable causante: Sin falla • Su...	22:49

Figura 44. Registro de correos electrónicos de alerta en Gmail.

3.2.1 Resultados de pruebas de funcionamiento

Para la interpretación de los resultados se procedió a obtener un archivo en formato CSV con las predicciones generadas por el sistema en operación. Durante las pruebas, el script *cliente_esp_influx.py* escribió en InfluxDB, para cada petición a la API, un conjunto de puntos asociados a la medición *esp_monitoreo*, incluyendo como campos

tanto las variables de sensores como las salidas del modelo de clasificación multietiqueta, diferenciando explícitamente entre predicciones (**_pred*) y etiquetas reales (**_real*). Estas series de tiempo quedaron almacenadas en el *bucket bombas_esp* de la organización OrgESP, con el identificador de bomba *bomba_id* como etiqueta (*tag*) y el instante de ejecución como marca temporal (*_time*).

A partir de este repositorio, se utilizó el módulo Data Explorer de InfluxDB para construir una consulta que recuperara únicamente la información relevante para el análisis de funcionamiento del sistema. En dicha consulta se seleccionó el *bucket bombas_esp*, se acotó el rango temporal correspondiente al período de pruebas y se filtró la medición *esp_monitoreo*, restringiendo los campos a las etiquetas de interés: *estado_bomba_pred*, *estado_bomba_real*, *severidad_falla_pred*, *severidad_falla_real*, *subcomponente_falla_pred*, *subcomponente_falla_real*, *variable_falla_pred* y *variable_falla_real*. Posteriormente, se aplicó una operación de pivotado sobre los resultados, de manera que cada combinación de *_time* y *bomba_id* quedara representada en una única fila, y cada etiqueta (*predicha* y *real*) se organizara como una columna independiente. De este modo, se obtuvo una tabla estructurada en la que las columnas principales correspondieron al tiempo de la lectura (*_time*), el identificador de la bomba (*bomba_id*) y las etiquetas de salida del modelo junto con sus valores reales asociados. En la Figura 45 se muestran los resultados de las pruebas del modelo desplegado, registradas en un archivo CSV de predicciones.

```
(venv) PS C:\Users\HP\Downloads\EntModelosML> & C:/Users/HP/Downloads/EntModelosML/venv/Scripts/python.exe c:/Users/HP/Downloads/EntModelosML/Scripts/registro_predicciones.py
=====
RESUMEN DEL ARCHIVO: resultados_todas_bombas_pivot.csv
=====

Primeras filas del DataFrame:
   _time      bomba_id  estado_bomba_pred  ...  subcomponente_falla_real  variable_falla_pred  variable_falla_real
0  2025-12-09T23:45:13.883074Z           1           1  ...                1                5                5
1  2025-12-09T23:45:13.883103Z           2           0  ...                0                0                0
3  2025-12-09T23:45:13.883124Z           4           0  ...                2                0                6
4  2025-12-09T23:45:13.883134Z           5           0  ...                0                0                0

[5 rows x 10 columns]

=====
COLUMNAS DISPONIBLES EN EL DATAFRAME
=====
['_time', 'bomba_id', 'estado_bomba_pred', 'estado_bomba_real', 'severidad_falla_pred', 'severidad_falla_real', 'subcomponente_falla_pred', 'subcomponente_falla_real', 'variable_falla_pred', 'variable_falla_real']

Columnas de predicciones del modelo (_pred):
['estado_bomba_pred', 'severidad_falla_pred', 'subcomponente_falla_pred', 'variable_falla_pred']

Columnas de etiquetas reales (_real):
['estado_bomba_real', 'severidad_falla_real', 'subcomponente_falla_real', 'variable_falla_real']
(venv) PS C:\Users\HP\Downloads\EntModelosML> []
```

Figura 45. Registro de predicciones del modelo desplegado.

3.2.2 Evaluación y validación del modelo

En términos generales, el modelo de clasificación multisalida fue evaluado de manera consistente sobre un mismo conjunto de 3 575 registros. En la salida *variable_falla*, como se observa en la Figura 46, la clase 0 (sin falla) presentó el mejor desempeño, con una alta concentración de aciertos en la diagonal principal de la matriz de confusión, lo que se reflejó en una precisión de 0,9348 y un recall de 0,9495. Esto indicó que el sistema identificó de forma fiable los instantes en los que ninguna variable presentaba condición anómala, con un número reducido de falsos positivos hacia clases de falla.

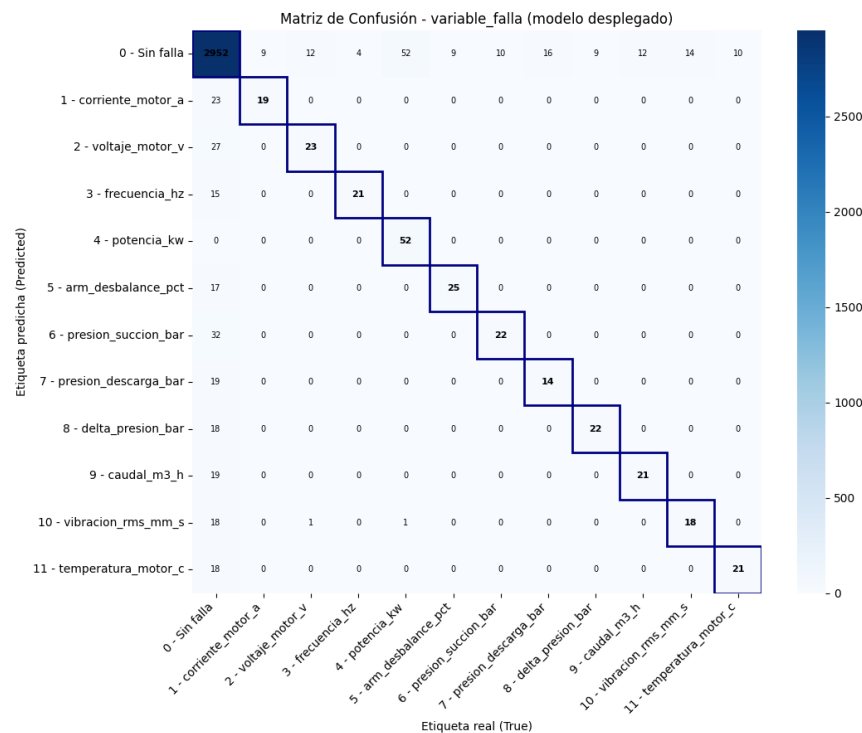


Figura 46. Matriz de confusión de la salida *variable_falla* del modelo de clasificación multietiqueta.

En cambio, las clases de falla específicas (1 a 11) mostraron un rendimiento moderado, condicionado por su menor soporte. Como se evidencia en la Figura 44 y se resumió en la Tabla 27, el modelo alcanzó F1-Scores en el rango aproximado de 0,44–0,69 para la mayoría de las categorías de falla, con casos particulares como la clase 4, donde se observó un recall de 1,0 pero una precisión más baja (0,4952), lo que implicó que el

modelo no omitió fallas de este tipo, aunque generó ciertos falsos positivos hacia dicha categoría.

