



**Escola Politècnica Superior
d'Enginyeria de Vilanova i la Geltrú**

UNIVERSITAT POLITÈCNICA DE CATALUNYA

TRABAJO FINAL DE GRADO

TÍTULO:

USO DE MINERÍA DE DATOS EN INDUSTRIA 4.0

AUTOR:

HMIMOU, ACHRAF

FECHA DE PRESENTACIÓN: Julio, 2022

APELLIDOS: HMIMOU

NOMBRE: ACHRAF

TITULACIÓN: GRADO EN INGENIERÍA INFORMÀTICA

PLAN: 2018

DIRECTORA: NEUS CATALÀ I ROIG

DEPARTAMENTO: CIENCIAS DE LA COMPUTACIÓN

CALIFICACIÓN DEL TFG

TRIBUNAL

PRESIDENTE

SECRETARIO

VOCAL

FECHA DE LECTURA:

13 de Julio, 2022

Este proyecto tiene en cuenta factores medioambientales: Sí X No

RESUMEN

La gran cantidad de datos generada por la cuarta revolución industrial (Industria 4.0) puede suponer un avance significativo en la mejora de los procesos. Extraer información útil de estos grandes volúmenes de datos es el principal reto y es, también, donde la Minería de Datos juega un papel clave.

Este Trabajo Final de Grado parte de la idea de seguir un proceso de minería de datos en un escenario real de producción industrial. El escenario concreto es una célula de fabricación de piezas para el sector automovilístico mediante fundición a presión.

La aplicación central es la implementación de mantenimiento predictivo, una técnica que permite la detección de comportamientos anómalos para poder anticiparse a posibles fallos. La predicción se realiza mediante el análisis y aprendizaje del comportamiento de los sensores instalados en las máquinas o equipos.

Siguiendo el método estándar de minería de datos, los dos objetivos principales son: 1) Obtener modelos de aprendizaje automático para implementar el mantenimiento predictivo, y 2) Crear una aplicación que automatice el proceso que incluye la recogida de datos, el tratamiento de estos y, finalmente, la visualización de los resultados en tiempo real.

Para llevar a cabo estos objetivos, se exploran primero diferentes propiedades y técnicas de tratamiento de datos obtenidos de sensores. Con los datos tratados, se crean y evalúan modelos que aprenden de los datos y puedan ser usados para predecir futuros comportamientos. Finalmente, se desarrolla una aplicación combinando diferentes tecnologías para poder efectuar análisis de resultados y monitorización de la evolución del estado de los sensores.

Palabras clave (màximo 10):

Minería de datos	Industria 4.0	IoT	Mantenimiento predictivo
Fundición a presión	CRISP-DM	Redes Neuronales	Node-RED
Python	Vue JS		

ABSTRACT

The large amount of data generated by the fourth industrial revolution (Industry 4.0) can represent a significant advance in the improvement of processes. Extracting useful information from these large volumes of data is the main challenge and it is also where Data Mining plays a key role.

This Final Degree Project is based on the idea of following a data mining process in a real scenario of industrial production. The specific scenario is a cell for the manufacture of parts for the automotive sector by means of die casting.

The central application is the implementation of predictive maintenance, a technique that allows the detection of abnormal behavior in order to anticipate possible failures. The prediction is made through the analysis and learning of the behavior of the sensors installed in the machines or equipment.

Following the standard method of data mining, the two main objectives are: 1) Obtain machine learning models to implement predictive maintenance, and 2) Create an application that automates the process that includes data collection, treatment and, finally, the visualization of the results in real time.

To carry out these objectives, different properties and data processing techniques obtained from sensors are first explored. With the processed data, models are created and evaluated that learn from the data and can be used to predict future behavior. Finally, an application is created combining different technologies to be able to analyze the results and monitor the evolution of the state of the sensors.

Keywords (10 maximum):

Data mining	Industry 4.0	IoT	Predictive maintenance
Die Casting	CRISP-DM	Neural Networks	Node-RED
Python	Vue JS		

ÍNDICE

1. INTRODUCCIÓN	12
1.1. CIE C. Vilanova	12
1.2. Estado del arte: Mantenimiento predictivo	12
1.3. Motivación y alcance	14
2. ESCENARIO Y OBJETIVOS	15
2.1. Escenario: Célula 86	15
2.2. Objetivos	22
3. MINERÍA DE DATOS	24
3.1. Metodología: CRISP-DM	24
3.2. Análisis de negocio	25
3.3. Análisis de datos	26
3.3.1. Análisis de series temporales	27
3.3.2. Análisis multivariado	31
3.4. Preprocesado	38
3.5. Modelado	39
3.5.1. Predicción temporal (Regresión)	40
3.5.2. Detección de anomalías (Clasificación)	44
3.6. Evaluación	45
4. DESARROLLO Y DESPLIEGUE APLICACIÓN	49
4.1. Herramientas	49
4.2. Arquitectura	51
4.3. Desarrollo	53
4.3.1. Pipeline de datos	54
4.3.2. Entrenamiento de modelos	55
4.3.2. Aplicación web	55
4.4. Demo	57
5. TRABAJO PARA FUTURO	64
6. CONCLUSIONES	66
7. GESTIÓN DEL PROYECTO	67
7.1. Planificación temporal	67
7.2. Gestión económica	70
8. AGRADECIMIENTOS	71
9. REFERENCIAS	72
9.1. Documentos	72
9.2. Referencias Web	72
10. ANEXOS	76
ANEXO A: FUNDICIÓN A PRESIÓN	76
Máquina	76
Molde	80

El material	87
Parámetros de proceso	88
ANEXO B: SEÑALES CRÍTICAS	90
ANEXO C: VECTORES AUTORREGRESIVOS	92
ANEXO D: MÓDULOS PYTHON	94
ANEXO E: ARQUITECTURA (docker-compose.yml)	96
ANEXO F: APLICACIÓN WEB (package.json)	98
ANEXO G: DIAGRAMA GANTT AMPLIADO	100

SUMARIO DE FIGURAS Y TABLAS

Figura 1.1. Imagen de la planta CIE C.Vilanova.	12
Figura 1.2. Comparación de mantenimiento correctivo vs preventivo vs predictivo.	13
Tabla 2.1. Elementos de los grupos de Inyección y Periféricos.	16
Figura 2.2. Mapa de la Célula 86.	16
Figura 2.3. Esquema de comunicación Célula 86.	17
Figura 2.4. Cuadro de control del PLC Máster.	17
Figura 2.5. Màquina de Inyección IDRA OL 1600S.	18
Figura 2.6. Horno de mantenimiento sin dosificador de la CEL86	18
Figura 2.7. Cargador lineal de metal de la CEL86.	18
Figura 2.8. Sistema lubrificador de dos ejes de la CEL86.	19
Figura 2.9. Atemperadores de CEL86.	19
Figura 2.10. Sistema completo de robot: controlador y manipulador.	19
Figura 2.11. Prensa de troquel de la CEL86.	20
Figura 2.12. Cuba de refrigeración con elevador.	20
Figura 2.13. Máquina de secado por vacío	20
Figura 2.14. Esquema granalladora convencional	21
Figura 2.15. Máquina de secado por vacío	21
Figura 2.16. Ciclo de fabricación de una pieza por inyección	21
Figura 2.17. Treemap de todas las señales disponibles en la CEL86 agrupadas.	22
Figura 3.1. Esquema del ciclo de vida de un proyecto mediante la metodología CRISP-DM	24
Figura 3.2. Muestra de 24 horas de señales críticas.	26
Figura 3.3. Ejemplo de la muestra.	27
Figura 3.4. Gráficas temporales de la muestra de señales críticas.	28
Figura 3.5. Gráfica temporal de la variable Temp_Horno.	28
Figura 3.6. Componentes de una serie temporal.	29
Figura 3.7. Descomposición de la serie temporal de Temp_Horno.	29
Figura 3.8. Resultados del Augmented Dickey-Fuller Test sobre Temp_Horno.	30
Figura 3.9. Resultados de aplicar la técnica de <i>detrending</i> sobre temp Horno.	30