Tabla 27. Métricas de clasificación por clase para la salida variable_falla

Clase	Precisión	Recall	F1-score	Soporte
0	0,9348	0,9495	0,9421	3109
1	0,6786	0,4524	0,5429	42
2	0,6389	0,4600	0,5349	50
3	0,8400	0,5833	0,6885	36
4	0,4952	1,0000	0,6624	52
5	0,7353	0,5952	0,6579	42
6	0,6875	0,4074	0,5116	54
7	0,4667	0,4242	0,4444	33
8	0,7097	0,5500	0,6197	40
9	0,6364	0,5250	0,5753	40
10	0,5625	0,4737	0,5143	38
11	0,6774	0,5385	0,6000	39

Para el *subcomponente_falla*, descrito en la Figura 47, la clase 0 (condición normal del subcomponente) concentró la mayor parte de las observaciones y presentó un desempeño elevado, con precisión de 0,9343 y recall de 0,9382, lo que evidenció que el sistema discriminó adecuadamente entre estados normales y anómalos a nivel de subcomponente. Tal como se pudo evidenciar en la matriz de confusión, la mayoría de los errores asociados a la clase 0 se distribuyeron hacia las clases de falla, pero su impacto relativo fue bajo frente al volumen total de muestras normales.

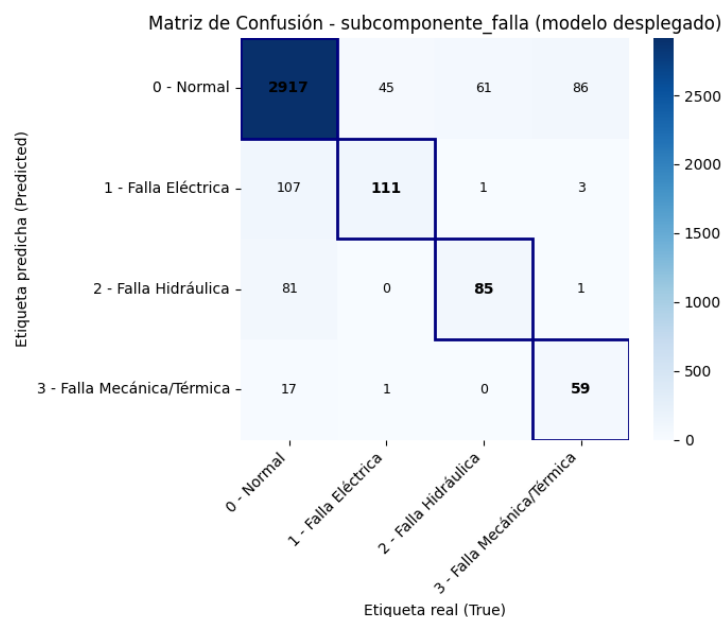


Figura 47. Matriz de confusión de la salida subcomponente_falla del modelo de clasificación multietiqueta.

En las clases de falla (1: eléctrica, 2: hidráulica, 3: mecánica/térmica) el comportamiento fue moderado, con F1-Scores de 0,5858, 0,5414 y 0,5221, respectivamente, tal como se resumió en la Tabla 28. El modelo identificó con mejor desempeño las fallas eléctricas (clase 1), mientras que en la clase 3 se observó un recall alto (0,7662) acompañado de una precisión más baja (0,3960), lo que indicó una mayor sensibilidad para detectar fallas mecánico-térmicas a costa de un número mayor de falsos positivos.

Tabla 28. Métricas de clasificación por clase para la salida subcomponente_falla

Clase	Precisión	Recall	F1-score	Soporte
0	0,9343	0,9382	0,9363	3109
1	0,7070	0,5000	0,5858	222
2	0,5782	0,5090	0,5414	167
3	0,3960	0,7662	0,5221	77

En cambio, en la salida *severidad_falla* detallada en la Figura 48, la clase 0 (condición normal) presentó un desempeño sobresaliente, con precisión de 0,9299 y recall de (0,9601), lo que confirmó que el modelo discriminó de forma consistente los instantes sin anomalías de severidad. De igual manera, la clase 3 (falla crítica) alcanzó un recall de 1,0 y un F1-Score de (0,7993), evidenciando que el sistema no dejó pasar eventos catalogados como críticos, aun cuando incurrió en algunos falsos positivos hacia esta categoría.

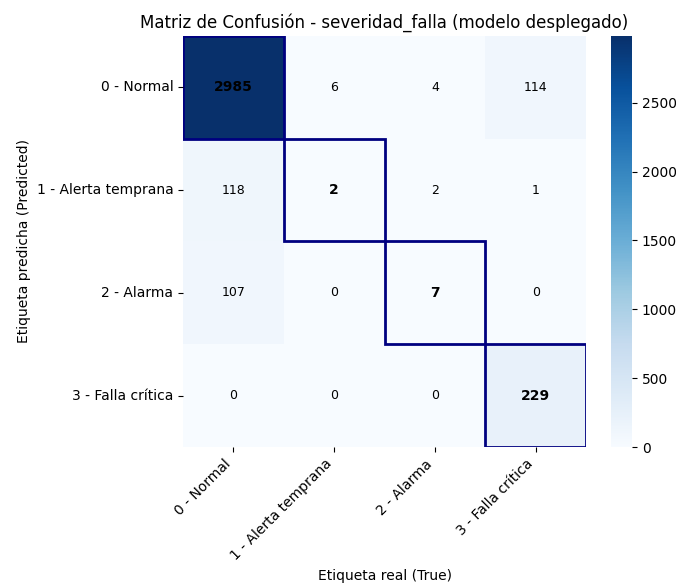


Figura 48. Matriz de confusión de la salida *severidad_falla* del modelo de clasificación multietiqueta.

Por otro lado, como se evidencia en la Figura 48 y se resumió en la Tabla 29, las clases intermedias 1 (alerta temprana) y 2 (alarma) presentaron recall muy reducido (0,0163 y 0,0614, respectivamente), lo que indicó que estos niveles de severidad tendieron a ser confundidos con estados normales o críticos. En consecuencia, el modelo resultó especialmente adecuado para diferenciar entre condiciones normales y fallas críticas, pero ofreció menor capacidad de discriminación fina entre los niveles intermedios de severidad, lo cual fue coherente con el menor soporte y la similitud de patrones entre estas clases.

Tabla 29. Métricas de clasificación por clase para la salida *severidad_falla*.

Clase	Precisión	Recall	F1-score	Soporte
0	0,9299	0,9601	0,9448	3109
1	0,2500	0,0163	0,0305	123
2	0,5385	0,0614	0,1102	114
3	0,6657	1,0000	0,7993	229

En la salida *estado_bomba*, se consideraron dos clases: 0 (bomba en estado normal) y 1 (bomba en estado de falla). Como se observa en la Figura 49, la matriz de confusión mostró una alta concentración de aciertos en la clase 0, con 2 793 verdaderos negativos y únicamente 149 falsos negativos, lo que se reflejó en una precisión de 0,9494 y un recall de 0,8984 para la condición normal. Este comportamiento evidenció que el modelo discriminó de manera consistente los instantes en los que la bomba operó sin anomalías, manteniendo un porcentaje bajo de fallas no detectadas.

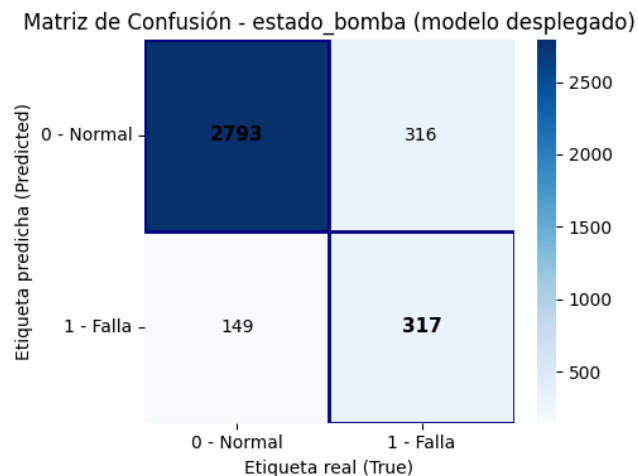


Figura 49. Matriz de confusión de la salida *estado_bomba* del modelo de clasificación multietiqueta.