Figura 3.10. Gráfica de autocorrelación y autocorrelación parcial sobre Temp_Horno.	31
Figura 3.11. Resumen estadístico de la muestra de datos.	31
Figura 3.12. Matriz de correlación de la muestra.	32
Figura 3.13. <i>Pairplot</i> entre la variable Caudal_Piston y Pres_Entrada_Agua_Maquina	33
Figura 3.14. Transformación de variables a componentes	33
Figura 3.15. Ecuación de un componente principal	33
Figura 3.16. Ejemplo de representación geométrica de dos componentes principales	34
Figura 3.17. Resultados obtenidos al aplicar PCA sobre la muestra.	34
Figura 3.18. Mapa de calor que indica la variabilidad de las variables en los componentes	35
Figura 3.19. Aplicación de K-Means y análisis de silueta sobre la muestra.	37
Figura 3.20. Aplicación de PCA + K-Means y análisis de silueta sobre la muestra.	38
Figura 3.21. Representación de Interpolación lineal de dos puntos	39
Figura 3.22. Arquitectura genérica de una red neuronal feed-forward	42
Figura 3.23. Composición de una neurona LSTM	42
Figura 3.24. Arquitectura del modelo final (generada mediante la herramienta netron.app)	43
Figura 3.25. Ejemplo de detección multivariada de anomalías	44
Figura 3.26. Ejemplo visual de cómo se detectan las anomalías	45
Figura 3.27. Diagrama generado de la pipenet creada que combina PCA + KMeans.	45
Figura 3.28. Métricas del modelo en el conjunto de entrenamiento y validación, con una muestra de 48h.	46
Figura 3.29. Fórmula del error cuadrático medio	46
Figura 3.30. Gráficos generados por el método walk validation sobre el conjunto de test.	47
Figura 3.31. Resultado de detección de anomalía multivariada sobre una muestra de 24h.	47
Figura 4.1. Logo y vista de Node-RED	49
Figura 4.2. Logo Python	50
Figura 4.3. Logo InfluxDB	50

Figura 4.4. Logo Docker	51
Figura 4.5. Hardware proporcionado para el proyecto	51
Figura 4.6. Diagrama de la arquitectura de servicios y componentes de la aplicación final.	52
Figura 4.7. Flujo principal de Node-RED donde se procesan los datos.	54
Figura 4.8. Flujo de entrenamiento de modelos y visualización de resultados.	55
Figura 4.9. Flujo de la instancia web (UIBuilder) y posibilidad de consultas a InfluxDB.	56
Figura 4.10. Boceto inicial del dashboard web	57
Figura 4.11. Partes principales del dashboard web.	57
Figura 4.12. Opciones barra lateral.	58
Figura 4.13. widget PlotlyGrafica y sus características.	59
Figura 4.14. widget Medidor.	59
Figura 4.15. widget ApexGrafica y sus características.	60
Figura 4.16. widget EchartsGrafica.	60
Figura 4.17. Opciones generales de cada widget.	61
Figura 4.18. Opciones internas del widget PlotlyGrafic	61
Figura 4.19. Dashboard con widgets y la barra lateral escondida.	62
Figura 4.20. Modal con manual de la aplicación e información de los modelos.	62
Figura 5.1. Roadmap de las próximas versiones de la aplicación.	64
Figura 7.1. Planificación inicial de etapas realizadas	67
Tabla 7.2. Tabla de tareas realizadas con la etapa y el rol que pertenecen.	68
Figura 7.3. Diagrama Gantt de las tareas realizadas.	69
Figura 7.4. Gráfico circular que indica la distribución de horas por etapas.	69
Tabla 7.5. Tabla de estimación de coste de recursos humanos.	70
Tabla 7.6. Tabla de estimación de costes de recursos.	70
Tabla 7.7. Tabla de estimación de costes totales del proyecto.	70
Figura 9.1. Esquema unidad de inyección en la máquina inyectora.	76
Figura 9.2. Esquema de la primera fase de la inyección (fase de aproximación).	77
Figura 9.3. Esquema de la segunda fase de inyección (fase de velocidad).	78
Figura 9.4. Esquema de la tercera fase de inyección (Fase de multiplicación)	78

Figura 9.5. Ejemplo de curva de inyección.	79
Figura 9.6. Esquema de la unidad de cierre de una máquina de inyección.	80
Figura 9.7. Ejemplo de diseño de la geometría del molde.	80
Tabla 9.8. Tabla de relaciones entre la temperatura de molde, temperatura de material y la velocidad de llenado.	82
Figura 9.9. Esquema de un tanque de vacío para la evacuación del aire del mode.	83
Tabla 9.10. Tabla clasificatoria de los defectos típicos	83
Figura 9.11. Almacén de reserva de aluminio	87
Tabla 9.12. Tabla de los parámetros principales de proceso.	88
Tabla 9.13. Tabla de las señales críticas recibidas desde el PLC Máster.	90
Figura 9.14. Ecuaciones que definen los vectores autorregresivos	92
Figura 9.15. Métricas obtenidas de modelar mediante vectores autorregresivos.	92
Figura 9.16. Diagrama Gantt ampliado	100

GLOSARIO DE ABREVIATURAS, ACRÓNIMOS Y TÉRMINOS

API (del inglés *Application Programming Interface*): Se trata del conjunto de procesos, funciones o métodos abstractos de una aplicación que permite ser empleada por otra.

Dashboard: Herramienta de gestión de la información donde se pueden visualizar métricas y datos de un proceso específico.

Dataset: Colección o conjunto de datos habitualmente tabulados.

Deep Learning: Rama del aprendizaje automático que se basa en modelos que intentan asemejar en mayor medida a la organización del sistema nervioso: las redes neuronales. Se habla de *deep* (profundo) cuando la arquitectura de la red neuronal está jerarquizada y formada por gran cantidad de capas de neuronas (*layers*).

IoT (del inglés *Internet of Things*): Conjunto de dispositivos y objetos que forman parte de una red donde se pueden comunicar e interaccionar entre ellos.

LSTM (del inglés *Long Short-Term Memory*): Dentro de las redes neuronales recurrentes, es un tipo especial de neurona que es capaz de simular la memoria humana.

Overfitting: En el contexto del aprendizaje automático, este fenómeno se produce cuando un modelo funciona muy bien con el conjunto de entrenamiento, pero mal en observaciones nuevas.

Pipeline de datos: Construcción lógica que representa los diferentes procesos que involucran el movimiento o procesamiento de datos.

PLC (del inglés *Programmable Logic Controller*): Computadora utilizada para automatizar procesos industriales.

RNN (del inglés *Recurrent Neural Network*): Tipo de red neuronal que contiene componentes cílicos. Por lo tanto, permite utilizar predicciones antiguas como datos de entrada.

Token: Referencia o identificador que permite sustituir un valor sensible y puede ser utilizado para otorgar algún permiso o acceso.

TSDB (del inglés *Time Series DataBase*): Base de datos optimizada para datos temporales o series temporales.

1. INTRODUCCIÓN

La inmensa cantidad de información disponible hoy en día y los grandes avances en la informática hace que la **minería de datos** sea una de las disciplinas más populares. Encontrar patrones, tendencias, anomalías y extraer información útil de grandes volúmenes de datos puede ser clave para afrontar futuras decisiones.

Por otra parte, la cuarta revolución industrial, también conocida como **Industria 4.0**, supone un paso gigantesco en la mejora de los procesos industriales. Los sensores **IoT** (*Internet of Things*) son la base de esta revolución y permiten una gran recolección de datos que pueden ser explotados.

Este proyecto se titula “*Uso de minería de datos en industria 4.0*” porque el objetivo es utilizar la información generada por los sensores durante un proceso productivo para mejorarlo mediante minería de datos.

1.1. CIE C. Vilanova

Este proyecto está realizado en CIE C. Vilanova [4], una fábrica que forma parte de la compañía **CIE Automotive**, uno de los principales proveedores de piezas para el sector de la automoción. La fábrica tiene una refinería, en la que se crea la aleación que forma las piezas, y cuenta además con las diferentes máquinas que permiten la creación de las piezas mediante inyección. También dispone de áreas donde se mecanizan las piezas. El producto final son piezas mecanizadas o sin mecanizar.



Figura 1.1. Imagen de la planta CIE C. Vilanova

Durante el proceso productivo hay mucha información generada que va a diferentes aplicaciones. Parte de la información es generada internamente y puede ser fuente de diferentes aplicaciones.

Concretamente, este proyecto se centra en los datos generados por las células de inyección. En estas intervienen diferentes componentes para la elaboración de las piezas. El tener bajo control el funcionamiento de los componentes es vital para el mantenimiento de los componentes, así como asegurar la calidad del producto.

Una de las aplicaciones más relevantes del uso de los datos obtenidos es la del **mantenimiento predictivo**. Esta práctica consiste en utilizar métodos de aprendizaje automático para detectar anomalías en el funcionamiento de una célula de producción y poder predecir el comportamiento de sus elementos.

1.2. Estado del arte: Mantenimiento predictivo

Reducir los costes de mantenimiento es uno de los grandes retos en la industria. Las anomalías y fallos afectan mucho al rendimiento productivo y suponen grandes pérdidas para la empresa.

Por mucho tiempo, el **mantenimiento correctivo** ha sido la metodología principal del sector. Esperar hasta que falle la máquina es una mala opción debido a que las pérdidas en tiempo y en producción pueden ser muy graves y provocar grandes pérdidas económicas.

Actualmente, es mucho más común el **mantenimiento preventivo** en el que se toman acciones de forma periódica para revisar el estado y evitar las fallidas. De este modo, se puede prevenir el desgaste y reducir el impacto que pueden tener los fallos. Aun así, requiere mucha planificación y costes.

El **mantenimiento predictivo** es una propuesta que se apoya en los avances de la Inteligencia Artificial para que la tarea de mantenimiento se realice solamente cuando sea necesario. La clave está en estudiar los patrones de comportamiento y detectar posibles anomalías que puedan afectar a futuro en la máquina.

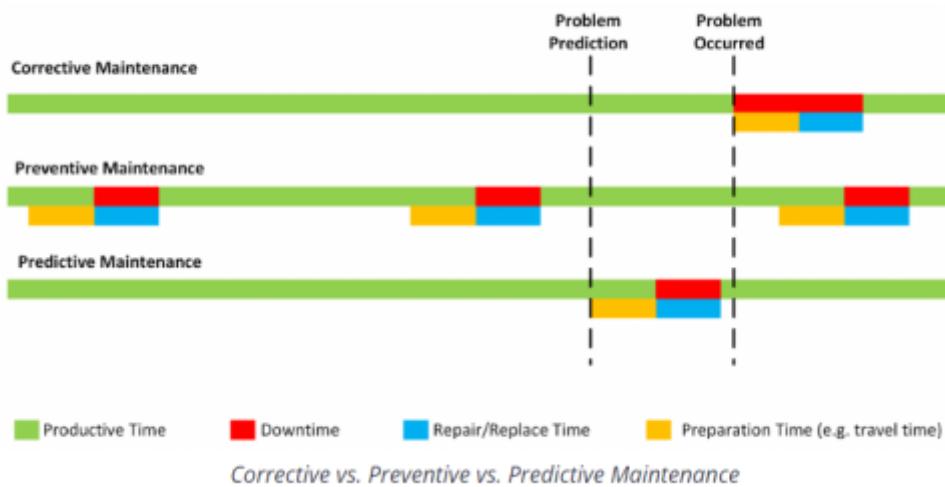


Figura 1.2. Comparación de mantenimiento correctivo vs preventivo vs predictivo
(fuente: [2]).

Como se puede observar en la Figura 1.2, el mantenimiento predictivo permite anticiparse al posible fallo y poder realizar una preparación para la intervención. Esta predicción puede ser dada por la frecuente aparición de una anomalía en los sensores de la máquina. De esta manera, es posible minimizar las frecuentes revisiones y solo efectuar el mantenimiento cuando se detecte un comportamiento no habitual.

Uno de los trabajos más recientes sobre mantenimiento predictivo, donde se revisan diferentes enfoques y metodologías, concluye indicando que las propuestas que incluyen ingeniería de datos y matemática, cada vez están más presentes en un área donde antes solamente había ingenieros expertos en el dominio. Por lo tanto, es muy importante el enfoque multidisciplinario en la Industria 4.0 [1].

Otro trabajo donde se contextualizan los avances más actuales en el tema teniendo en cuenta factores tecnológicos como arquitecturas, escalabilidad y seguridad, apunta a que el mantenimiento predictivo, aun siendo un tema muy de moda en el contexto Industria 4.0, todavía requiere mucha investigación en su aplicación y razonamiento [2].

Finalmente, concluir esta sección indicando que se trata de un concepto muy reciente y todavía hay mucha investigación al respecto. Ya hay algunas compañías y startups

que ofrecen servicios para la implementación de mantenimiento predictivo. Algunas de las soluciones más destacadas incluyen Siemens Mindsphere [5] y LimbleCMMS [6].

1.3. Motivación y alcance

Inicialmente, este proyecto pretendía encontrar una solución aplicable a uno de los componentes de fabricación (célula de inyección) sin optar por un método concreto. A medida que se iban explorando los datos y analizando las diferentes posibilidades, varias ideas surgieron (ver apartado [3.2](#)) de las cuales se consideró que la más viable era la del mantenimiento predictivo.

La implementación de un proyecto de este tipo suponía todo un reto y también estaba la incertidumbre de conseguir un buen resultado a causa de la falta de experiencia en la parte de minería de datos. Aun así, había viabilidad dado que se disponía de un buen escenario y de las herramientas.

En general, he intentado encontrar un balance entre desarrollo, que es la parte más familiar a la titulación, y minería de datos, la parte más desconocida y que requería más aprendizaje por mi parte.

El alcance de este proyecto se constituye en los siguientes puntos:

- Descripción de los elementos que intervienen en una célula de producción.
- Aplicación de la metodología de minería de datos (CRISP-DM) a un proceso productivo.
- Análisis de negocio y exploración de potenciales aplicaciones (Mantenimiento predictivo).
- Analizar datos de naturaleza temporal y técnicas de análisis multivariado.
- Creación y evaluación de modelos de aprendizaje automático.
- Diseño y despliegue de una arquitectura que automatice el proceso de recolección, tratamiento y visualización de datos.
- Desarrollo web basado en frameworks modernos (Vue JS).

Adicionalmente, también se describen conceptos contextuales como la fundición a presión.

2. ESCENARIO Y OBJETIVOS

Para entender el contexto de este proyecto y entender muchos de los elementos que forman parte de él, es de gran ayuda entender el proceso de **fundición a presión**.

La fundición a presión consiste en introducir material fundido dentro de un molde de acero bajo un gran impulso. La alta presión en la que se inyecta el material líquido, hace que el material se introduzca en las secciones estrechas y se comprima contra las paredes del molde.