Para la clase 1 (falla), la matriz de confusión evidenció 317 verdaderos positivos frente a 316 falsos positivos. En términos de métricas, la clase 1 alcanzó una precisión de 0,5008 y un recall de (0,6803), con un F1-Score de (0,5769), tal como se resumió en la Tabla 30. Esto indicó que el modelo priorizó la sensibilidad en la detección de fallas, a costa de generar un número moderado de falsas alarmas.

Tabla 30. Métricas de clasificación por clase para la salida estado_bomba.

Clase	Descripción	Precisión	Recall	F1-score	Soporte
0	Estado normal	0,9494	0,8984	0,9232	3109
1	Estado de falla	0,5008	0,6803	0,5769	466

En la *salida falla_predicha_24h* se consideraron dos clases: 0 (no fallará en 24 h) y 1 (fallará en 24 h). Como se evidencia en la Figura 50, la matriz de confusión mostró 3110 verdaderos negativos y 241 verdaderos positivos, frente a 122 falsos positivos y 102 falsos negativos. Para la clase 0, se obtuvo una precisión de 0,9682 y un recall de (0,9623), lo que evidenció que el modelo identificó de forma muy confiable los instantes en los que no se esperaba una falla en el horizonte de 24 horas.

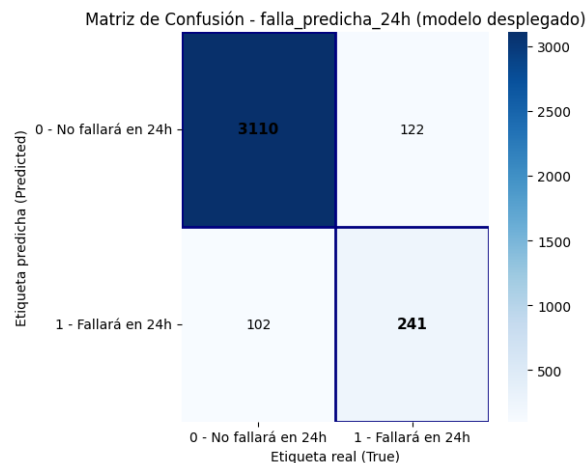


Figura 50. Matriz de confusión de la salida falla_predicha_24h del modelo de clasificación multietiqueta.

En la clase 1 (fallará en 24 h), el modelo alcanzó una precisión de 0,6639 y un recall de (0,7026), con un F1-Score de (0,6827), como se evidencia en la Figura 50 y se resumió en la Tabla 31. Esto implicó que el sistema fue capaz de detectar cerca del 70 % de las fallas que ocurrieron dentro de las siguientes 24 horas, con un nivel moderado de falsas alarmas asociado a esta etiqueta.

Tabla 31. Métricas de clasificación por clase para la salida falla_predicha_24h.

Clase	Descripción	Precisión	Recall	F1-score	Soporte
0	No fallará en 24 h	0,9682	0,9623	0,9652	3232
1	Fallará en 24 h	0,6639	0,7026	0,6827	343

En la Tabla 32 se presentaron de forma resumida los resultados globales obtenidos para las cinco salidas de clasificación del modelo desplegado, considerando los 3 575 registros generados por la arquitectura de inferencia en operación.

Tabla 32. Resumen de métricas globales del modelo desplegado para cada salida de clasificación.

Salida	Tipo de problema	Precisión (w)	Recall (w)	F1-score (w)	Accuracy	Soporte
variable_falla	Multiclase (12 clases)	0,8973	0,8979	0,8946	0,8979	3575
subcomponente_falla	Multiclase (4 clases)	0,8920	0,8873	0,8872	0,8873	3575
severidad_falla	Multiclase (4 clases)	0,8771	0,9015	0,8774	0,9015	3575
estado_bomba	Binaria (0 = normal, 1 = falla)	0,8909	0,8699	0,8780	0,8699	3575
falla_predicha_24h	Binaria (0 = no falla, 1 = falla en 24 h)	0,9390	0,9373	0,9381	0,9373	3575

3.3 Presupuesto

Para el cálculo del presupuesto de mano de obra se consideró la remuneración de un ingeniero en Telecomunicaciones responsable del diseño, desarrollo, implementación y validación del sistema de monitoreo predictivo de bombas ESP. Se asumió un salario mensual referencial de USD 858,00, con una jornada de 21 días laborables al mes y 8 horas diarias de trabajo efectivo. Con estos parámetros se determinó, en primera instancia, la remuneración por día y, posteriormente, la remuneración por hora, empleando la relación general mostrada en la Ecuación (11).

$$\text{Remuneración por día} = \frac{\text{Remuneración por mes}}{\text{Días laborables}} \quad (11)$$

$$\text{Remuneración por día} = \frac{858,00}{21} = 40,86 \text{ USD/día}$$

$$\text{Remuneración por hora} = \frac{\text{Horas de trabajo por día}}{\text{Remuneración por día}}$$

$$\text{Remuneración por hora} = \frac{8}{40,86} = 5,11 \text{ USD/hora}$$

Se consideró que las horas empleadas para el desarrollo del proyecto fueron de 120 horas. Por lo tanto, el costo total estimado de mano de obra asociado al diseño e implementación del sistema asciende a:

$$\text{Presupuesto}_{\text{diseño}} = 5,11 \times 120 = 613,20 \text{ US}$$

Los costos de implementación consideraron los recursos de hardware necesarios para el desarrollo, despliegue y validación del sistema de monitoreo predictivo de bombas ESP, así como los precios referenciales del mercado, impuestos e imprevistos asociados a la puesta en operación del entorno de trabajo. La Tabla 33 resume de forma sintética estos costos de implementación de la solución propuesta.

Tabla 33. Recursos de hardware utilizados para el desarrollo y despliegue del sistema IoT-ESP.

Ítem	Recurso de hardware	Descripción técnica resumida	Cantidad	Costo unitario [USD]	Costo total [USD]
1	Computador portátil para desarrollo	Laptop HP para entrenamiento de modelos, ejecución de Docker Desktop, InfluxDB, Grafana y API de inferencia. CPU multinúcleo, 8–16 GB RAM, SSD.	1	600,00	600,00
2	Almacenamiento externo	Disco duro externo de 1 TB para respaldo de datasets, modelos entrenados y contenedores.	1	80,00	80,00
3	Router / punto de acceso WiFi	Router doméstico para conexión a Internet y pruebas de acceso remoto a la API y paneles web.	1	60,00	60,00
4	Periféricos de apoyo	Teclado, ratón y soporte ergonómico utilizados durante el desarrollo y pruebas del sistema.	1 set	50,00	50,00
Total					790,00 USD