El calor del material fundido se transfiere al molde y hace que el material se solidifique. Una vez se solidifica la pieza, se abre el molde y se extrae la pieza.

El proceso básico es relativamente sencillo de entender y hace que sea una de las técnicas más competitivas debido a la rapidez de producción y la calidad de las piezas obtenidas.

Aunque la idea general es sencilla, hay muchos factores que influyen en el proceso y son muchos los retos que el departamento de ingeniería debe afrontar. Los principales factores son: la **máquina**, el **molde**, el **material** y los **parámetros de proceso**. Para entender estos componentes con mayor profundidad ver [ANEXO A](#).

CIE C. Vilanova tiene muchas células de producción donde se crea la pieza mediante fundición a presión. La inmensa cantidad de datos generados por las células y la gran variabilidad que hay entre ellas hace que aplicar minería de datos sea un problema muy complejo. Con esta premisa, se ha optado por empezar con un escenario muy simplificado para realizar este proyecto.

El escenario donde se desarrollará el proyecto será en una sola célula de producción y teniendo en cuenta que el análisis se efectuará sobre una sola pieza. Esto simplifica el problema y elimina la variabilidad que supone diferentes células y diferentes máquinas por muy similares que puedan ser los parámetros.

La idea ha sido emplear un escenario de producción real, pero simplificado de manera que se pueda empezar a aplicar minería de datos. También recalcar que aunque el escenario sea acotado, muchos de los elementos que intervienen en la producción de una pieza estarán presentes. El objetivo es crear un proyecto base que permita que en un futuro se pueda extrapolar, añadiendo la variabilidad que supone más máquinas, piezas, sensores u otros elementos.

2.1. Escenario: Célula 86

La Célula 86 ha sido el escenario del proyecto debido a que recientemente se le han incorporado una gran cantidad de sensores adicionales. Además, se trata de una célula nueva y dispone de una muy buena documentación.

La célula consta de diferentes elementos que se dividen en dos grupos: Inyección y Periféricos.

- **Inyección:** Elementos que intervienen en el proceso de inyección.
- **Periféricos:** Elementos periféricos de acabado de la pieza de fundición.

En la Tabla 2.1, se detallan los elementos y se indica su pertenencia a uno de los dos grupos.

Tabla 2.1. Elementos de los grupos de Inyección y Periféricos.

Inyección	Periféricos
Máquina Inyectora	Robot
Horno de mantenimiento	Prensa
Cargador de metal	Cuba de refrigeración con elevación
Lubrificador	Máquina de secado por vacío
	Mesa de reintroducción de piezas
	Granalladora
	Atemperadores

La distribución de los elementos en la célula se puede ver en el mapa de la Figura 2.2.

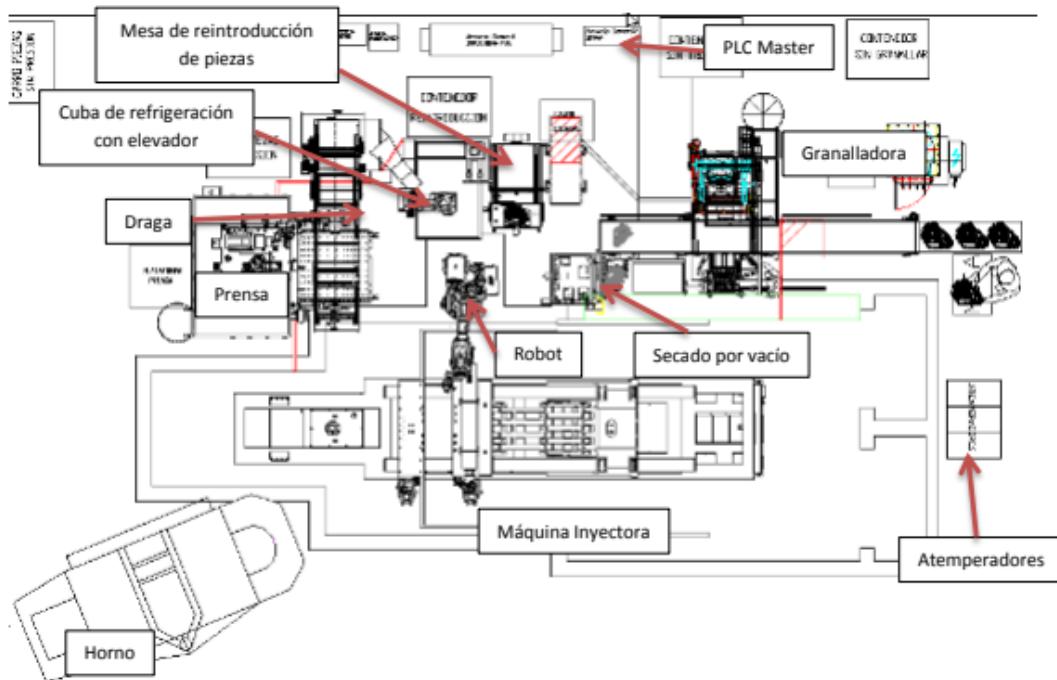


Figura 2.2. Mapa de la Célula 86.

La comunicación de los diferentes elementos se puede representar en el esquema que muestra la Figura 2.3.

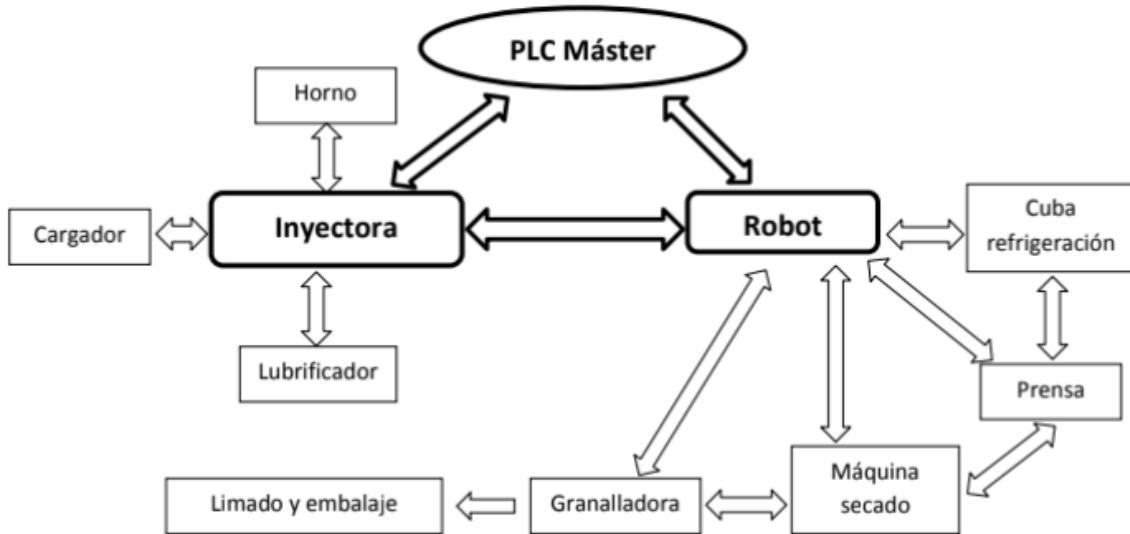


Figura 2.3. Esquema de comunicación Célula 86.

Como se puede observar, el **PLC Máster** es el que se encarga de comunicarse con los elementos principales (la máquina inyectora y el robot). La máquina se comunica con el grupo de inyección y el robot se encarga de comunicarse con el grupo de periféricos. La máquina y el robot también mantienen comunicación para seguir el proceso.

Para entender el proceso, es importante entender la función de los diferentes elementos que intervienen.

• PLC Máster

Se trata del elemento que se comunica con el robot y la máquina. Estos se comunican con todos los demás periféricos. Por lo tanto, el PLC Máster tiene acceso a todos los elementos que intervienen en el proceso.

Desde este dispositivo se dan órdenes y también se trata del intermediario del cual se recogen los datos de los sensores de la célula.



Figura 2.4. Cuadro de control del PLC Máster

• Máquina 86

La Máquina 86 es la responsable de inyectar el material en el molde. Se divide en dos partes: grupo de inyección y grupo de cierre. Para ver en detalle el funcionamiento de la máquina, ver el apartado **máquina** en [ANEXO A](#).

La máquina inyectora que se encuentra en la Célula 86 es marca IDRA OL 1600S [8]. Se trata de una máquina que contiene un panel de mandos, un cuadro general eléctrico y una interfaz para el PLC.

Desde la interfaz de usuario se pueden consultar y modificar los parámetros de proceso relacionados con la máquina de inyección y de sus periféricos. También se pueden realizar diagnósticos y gestionar y registrar alarmas derivadas del proceso.

La máquina también tiene diferentes modos de ciclo de trabajo:

- **Ciclo de montaje de moldes:** Los elementos se controlan de forma manual y las velocidades son reducidas. Esta fase sirve para montar moldes y hacer pruebas, así como mantenimiento de la propia máquina.
- **Ciclo manual:** Todos los movimientos de la máquina se controlan mediante los botones y utilizando los parámetros configurados.
- **Ciclo semiautomático:** Representa el modo de trabajo en condiciones normales. En este caso el proceso es automático, pero el operador tiene que dar la orden de re-start al finalizar el ciclo de la máquina.
- **Ciclo automático:** En este ciclo no interviene el operador, por lo que tienen que estar programados tanto la máquina como los elementos auxiliares para hacer todo el proceso de manera automática.
- **Horno de mantenimiento**

La función principal es mantener el material a la temperatura adecuada para el tipo de proyecto que se quiera construir.

Las paredes están formadas de un material refractario que evita que el material se oxide. También es crucial limpiar con frecuencia el horno para evitar impurezas.

- **Cargador lineal de metal**

El cargador es el dispositivo de 3 ejes que se encarga de recoger el aluminio del horno y transportarlo hasta la máquina para ser inyectado.

El eje de sistema de tornillo que se encarga de verter el material en la máquina se puede programar según las necesidades del proceso.

Máquina 86: IDRA OL 1600S



Figura 2.5. Máquina de Inyección IDRA OL 1600S



Figura 2.6. Horno de mantenimiento sin dosificador de la CEL86.



Figura 2.7. Cargador lineal de metal de la CEL86.

● Lubrificador

El lubrificador es un componente muy importante porque se encarga de enfriar el molde, de preparar el molde para el desmoldeo de la siguiente pieza, evitar la soldadura del aluminio sobre machos y de soplar el molde al final de soplar el molde a final de ciclo de lubrificación para evitar aguas residuales que puedan provocar poros.

Normalmente, los sistemas de lubrificación son a dos ejes (horizontal y vertical). Se utilizan boquillas pulverizadoras que pueden ajustarse y dosificarse en función de la zona a lubrificar. Dispone también de sistemas de comunicación entre la máquina y el lubrificador y sistemas de control directo sobre el lubrificador.

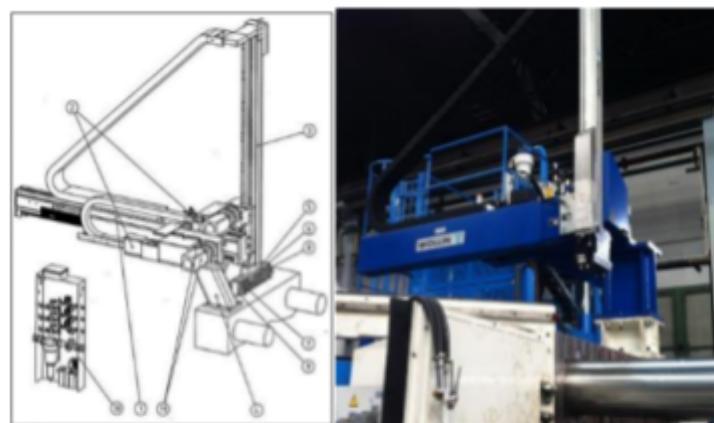


Figura 2.8. Sistema lubrificador de dos ejes de la CEL86.

● Atemperadores

Son dispositivos encargados de suministrar las líneas de refrigeración necesarias para tener un control térmico. Tienen dos modos de funcionamiento programables:

- Modo régimen: Con la máquina en marcha, funcionan como circuito de refrigeración
- Modo parada: Ante averías, funcionan para mantener el molde a temperatura óptima.



Figura 2.9. Atemperadores de CEL.86.

● Robot

El robot se encarga de retirar la pieza del molde y hacer todos los procesos auxiliares donde intervienen los periféricos.

Los procesos son: comprobar la integridad de la pieza, colocar la pieza en la cuba de refrigeración y llevar la pieza a la troqueladora, la máquina de

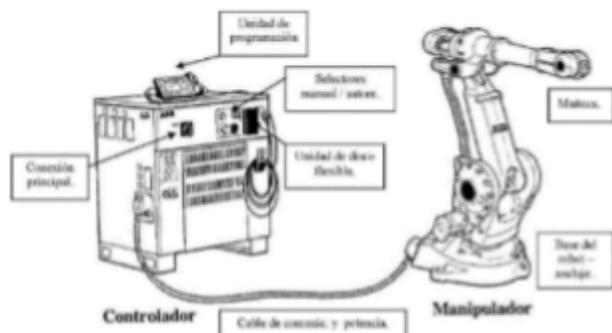


Figura 2.10. Sistema completo de robot: controlador y manipulador.

secado por vacío y a las cintas de la granalladora.

El robot está formado por 6 ejes que permite todos los grados de libertad (x-y-z). Tiene unas pinzas que son intercambiables dependiendo del proyecto.

La comunicación la establece con la máquina, el PLC y los periféricos. Es programable mediante un software.

- **Prensa**

La prensa de troquel es una máquina hidráulica que corta los elementos que no forman parte de la pieza (bebederos). Los cortes dependen de cada proyecto y la parte cortada se recupera para ser reutilizada.

La máquina contiene muchos elementos (radiales, sopladores, etc.) que funcionan de forma secuencial según el programa establecido.

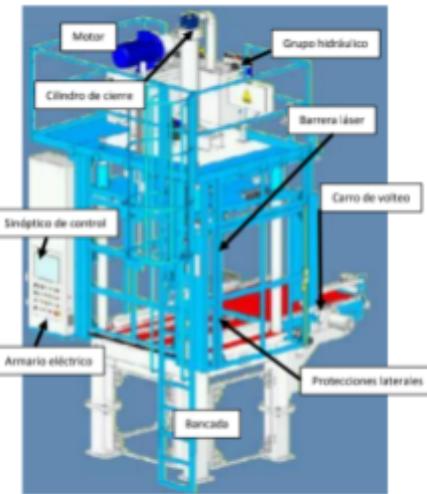


Figura 2.11. Prensa de troquel de la CEL86.

- **Cuba de refrigeración con elevador**

Su función es refrigerar la pieza que acaba de salir del molde. Se sumerge la pieza dentro de la bañera, obteniendo una refrigeración uniforme en toda la pieza.

Se puede programar la velocidad de bajada y el tiempo el cual la pieza se encuentra sumergida.



Figura 2.12. Cuba de refrigeración con elevador.