Se obtiene un presupuesto total estimado de:

$$\text{Presupuesto}_{\text{total}} = \text{Presupuesto}_{\text{diseño}} + \text{Presupuesto}_{\text{implementación}}$$

$$\text{Presupuesto}_{\text{total}} = 613,20 + 790,00 = 1403,20 \text{ USD}$$

CAPÍTULO IV. CONCLUSIONES Y RECOMENDACIONES

4.1 Conclusiones

- La aplicación de la metodología CRISP-DM permitió estructurar de manera ordenada el proceso de comprensión del negocio, análisis de datos, preparación de los conjuntos de entrenamiento y evaluación de modelos de machine learning. Este enfoque metodológico facilitó la integración de variables eléctricas, hidráulicas, mecánicas y térmicas en un dataset consolidado, así como la definición de etiquetas multisalida, garantizando la trazabilidad de las decisiones adoptadas en cada fase del ciclo analítico.
- El análisis y la preparación de los datos confirmaron que fue posible construir un conjunto de entrenamiento consistente a partir de variables eléctricas, hidráulicas, mecánicas y térmicas medidas en intervalos regulares de 5 minutos. La consolidación de los archivos DataSet_BombaESP.csv permitió disponer de un dataset estructurado para tareas multietiqueta, en el que se controlaron problemas de valores atípicos, registros incompletos y desbalance entre clases. Estas actividades de preprocesamiento incidieron directamente en la estabilidad y desempeño de los modelos entrenados.
- La comparación de diferentes algoritmos de clasificación evidenció que los modelos de ensamble basados en árboles de decisión son adecuados para el diagnóstico y la predicción de fallas en sistemas ESP. Random Forest, XGBoost, LightGBM y CatBoost mostraron métricas de desempeño competitivas en escenarios multiclase, siendo el modelo de Random Forest multisalida el que ofreció un mejor equilibrio entre exactitud global, robustez ante ruido y facilidad de interpretación. Este comportamiento lo convirtió en el candidato principal para el despliegue dentro de la arquitectura IoT propuesta.
- El estudio detallado de matrices de confusión, métricas ponderadas y curvas Precision–Recall y F1 por clase permitió comprender el comportamiento real de los modelos más allá de un único indicador global. Este análisis mostró la

importancia de tratar de manera explícita el desbalance de clases, en especial en etiquetas sensibles como `severidad_falla`, `estado_bomba` y `falla_predicha_24h`, donde los falsos negativos tienen impacto operativo directo. Gracias a estas evaluaciones, fue posible ajustar umbrales y estrategias de entrenamiento para priorizar la detección temprana de estados anómalos.

- La evaluación del modelo de Random Forest multisalida a partir de los 3 575 registros obtenidos desde la base de datos InfluxDB, generados por la arquitectura de inferencia en operación, evidenció que el esquema de clasificación mantuvo un desempeño robusto en condiciones de despliegue real. En la Tabla 32 se presentan de forma resumida las métricas globales, donde se observó que las salidas multiclase (`variable_falla`, `subcomponente_falla` y `severidad_falla`) alcanzaron valores de precisión, recall y F1 ponderados cercanos o superiores a 0,88, mientras que las salidas binarias `estado_bomba` y `falla_predicha_24h` obtuvieron accuracies de 0,8699 y 0,9373, respectivamente. Estos resultados confirmaron que el modelo de Random Forest desplegado fue capaz de diagnosticar el estado operativo de la bomba y anticipar fallas en un horizonte de 24 horas con altos niveles de exactitud y equilibrio entre falsos positivos y falsos negativos, validando la pertinencia de su integración dentro de la arquitectura IoT propuesta.
- La arquitectura de despliegue basada en contenedores Docker, un servicio de inferencia construido en FastAPI, almacenamiento en InfluxDB y visualización mediante Grafana demostró ser técnicamente viable para el monitoreo de baja latencia de bombas ESP. La integración entre el microservicio de predicción y los paneles de supervisión permitió combinar en un mismo entorno las series temporales de proceso y las salidas de los modelos de machine learning, habilitando la generación de alertas y el seguimiento continuo del estado de la bomba sin interrumpir la operación del sistema.

4.2 Recomendaciones

- Se recomienda mantener, durante la implementación del sistema en entornos reales, la misma estructura de variables de entrada y el flujo de preprocesamiento definido en el desarrollo del proyecto. Conservar el esquema de datos, las validaciones y las transformaciones aplicadas en la etapa de modelado facilita que el sistema de monitoreo y predicción funcione de manera consistente y reduzca errores por incompatibilidades de formato o tipo de dato.
- Se recomienda implementar una estrategia de gestión del ciclo de vida de los modelos dentro del sistema, que incluya versionamiento, registro de cambios y mecanismos para reemplazar de forma controlada los modelos en producción. Esta práctica contribuye a que la plataforma pueda incorporar ajustes o mejoras al modelo seleccionado sin interrumpir el funcionamiento del servicio ni perder trazabilidad sobre qué versión generó cada predicción.
- Con el fin de aprovechar plenamente la arquitectura desarrollada, se sugiere mantener una integración clara entre el servicio de inferencia (FastAPI), la base de datos de series temporales (InfluxDB) y los tableros de Grafana. La definición de rutas de datos bien documentadas, nombres de mediciones estandarizados y paneles organizados por bomba o zona de operación facilita la interpretación de la información y el uso del sistema como herramienta diaria de supervisión.
- Para asegurar la sostenibilidad del sistema a largo plazo, se sugiere establecer políticas de respaldo y monitoreo de la infraestructura basada en contenedores. Incluir copias de seguridad periódicas de la base de datos, seguimiento del uso de recursos y registros de eventos de los contenedores permite actuar de manera preventiva ante fallas técnicas y conservar la disponibilidad del sistema de análisis y monitoreo de las bombas ESP.

REFERENCIAS BIBLIOGRÁFICAS

- [1] U. E. Akpan, O. Okoye, and M. Gervais, “Internet of Things in Petroleum: A Review of Benefits and Challenges,” *Journal of Engineering Research and Reports*, vol. 27, no. 4, pp. 411–425, 2025.
- [2] A. Alshibani, S. M. Alkhathami, M. A. Hassanain, F. Tuffaha, D. Ouis, and A. Mohammed, “Hybrid Framework for Investigating Digital Transformation Barriers in the Oil and Gas Sector,” *Energies 2024, Vol. 17, Page 6151*, vol. 17, no. 23, p. 6151, Dec. 2024, doi: 10.3390/EN17236151.
- [3] S. K. Sharma, A. Rani, H. Bakhariya, R. Kumar, D. Tomar, and S. Ghosh, “The Role of IoT in Optimizing Operations in the Oil and Gas Sector: A Review,” *Transactions of the Indian National Academy of Engineering*, vol. 9, no. 2, pp. 293–312, Jun. 2024, doi: 10.1007/s41403-024-00464-9.
- [4] O. Álvarez Alonso, A. Díaz Echeverría, N. Afonso, A. Sánchez Campos, and C. Bordiu García-Ovies, “Hoja de ruta para la transformación digital del sector energético en América Latina y el Caribe,” Washington, D.C., 2023. [Online]. Available: <https://publications.iadb.org/publications/spanish/document/Hoja-de-ruta-para-la-transformacion-digital-del-sector-energetico-en-America-Latina-y-el-Caribe.pdf>
- [5] C. P. Jaramillo Álvarez and A. Ramos Alvarez, “Inteligencia artificial en la gestión de proyectos petroleros en América Latina: Aplicabilidad en Ecuador,” *InnDev*, vol. 4, no. 2, pp. 1–23, Sep. 2025, doi: 10.69583/inndev.v4n2.2025.162.
- [6] S. Saptadi, A. Widodo, M. F. Athaillah, and M. F. Ayyasyi, “Implementation of machine learning methods in predicting failures in electrical submersible pump machines,” *Multidisciplinary Science Journal*, vol. 7, no. 3, Mar. 2025, doi: 10.31893/multiscience.2025137.
- [7] I. Saker, “Predictive Electrical Submersible Pumps (ESPs) Performance for Better Production Sustainability Using Machine Learning,” Université Kasdi Merbah Ouargla, Ouargla, Algeria, 2024.