- **Máquina de secado por vacío**

La función principal es secar la pieza completamente para prepararla para el proceso de granallado. La pieza es introducida dentro de la cámara que genera una depresión que elimina el agua de la pieza.

Este proceso es importante para evitar que en la granalladora queden aguas o lubricantes que puedan afectar a la calidad de la pieza.



Figura 2.13. Máquina de secado por vacío.

- **Granalladora**

Se trata de un proceso en el cual se proyectan partículas esféricas a alta velocidad (200-250 Km/h) para hacer un tratamiento superficial, haciendo una limpieza superficial, eliminando pequeñas rebabas y dándole rugosidad a la pieza para soportar mejor futuros tratamientos.

Es importante controlar y filtrar correctamente el polvo que surge entre el impacto de la granalla y la pieza puesto a que este es altamente explosivo e inflamable.

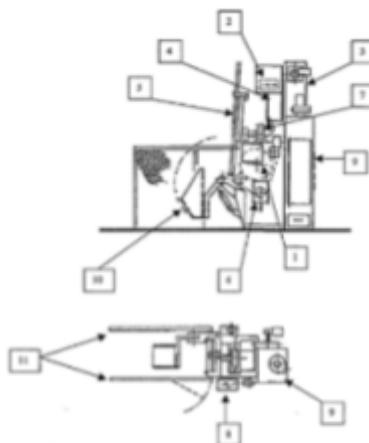


Figura 2.14. Esquema granalladora convencional

- **Mesa de reintroducción de piezas**

Este periférico sirve en caso de que alguna pieza salga del flujo normal de proceso. La pieza es posicionada en la mesa y el operario comunica al PLC Máster los procesos pendientes y el robot se encargará de llevarla a la parte o partes del proceso pendientes.

Un ciclo completo de inyección donde intervengan todos los elementos descritos anteriormente se compone de los siguientes pasos:

1. El cargador lineal de metal recoge el material fundido del horno de mantenimiento y lo lleva a la máquina para ser inyectado.
2. La máquina inyecta la aleación en el molde.
3. Una vez solidificado se abre el molde, el robot extrae la pieza para procesos posteriores y se lubrifica el molde.
4. El robot comprueba la integridad de la pieza mediante una fotocélula.
5. Despues, se encarga de llevar la pieza al útil de la cuba de refrigeración.
6. Posteriormente, el robot lleva pieza al proceso de prensado para eliminar los elementos que no forman parte de la pieza (bebederos)
7. Posteriormente, el robot coloca la pieza en la máquina de secado por vacío
8. Finalmente, el robot también coloca la pieza en la granalladora para acabados superficiales.



Figura 2.15. Máquina de secado por vacío.

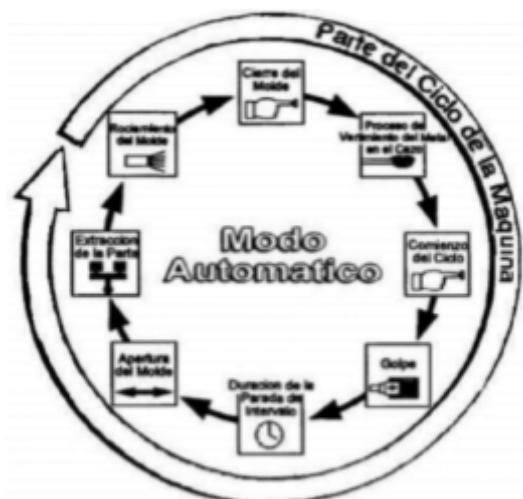


Figura 2.16. Ciclo de fabricación de una pieza por inyección.

Durante este proceso, los componentes que intervienen generan datos que pueden ser leídos desde el PLC Máster. Estos datos pueden describir el estado del proceso y también pueden ser utilizados para obtener información útil mediante minería de datos.

2.2. Objetivos

La Célula 86 ha sido sensorizada recientemente y es posible leer los valores de los diferentes sensores que forman parte de esta mediante el PLC Máster. Las señales de los sensores recogidas desde el PLC Máster son las siguientes:



Figura 2.17. Treemap de todas las señales disponibles en la CEL86 agrupadas (Plotly)

Como se puede observar en la Figura 2.17, hay 116 señales de sensores disponibles, las cuales están agrupadas arbitrariamente por ingeniería. Los datos de los sensores en un instante de tiempo determinado describen el estado de la célula.

Con estos datos son muchas las aplicaciones que se pueden realizar. Algunas de las aplicaciones más comunes son la de monitorización y estudio de evolución, así como otras aplicaciones que incluyen componentes predictivos.

En este proyecto hay dos objetivos principales que intentan demostrar la aplicación de minería de datos al proceso. Dichos objetivos son:

- El objetivo principal es aplicar minería de datos sobre los datos recogidos de los sensores para implementar el concepto de **mantenimiento predictivo**. Concretamente, el objetivo es obtener modelos de aprendizaje automático que aprendan del comportamiento de la célula y que sea posible detectar **anomalías**, así como **predecir en el futuro** el comportamiento de esta.
 - Por otra parte, aprovechando la información generada, el otro objetivo principal es crear una **aplicación** que efectúe: 1) El proceso de recolección de datos, 2) su procesado mediante modelos y 3) su visualización. La aplicación final será un visualizador que permite ver los resultados obtenidos del estudio efectuado.

La aplicación está ideada para el departamento de ingeniería. Concretamente, para que se puedan efectuar tareas de análisis y monitorización de los datos.

Ambos objetivos están relacionados entre sí, aunque se puede considerar que la minería de datos es una capa superior de proceso añadida a la aplicación. Finalmente, recalcar que ambos objetivos se encuentran dentro de la metodología de minería de datos estándar (ver apartado [3.1](#)).

3. MINERÍA DE DATOS

La minería de datos combina la informática, la matemática y el conocimiento del dominio para extraer información útil a través de datos.

Normalmente, la disciplina se aplica sobre grandes volúmenes de datos y el objetivo es descubrir patrones, dependencias, tendencias, grupos, etc. Con los últimos avances tecnológicos, es posible procesar cantidades abismales de información, pudiendo realizar cálculos y combinaciones de forma semiautomática o automática.

Mediante técnicas de aprendizaje automático, es posible crear modelos que resuman el conocimiento observado y se puedan utilizar para predecir eventos futuros, clasificar información, obtener patrones recurrentes, etc. Estos modelos pueden ahorrar costes y tiempo efectuando tareas que el ser humano sería incapaz o que serían inviables por el tiempo requerido. También pueden ser un gran soporte para el análisis de datos debido a que se basan en la estadística y la matemática para obtener resultados.

3.1. Metodología: CRISP-DM

La metodología que se aplica en este proyecto se llama CRISP-DM (*Cross Industry Standard Process for Data Mining*). Se trata de la metodología más utilizada para proyectos de minería de datos. Esta metodología consiste en 6 etapas, cuyo desarrollo no es lineal sino más bien iterativo.

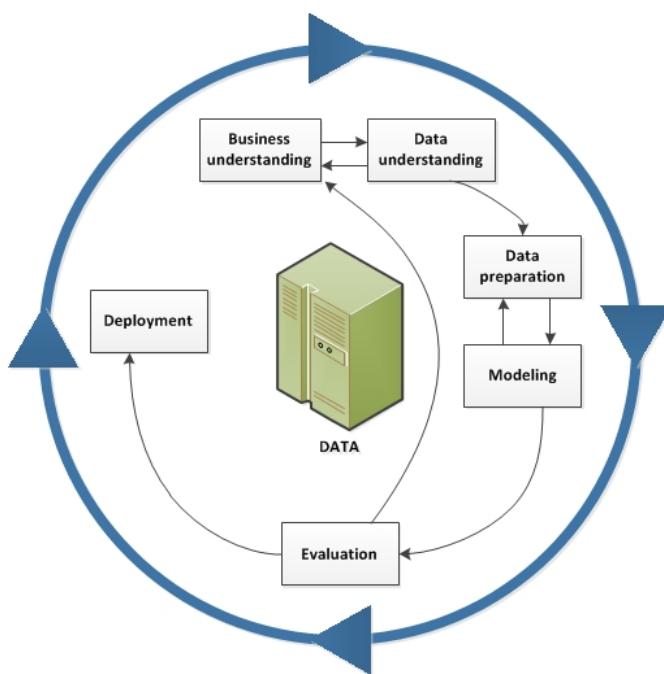


Figura 3.1. Esquema del ciclo de vida de un proyecto mediante la metodología CRISP-DM
[fuente: [\[52\]](#)]

- **Análisis de negocio:** Esta es la base y la parte esencial de cualquier proyecto, puesto que se basa en estudiar el negocio y cómo opera para encontrar esas preguntas que se quieren responder con el proyecto.
- **Análisis de datos:** Una vez se entiende el funcionamiento del negocio y se han generado las hipótesis a responder, es importante obtener los datos

necesarios para poder dar respuesta a esas preguntas y entender su naturaleza. Esta parte es muy relevante, puesto que la cantidad y calidad de los datos y el entendimiento de estos, serán la esencia del proyecto y lo que nos permitirá obtener mejores o peores resultados.

- **Preprocesado:** Con los datos recogidos, es relevante “limpiar” los datos y dejarlos listos para que los modelos sean capaces de entenderlos. Limpiar los datos consiste en valorar la calidad de estos, normalizarlos y hacer diferentes transformaciones con el objetivo de mejorar la calidad de estos.
- **Modelaje:** En esta etapa hay que estudiar diferentes modelos que expliquen el comportamiento de los datos, según el problema y los datos obtenidos. Es relevante valorar diferentes hipótesis, escenarios y parámetros para obtener resultados buenos y estables.
- **Evaluación:** En la etapa de evaluación se analizan los resultados de los diferentes modelos utilizados y se llevan a cabo diferentes comparativas y simulaciones. En función del análisis, se tendrá que valorar si los resultados son lo suficientemente buenos y estables o no son suficientes, por lo que habría que volver a repetir los pasos anteriores.
- **Despliegue:** Una vez se tiene un buen modelo, hará falta desarrollar la plataforma que aplique los diferentes pasos para implementar la solución.

Esta metodología es flexible y permite tener una concepción de las diferentes etapas necesarias para llevar a cabo un proyecto de este tipo. En este proyecto se ha intentado seguir esta metodología y en los siguientes apartados se describen las decisiones y el desarrollo de las diferentes etapas.

3.2. Análisis de negocio

Antes de empezar cualquier proyecto de minería de datos, es vital entender el contexto del negocio y estudiar las posibles soluciones que se podrían aportar mediante minería de datos.

Al estudiar las soluciones, también es importante responder a las preguntas relacionadas con viabilidad y recursos. Es posible encontrar una muy buena solución, pero no disponer de la información o recursos necesarios para obtener resultados o, simplemente por dinero o tiempo, no es posible llegar a desarrollarla. En este caso, la experiencia y el tiempo dedicado al estudio juegan un factor clave.

Este proyecto tiene un título genérico debido a que en el momento de hacer el registro, todavía se estaba estudiando la viabilidad de diferentes soluciones. Sabía que el proyecto se iba a situar en la Célula 86, pero no estaba claro a qué hipótesis se iba a intentar dar respuesta.

Las ideas principales fueron: el mantenimiento predictivo, la calidad predictiva y el estudio multivariado de parámetros de proceso para encontrar los más influyentes en el funcionamiento de la máquina.

La calidad predictiva requiere la trazabilidad de la pieza durante el proceso y al pasar los controles de calidad. Esta premisa requiere también que el muestreo sea lo suficientemente grande. Sin embargo, los controles de calidad son relativamente limitados respecto al gran volumen de muestras requeridas para obtener buenos resultados.

En cuanto al análisis de los parámetros más influyentes, se trata de una tarea complicada de la cual era difícil encontrar un enfoque adecuado. No obstante, podía formar parte de la optimización de otra solución debido a que está relacionado con la ingeniería de características.

Finalmente, la opción más viable con los datos que se podían obtener, es la del mantenimiento predictivo, que además podía suponer una buena herramienta para mejorar el rendimiento de la máquina. Aun así, es importante destacar que se tuvo en cuenta que se trataba de un objetivo donde es difícil obtener buenos resultados debido a la dimensionalidad del problema.

3.3. Análisis de datos

El paso inicial al aplicar minería de datos es estudiar los datos y entender su naturaleza. La exploración de datos es fundamental para las próximas decisiones, puesto que permite entender los datos para realizar hipótesis.

El estudio se ha efectuado mediante la herramienta *Google Colab*. Esta permite ejecutar código en lenguaje Python utilizando un formato de libreta interactiva. Esto lo hace un entorno ideal para el estudio de datos. Se puede visualizar la libreta empleada mediante en el enlace de la referencia [9].

El estudio y modelaje inicial se efectuará sobre una muestra de aproximadamente 24 horas extraída de la base de datos. También es importante recalcar que para simplificar el análisis y modelaje en este proyecto, solo se tienen en cuenta las señales **críticas** (ver [Anexo B](#)), definidas arbitrariamente por ingeniería. No obstante, el proceso de análisis puede ser extrapolado al total de las 116 señales en un futuro. El dataset de la muestra descrita tiene la estructura mostrada en la Figura 3.2.

```
[ ] críticas.info()

<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 86042 entries, 2022-06-13 07:10:04.064000+00:00 to 2022-06-14 07:10:03.069000+00:00
Data columns (total 17 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Caudal_Piston    86042 non-null   int64  
 1   Flow_CIR_1        86042 non-null   float64 
 2   Flow_CIR_2        86042 non-null   float64 
 3   Flow_CIR_3        86042 non-null   float64 
 4   Grueso_Colada    86042 non-null   float64 
 5   Pres_Entrada_Agua_Maquina 86042 non-null   float64 
 6   Pres_Entrada_Aire 86042 non-null   float64 
 7   Pres_Final_Mult  86042 non-null   int64  
 8   Pres_Maxima_Mult 86042 non-null   int64  
 9   Pres_Retorno_Agua_Maquina 86042 non-null   float64 
 10  Pres_Retorno_Agua_Holde 86042 non-null   float64 
 11  Temp_Cuba         86042 non-null   float64 
 12  Temp_Horno        86042 non-null   float64 
 13  Tiempo_Subida_RT 86042 non-null   float64 
 14  Vel_1a_Fase_Media 86042 non-null   float64 
 15  Vel_2a_Fase_Maxima 86042 non-null   float64 
 16  Vel_2a_Fase_Media 86042 non-null   float64 
dtypes: float64(14), int64(3)
memory usage: 11.8 MB
```

Figura 3.2. Muestra de 24 horas de señales críticas.

Como se puede observar, la muestra es del 13/06/2022 al 14/06/2022 y contiene 86042 muestras de los sensores con una frecuencia de muestreo de 1 segundo.

También se puede observar que no hay valores nulos. No obstante, es importante saber tratarlos porque es posible que haya fallos de comunicación y se pierda información.

Otro factor relevante es ver que no hay información categórica que permita clasificar la muestra. Este factor es relevante para el modelaje.