- [8] Y. Khalili, M. Ahmadi, and M. K. Moraveji, "Time-aware predictive maintenance of electrical submersible pumps using catboost ensemble learning and trend-based labeling," *J Pet Explor Prod Technol*, vol. 15, no. 9, Sep. 2025, doi: 10.1007/s13202-025-02070-z.
- [9] H. Zhu, H. Yu, Q. Sun, Q. Wang, H. Jing, and R. Abdikadyrov, "Deep Learning-Based Performance Prediction of Electric Submersible Pumps Under Viscous and Gas–Liquid Flow Conditions," *Machines*, vol. 13, no. 2, Feb. 2025, doi: 10.3390/machines13020135.
- [10] C. Zhang *et al.*, "Data-Driven Fault Prediction for Electric Submersible Progressing Cavity Pump Wells," *Processes*, vol. 13, no. 9, Sep. 2025, doi: 10.3390/pr13092890.
- [11] F. Gong, S. Tong, C. Du, Z. Wan, and S. Qiu, "A Fault Diagnosis Model of an Electric Submersible Pump Based on Mechanism Knowledge," *Sensors*, vol. 25, no. 8, Apr. 2025, doi: 10.3390/s25082444.
- [12] M. A. M. Sobhy, "Detecting Electrical Submersible Pump (ESP) Failures and Estimating Run Life Using Artificial Neural Networks," The American University in Cairo, Cairo, Egypt, 2025. [Online]. Available: <https://fount.aucegypt.edu/etds/2605>
- [13] N. Bist, S. Panchal, R. Gupta, A. Soni, and A. Sircar, "Digital transformation and trends for tapping connectivity in the oil and gas sector," *Hybrid Advances*, vol. 6, p. 100256, Aug. 2024, doi: 10.1016/J.HYBADV.2024.100256.
- [14] D. N. Trindade, L. G. Duarte, I. Perico, and G. L. Bandeira, "Driving Change in the Oil and Gas Industry: A Digital Transformation Framework," *Offshore Technology Conference Brasil, OTCB 2023*, 2023, doi: 10.4043/32860-MS.
- [15] Tari Yvonne Elete, Emmanuella Onyinye Nwulu, Ovie Vincent Erhueh, Oluwaseyi Ayotunde Akano, and Adeoye Taofik Aderamo, "Digital transformation in the oil and gas industry: A comprehensive review of operational efficiencies and case studies," *International Journal of Applied Research in Social Sciences*, vol. 6, no. 11, pp. 2611–2643, Nov. 2024, doi: 10.51594/IJARSS.V6I11.1692.

- [16] D. Witzczak and S. Szymoniak, “Review of Monitoring and Control Systems Based on Internet of Things,” *Applied Sciences* 2024, Vol. 14, Page 8943, vol. 14, no. 19, p. 8943, Oct. 2024, doi: 10.3390/APP14198943.
- [17] X. Sui *et al.*, “Intelligent Oil Production Management System Based on Artificial Intelligence Technology,” *Processes* 2025, Vol. 13, Page 133, vol. 13, no. 1, p. 133, Jan. 2025, doi: 10.3390/PR13010133.
- [18] T. Domínguez-Bolaño, O. Campos, V. Barral, C. J. Escudero, and J. A. García-Naya, “An overview of IoT architectures, technologies, and existing open-source projects,” Nov. 01, 2022, *Elsevier B.V.* doi: 10.1016/j.iot.2022.100626.
- [19] M. Burhan, R. A. Rehman, B. Khan, and B. S. Kim, “IoT elements, layered architectures and security issues: A comprehensive survey,” *Sensors (Switzerland)*, vol. 18, no. 9, Sep. 2018, doi: 10.3390/s18092796.
- [20] R. G. Alakbarov and M. A. Hashimov, “Application and Security Issues of Internet of Things in Oil-Gas Industry,” *International Journal of Education and Management Engineering*, vol. 8, no. 6, pp. 24–36, Nov. 2018, doi: 10.5815/ijeme.2018.06.03.
- [21] Ministerio de Hidrocarburos, “Reglamento de Operaciones Hidrocarburíferas,” Quito, Ecuador, 2018.
- [22] Asamblea Nacional del Ecuador, “Ley Orgánica de Telecomunicaciones,” Quito, Ecuador, 2015.
- [23] Ministerio de Telecomunicaciones y de la Sociedad de la Información, “Política Nacional de Ciberseguridad,” Quito, Ecuador, 2021.
- [24] Agencia de Regulación y Control de las Telecomunicaciones (ARCOTEL), “Informe técnico No. IT-CRDS-GR-2022-046: Reformas normativas para la implementación y aplicación del nuevo servicio de conectividad de Internet de las Telecomunicaciones IoT,” Aug. 2022.

- [25] A. S. Nyamawe, M. M. Mjahidi, N. E. Nnko, S. A. Diwani, G. G. Minja, and K. Malyango, *Practical Machine Learning: A Beginner's Guide with Ethical Insights*. Boca Raton, FL: CRC Press, 2025. doi: 10.1201/9781003486817.
- [26] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning: With Applications in R*, 2nd ed. New York, NY: Springer, 2023. doi: 10.1007/978-1-0716-1693-2.
- [27] Rivaldo, R. Taufik, I. S. Ilman, and O. D. E. Wulansari, "A Comparative Study of XGBoost, LightGBM, and CatBoost Models for Customer Churn Prediction in the Banking Industry," *Jurnal Pepadun*, vol. 6, no. 2, pp. 178–187, 2025, doi: 10.23960/pepadun.v6i2.277.
- [28] I. D. Mienye, T. G. Swart, I. D. Mienye, and T. G. Swart, "A Comprehensive Review of Deep Learning: Architectures, Recent Advances, and Applications," *Information 2024, Vol. 15*, vol. 15, no. 12, Nov. 2024, doi: 10.3390/INFO15120755.
- [29] Jaydip Sen, *Machine Learning – Algorithms, Models and Applications*, vol. 7. in *Artificial Intelligence*, vol. 7. London, UK: IntechOpen, 2021. doi: 10.5772/intechopen.98758.
- [30] R. Z. Abdulkreem and A. M. Abdulazeez, "A Comparative Study of Multi-class Classification Based on Imbalanced Data: A Review," *The Indonesian Journal of Computer Science*, vol. 14, no. 5, pp. 8211–8233, 2025, doi: 10.33022/ijcs.v14i5.5020.
- [31] L. A. Gaafar, A. Z. Ghalwash, A. A. Youssif, and H. A. Ghalwesh, "Machine and Deep Learning Models for Multi-Class Sentiment Classification," *J Theor Appl Inf Technol*, vol. 102, no. 22, pp. 8325–8339, 2024.
- [32] O. M. Alyasiri and Y.-N. Cheah, "Multi-Class Text Classification using Machine Learning Techniques," *Engineering, Technology & Applied Science Research*, vol. 15, no. 3, pp. 22598–22604, 2025, doi: 10.48084/etasr.9994.
- [33] G. Labs, "Grafana Documentation," 2024. [Online]. Available: <https://grafana.com/docs/>

- [34] Elastic, “Kibana Documentation,” 2024. [Online]. Available: <https://www.elastic.co/guide/en/kibana/current/index.html>
- [35] M. Corporation, “Power BI Documentation – Overview,” 2024. [Online]. Available: <https://learn.microsoft.com/en-us/power-bi/fundamentals/power-bi-overview>
- [36] S. Fakher, A. Khlaifat, M. E. Hossain, and H. Nameer, “Rigorous Review of Electrical Submersible Pump Failure Mechanisms and Their Mitigation Measures,” *J Pet Explor Prod Technol*, vol. 11, no. 8, pp. 3799–3814, 2021, doi: 10.1007/s13202-021-01271-6.
- [37] M. A. Sobhy, G. M. Hegazy, and A. H. El-Banbi, “Dual-Level Electric Submersible Pump (ESP) Failure Classification: A Novel Comprehensive Classification Bridging Failure Modes and Root Cause Analysis,” *Energies (Basel)*, vol. 18, no. 3943, pp. 1–25, 2025, doi: 10.3390/en18153943.
- [38] S. Al-Ballam, H. Karami, and D. Devegowda, “A Data-Based Reliability Analysis of ESP Failures in Oil Production Wells,” *Journal of Energy and Power Technology*, vol. 4, no. 4, pp. 1–19, 2022, doi: 10.21926/jept.2204036.
- [39] L. Peng, G. Han, X. Sui, A. L. Pagou, L. Zhu, and J. Shu, “Predictive Approach to Perform Fault Detection in Electrical Submersible Pump Systems,” *ACS Omega*, vol. 6, no. 12, pp. 8104–8111, 2021, doi: 10.1021/acsomega.0c05808.
- [40] S. T. Pham, P. S. Vo, and D. N. Nguyen, “Effective Electrical Submersible Pump Management Using Machine Learning,” *Open Journal of Civil Engineering*, vol. 11, pp. 70–80, 2021, doi: 10.4236/ojce.2021.111005.
- [41] M. Bautina, “Reliability Assessment and Failure Prediction of Submersible Pumps Using Advanced Modeling Techniques,” *Computer-Integrated Technologies: Education, Science, Production*, vol. 58, pp. 29–37, 2025, doi: 10.36910/6775-2524-0560-2025-58-04.
- [42] R. Abdalla, H. Samara, N. Perozo, C. P. Carvajal, and P. Jaeger, “Machine Learning Approach for Predictive Maintenance of the Electrical Submersible Pumps

(ESPs),” *ACS Omega*, vol. 7, no. 20, pp. 17641–17651, 2022, doi: 10.1021/acsomega.1c05881.

[43] Y. Song, Y. Na, K. Kim, T. C. Nguyen, J. Wang, and Y. Kim, “Diagnosis of electrical submersible pump failure using deep learning model with sand-water flow experimental data,” *Geoenergy Science and Engineering*, vol. 243, p. 213279, 2024, doi: 10.1016/j.geoen.2024.213279.

[44] H. P. Inc., “HP Laptop 15-da0015la Notebook PC — Especificaciones del producto,” 2018.

[45] Google, “Google Colaboratory FAQ,” Google Research, 2024. [En línea]. Disponible en: <https://research.google.com/colaboratory/faq.html>. [Accedido: 1-dic-2025].

[46] Kaggle, “Kaggle Notebooks Documentation,” 2024. [En línea]. Disponible en: <https://www.kaggle.com/docs/notebooks>. [Accedido: 5-dic-2025].

[47] Amazon Web Services, Inc., “What Is Amazon SageMaker AI?,” Amazon SageMaker Documentation, 2024. [En línea]. Disponible en: <https://docs.aws.amazon.com/sagemaker/latest/dg/whatis.html>. [Accedido: 20-nov-2025].

[48] Projects, “Flask Documentation,” Flask 3.1.x Documentation, 2025. [En línea]. Disponible en: <https://flask.palletsprojects.com/en/stable/>. [Accedido: 4-dic-2025].

[49] CherryPy Team, “CherryPy Web Framework,” CherryPy Documentation, 2025. [En línea]. Disponible en: <https://docs.cherrypy.dev/>. [Accedido: 25-nov-2025].

[50] S. Ramírez, “FastAPI Documentation,” FastAPI Documentation, 2025. [En línea]. Disponible en: <https://fastapi.tiangolo.com/>. [Accedido: 3-dic-2025].

[51] InfluxData, Inc., “InfluxDB Documentation,” InfluxDB Documentation, 2024. [En línea]. Disponible en: <https://docs.influxdata.com/influxdb>. [Accedido: 10-dic-2025].

- [52] InfluxData, Inc., “InfluxDB: Time Series Platform,” InfluxData, 2024. [En línea]. Disponible en: <https://www.influxdata.com/products/influxdb>. [Accedido: 5-dic-2025].
- [53] Timescale, Inc., “TimescaleDB: The Open-Source Time-Series Database,” Timescale, 2024. [En línea]. Disponible en: <https://www.timescale.com>. [Accedido: 7-dic-2025].
- [54] Timescale, Inc., “TimescaleDB Documentation,” TimescaleDB Documentation, 2024. [En línea]. Disponible en: <https://docs.timescale.com>. [Accedido: 11-dic-2025].
- [55] Prometheus Authors and The Linux Foundation, “Prometheus TSDB Storage Documentation,” Prometheus Documentation, 2024. [En línea]. Disponible en: <https://prometheus.io/docs/prometheus/latest/storage>. [Accedido: 1-dic-2025].
- [56] Prometheus Authors and The Linux Foundation, “Prometheus: Monitoring System & Time Series Database,” Prometheus, 2024. [En línea]. Disponible en: <https://prometheus.io>. [Accedido: 6-dic-2025].
- [57] Docker Inc., “Docker Documentation,” Docker Docs, California, USA, 2024. [En línea]. Disponible en: <https://docs.docker.com/manuals/>. [Accedido: 12-nov-2025].
- [58] N. Poulton, *Docker Deep Dive*, 4th ed. Leanpub, 2023.
- [59] A. M. Shimaoka, R. C. Ferreira, and A. Goldman, “The evolution of CRISP-DM for Data Science: Methods, Processes and Frameworks,” *SBC Reviews on Computer Science*, Oct. 2024, doi: 10.5753/reviews.2024.3757.
- [60] J. Brzozowska, J. Pizoń, G. Baytikenova, A. Gola, A. Zakimova, and K. Piotrowska, “Data Engineering in CRISP-DM Process Production Data – Case Study,” *Applied Computer Science*, vol. 19, no. 3, pp. 83–95, 2023, doi: 10.35784/acs-2023-26.

ANEXOS

Anexo A. Categorización de variables.

En la Figura 51 y Figura 52 se muestra la categorización de variables para relacionadas al desarrollo del trabajo de titulación.



Figura 51. Variable Dependiente.

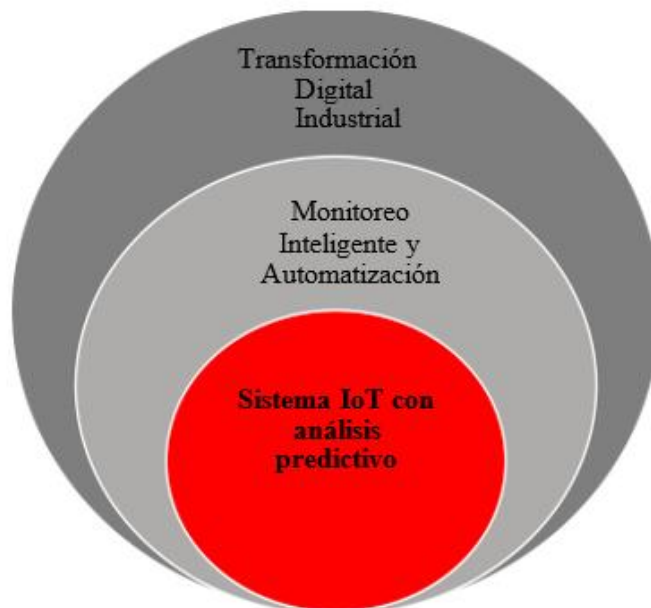


Figura 52. Variable Independiente.

En la Figura 53 y Figura 54 la constelación de ideas para el desarrollo del trabajo de titulación.

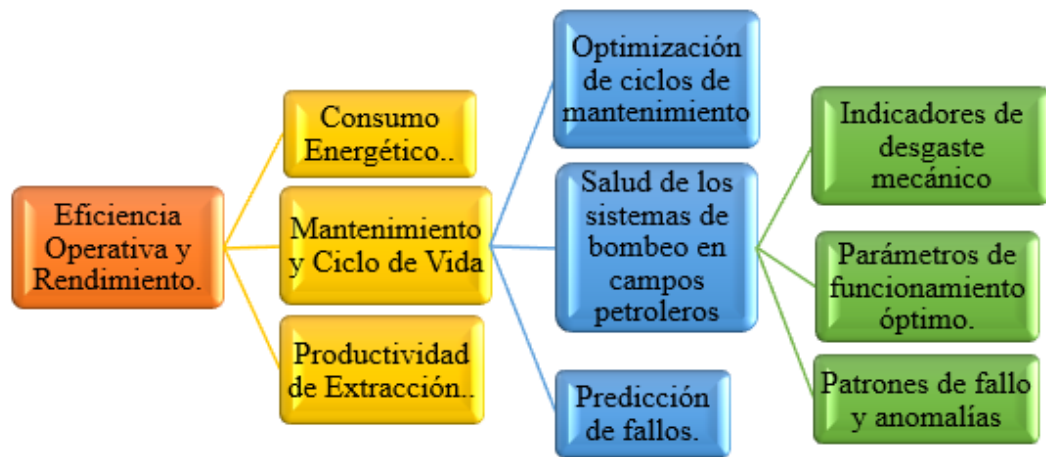


Figura 53. Constelación de ideas de la variable dependiente.

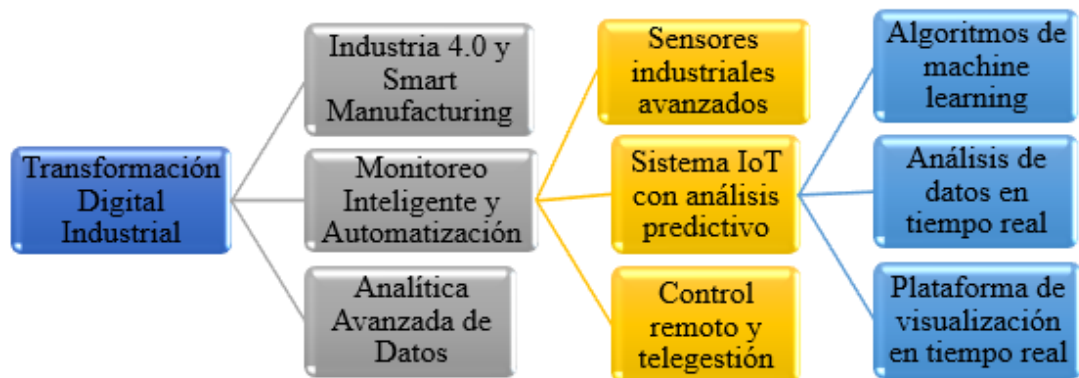


Figura 54. Constelación de ideas de la variable independiente.

Anexo B. Pruebas Adicionales.

En la Figura 55 se muestra el panel de Grafana empleado para supervisar la probabilidad de falla de las bombas ESP y las señales de voltaje y presión.

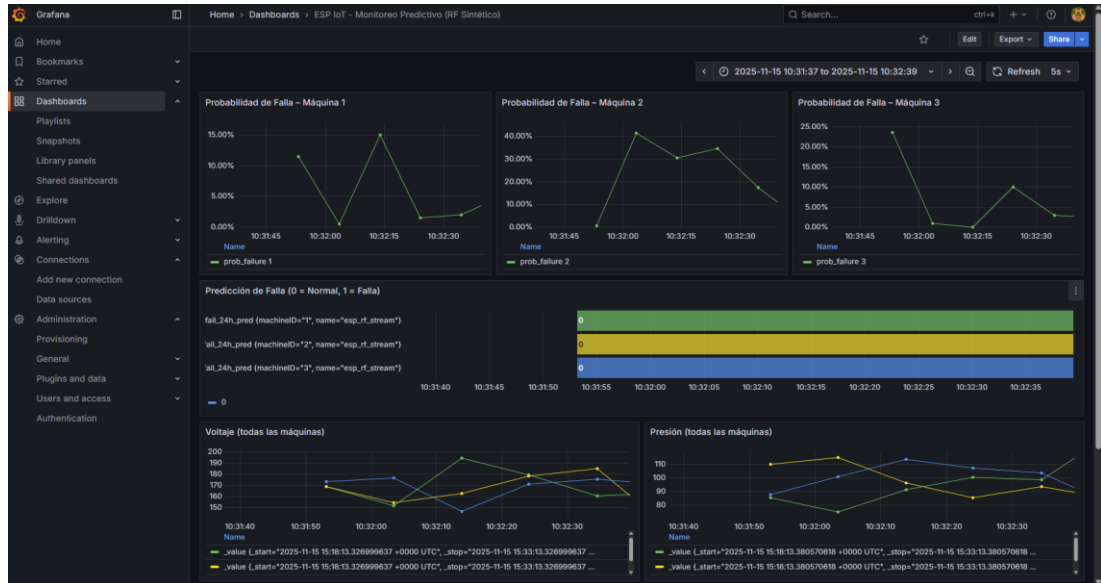


Figura 55. Dashboard de grafana.

En la Figura 56 se presenta el cuerpo de la petición POST y la respuesta JSON del servicio REST.



Figura 56. Respuesta de API.

Anexo C. Código de entrenamiento del modelo Random Forest.

```
# =====
# 1. LIBRERÍAS NECESARIAS
# =====
import numpy as np
import pandas as pd
import time
import os
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.utils import shuffle
import matplotlib
matplotlib.use("Agg")
import matplotlib.pyplot as plt
import seaborn as sns
import pickle
from datetime import datetime, timedelta
from sklearn.metrics import precision_recall_curve
# =====
# 2. CARGA DE ARCHIVOS LOCALES
# =====
print("📁 Cargando datasets desde la carpeta local...")
BASE_PATH = r"C:\Users\HP\Downloads\EntModelosML\"
file_X = BASE_PATH + "DataSet_BombaESP_X.csv"
file_Y = BASE_PATH + "DataSet_BombaESP_Y.csv"
X = pd.read_csv(file_X)
Y = pd.read_csv(file_Y)
print("Shape X:", X.shape)
print("Shape Y:", X.shape)
print("\nColumnas de X:", list(X.columns))
print("Columnas de Y:", list(Y.columns))
# =====
# 2.1 PROCESAR TIMESTAMP (FEATURE ENGINEERING)
# =====
print("\n⌚ Procesando columna 'timestamp'...")
X["timestamp"] = pd.to_datetime(X["timestamp"])
X["anio"] = X["timestamp"].dt.year
X["mes"] = X["timestamp"].dt.month
X["dia"] = X["timestamp"].dt.day
X["hora"] = X["timestamp"].dt.hour
X["dia_semana"] = X["timestamp"].dt.dayofweek # 0=lunes, 6=domingo
X["dia_anio"] = X["timestamp"].dt.dayofyear
X = X.drop(columns=["timestamp"])
print("Columnas de X después de procesar timestamp:")
```

```

print(list(X.columns))
# =====
# 3. DIVISIÓN TRAIN / TEST
# =====
print("\n🔪 Dividiendo dataset en Train/Test...")
X_train, X_test, Y_train, Y_test = train_test_split(
    X,
    Y,
    test_size=0.20,
    random_state=42,
    shuffle=True
)
print("\n===== SHAPES DESPUÉS DEL SPLIT =====")
print("X_train:", X_train.shape)
print("X_test :", X_test.shape)
print("Y_train:", Y_train.shape)
print("Y_test :", Y_test.shape)
print("\nDistribución en Train - estado_bomba:")
print(Y_train["estado_bomba"].value_counts())
# =====
# 4. DOWNSAMPLING INTELIGENTE
# =====
print("\n🔪 Realizando downsampling para acelerar entrenamiento...")
mask_normal = (Y_train["estado_bomba"] == 0)
mask_falla = (Y_train["estado_bomba"] != 0)
X_norm = X_train[mask_normal]
Y_norm = Y_train[mask_normal]
X_falla = X_train[mask_falla]
Y_falla = Y_train[mask_falla]
n_norm = min(300_000, len(X_norm))
rng = np.random.default_rng(42)
idx_norm = rng.choice(len(X_norm), size=n_norm, replace=False)
X_norm_sample = X_norm.iloc[idx_norm]
Y_norm_sample = Y_norm.iloc[idx_norm]
X_train_small = pd.concat([X_norm_sample, X_falla], axis=0)
Y_train_small = pd.concat([Y_norm_sample, Y_falla], axis=0)
X_train_small, Y_train_small = shuffle(X_train_small, Y_train_small,
    random_state=42)
print("Nuevo X_train_small:", X_train_small.shape)
print("Nuevo Y_train_small:", Y_train_small.shape)
# =====
# 5. ENTRENAMIENTO DEL MODELO
# =====
print("\n🧠 Entrenando Random Forest multisalida...")
rf = RandomForestClassifier(
    n_estimators=100,
    max_depth=15,
    min_samples_leaf=5,

```

```

        n_jobs=-1,
        random_state=42
    )
    modelo = MultiOutputClassifier(rf, n_jobs=-1)
    t0_ent = time.perf_counter()
    modelo.fit(X_train_small, Y_train_small)
    t1_ent = time.perf_counter()
    tiempo_entrenamiento = t1_ent - t0_ent
    min_ent, seg_ent = divmod(tiempo_entrenamiento, 60)
    print("☑ Modelo entrenado correctamente.")
    print(f"🕒 Tiempo de entrenamiento: {min_ent:.0f} min {seg_ent:.2f} s")
    # =====
    # 6. FUNCIÓN PARA MÉTRICAS E INFERENCIA
    # =====
    def evaluar(nombre_col, idx):
        print("\n" + "="*60)
        print(f" MÉTRICAS PARA: {nombre_col}")
        print("="*60)
        y_true = Y_test[nombre_col].values
        t0_inf = time.perf_counter()
        y_pred = modelo.estimators_[idx].predict(X_test)
        t1_inf = time.perf_counter()
        tiempo_inf = t1_inf - t0_inf
        print(f"🕒 Tiempo de inferencia para '{nombre_col}': {tiempo_inf:.4f} s")
        print("\n📊 Classification report:")
        print(classification_report(y_true, y_pred))
        cm = confusion_matrix(y_true, y_pred)
        print("Matriz de confusión:")
        print(cm)
        plt.figure(figsize=(6,4))
        sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)
        plt.title(f"Matriz de Confusión - {nombre_col}")
        plt.xlabel("Predicción")
        plt.ylabel("Real")
        plt.tight_layout()
        ruta_cm = BASE_PATH + f"RF_{nombre_col}.png"
        plt.savefig(ruta_cm)
        plt.close()
        print(f"📁 Matriz de confusión guardada en: {ruta_cm}")
    # =====
    # 7. EVALUAR LAS 5 SALIDAS
    # =====
    cols = list(Y.columns)
    evaluar("variable_falla", cols.index("variable_falla"))
    evaluar("subcomponente_falla", cols.index("subcomponente_falla"))
    evaluar("severidad_falla", cols.index("severidad_falla"))

```

```

evaluar("estado_bomba", cols.index("estado_bomba"))
evaluar("falla_predicha_24h", cols.index("falla_predicha_24h"))
# =====
# 8. IMPORTANCIA DE VARIABLES PARA 'estado_bomba'
# =====
print("\n🔗 Calculando importancia de variables para
'estado_bomba'...")
idx_estado = cols.index("estado_bomba")
modelo_estado = modelo.estimators_[idx_estado]
importances = modelo_estado.feature_importances_
features = np.array(X.columns)
idx_sorted = np.argsort(importances)[:,-1][:15]
plt.figure(figsize=(8,6))
sns.barplot(x=importances[idx_sorted], y=features[idx_sorted])
plt.title("Importancia de variables - estado_bomba")
plt.tight_layout()
ruta_import = BASE_PATH + "Importancias_estado_bomba.png"
plt.savefig(ruta_import)
plt.close()
print(f"🖼️ Gráfico de importancia de variables guardado en:
{ruta_import}")
# =====
# 9. GUARDAR EL MODELO ENTRENADO
# =====
ruta_modelo = BASE_PATH + "modelo_bombasESP_RF.pkl"
with open(ruta_modelo, "wb") as f:
    pickle.dump(modelo, f)
size_bytes = os.path.getsize(ruta_modelo)
size_mb = size_bytes / (1024 * 1024)
print(f"\n💾 Modelo guardado correctamente en:\n{ruta_modelo}")
print(f"📦 Tamaño del archivo del modelo: {size_mb:.2f} MB
({size_bytes:,} bytes)")

```

Anexo D. Lineas de despliegue para Docker.

En la Figura 57 se muestra el archivo docker-compose.yml utilizado para orquestar los servicios de la solución propuesta.

```
version: "3.9"

services:
  influxdb:
    image: influxdb:2.7
    container_name: influxdb
    ports:
      - "8086:8086"
    env_file:
      - .env
    volumes:
      - ./influxdb/data:/var/lib/influxdb2
      - ./influxdb/config:/etc/influxdb2

  python-app:
    build: ./python-app
    container_name: python-app
    env_file:
      - .env
    depends_on:
      - influxdb
    ports:
      - "8000:8000"

  cliente-esp:
    build: ./python-app
    container_name: cliente-esp
    env_file:
      - .env
    depends_on:
      - python-app
      - influxdb
    command: ["python", "cliente_esp_influx.py"]

  grafana:
    image: grafana/grafana:latest
    container_name: grafana
    env_file:
      - .env
    depends_on:
      - influxdb
    ports:
      - "3000:3000"
    volumes:
      - ./grafana/data:/var/lib/grafana
      - ./grafana/provisioning:/etc/grafana/provisioning
```

Figura 57. Archivo docker-compose.yml