	Caudal_Piston	Flow_CIR_1	Flow_CIR_2	Flow_CIR_3	Grueso_Colada	Pres_Entrada_Agua_Maquina	Pres_Entrada_Aire	Pres_Final_Mult	Pres_time
2022-06-13 07:10:04.084000+00:00	218	0.587246	2.570288	2.495488	53.910034	5.684	6.820	258	
2022-06-13 07:10:05.083000+00:00	217	0.587246	2.570288	2.495488	53.910034	5.684	6.832	258	
2022-06-13 07:10:06.067000+00:00	219	0.587246	2.570288	2.495488	53.910034	5.684	6.832	258	
2022-06-13 07:10:07.069000+00:00	217	0.587246	2.570288	2.495488	53.910034	5.684	6.844	258	
2022-06-13 07:10:08.077000+00:00	218	0.587246	2.570288	2.495488	53.910034	5.684	6.844	258	

Figura 3.3. Ejemplo de la muestra.

En la Figura 3.3, se puede confirmar que se trata de un dataset de **naturaleza temporal**, ya que se trata de un conjunto de muestras/datos ordenados cronológicamente observados en distintos momentos, y **multivariante** por el hecho de que hay diferentes variables a observar.

3.3.1. Análisis de series temporales

Los datos de los sensores de la máquina son de naturaleza **temporal** debido a que son un conjunto de mediciones que se realizan periódicamente en el tiempo. Este tipo de series de datos se pueden encontrar en muchos ámbitos, como el mercado de stocks, datos meteorológicos, sensores, etc.

Los datos de tipo *timestamp* se pueden clasificar en dos grandes categorías:

- **Stock time series data:** Se obtienen los datos en un instante de tiempo determinado. Es decir, se efectúa una foto del estado en un determinado momento.
- **Flow time series data:** Se obtienen datos durante un periodo de tiempo determinado.

Por ejemplo, en el marco de este proyecto, se podría efectuar un estudio de cómo evolucionan las señales de la máquina durante un periodo de tiempo (*flow time series*) o se podrían recoger los datos cada vez que la máquina hace la inyección (*stock time series*). Todo depende del objetivo que se quiera lograr. En este caso, se trata de un muestreo durante un periodo de tiempo (*flow time series*).

Un primer paso para hacer el análisis es graficar la información en el tiempo. De este proceso se puede extraer mucha información importante y empezar a formular hipótesis.

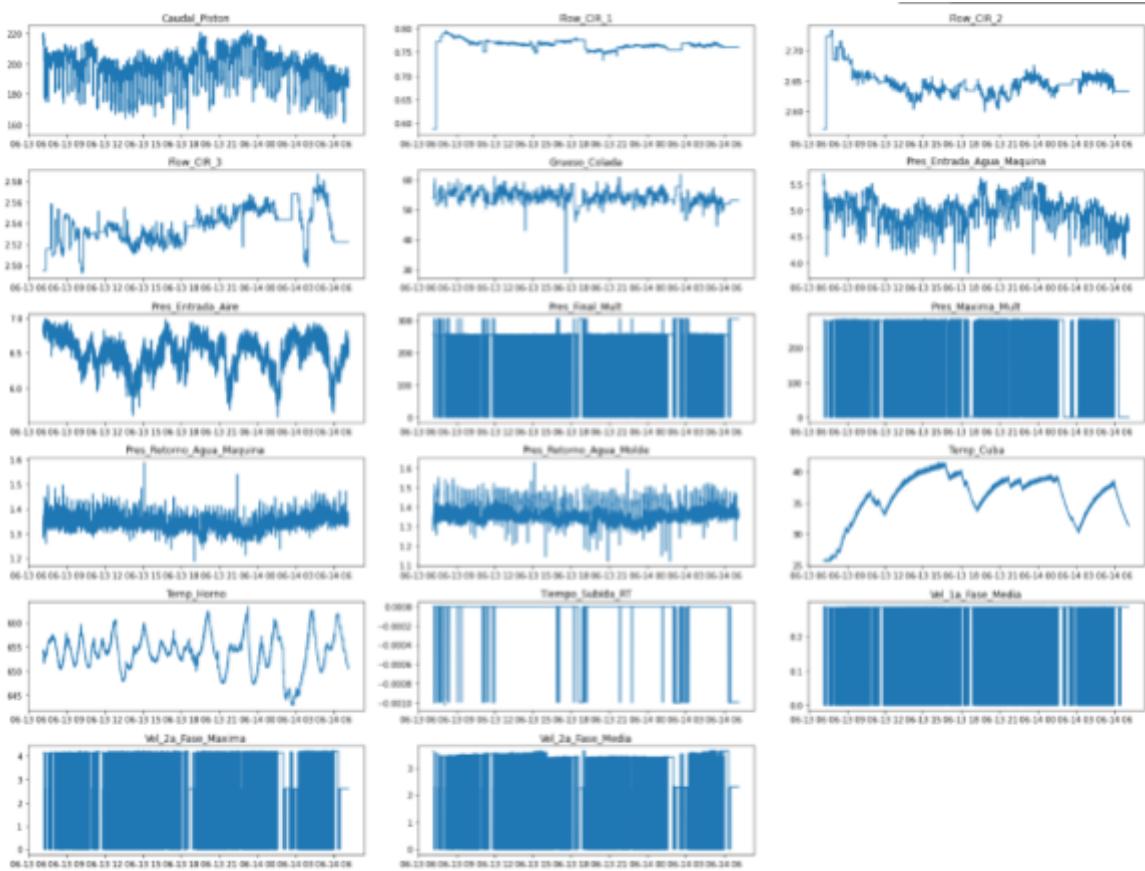


Figura 3.4. Gráficas temporales de la muestra de señales críticas.

En la Figura 3.4, se han graficado todos los puntos del dataset en el tiempo, obteniendo la evolución de los sensores. Como se puede observar, hay muchos sensores con comportamiento muy homogéneo y también hay algunos comportamientos irregulares que podrían ser anomalías.

Para entender mejor las características de las gráficas obtenidas, se ha escogido un sensor de ejemplo para estudiar sus propiedades. El sensor escogido es la Temperatura del horno que tiene la siguiente evolución:

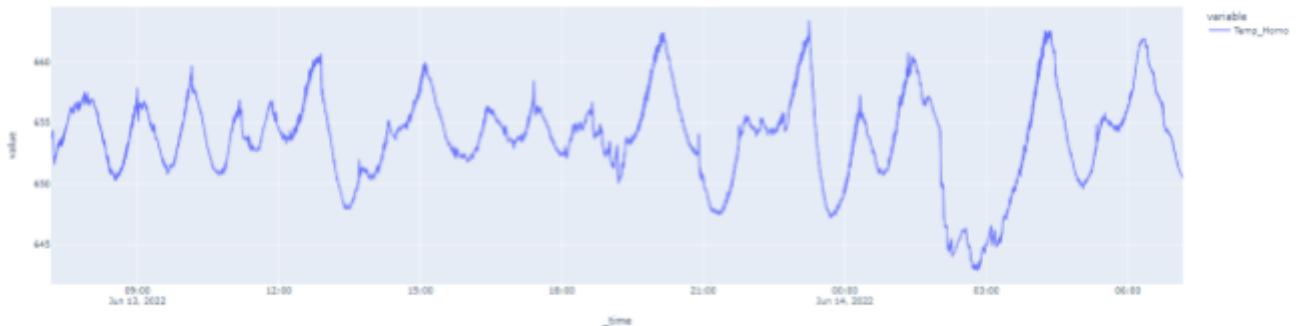


Figura 3.5. Gráfica temporal de la variable Temp_Horno.

Algunas de las propiedades principales a tener en cuenta al analizar los datos de tipo `timestamp` son:

- **Tendencia:** Representa el comportamiento general durante el tiempo.
- **Estacionalidad:** Patrones que se repiten frecuentemente en intervalos regulares de tiempo.
- **Componente Cíclica:** Patrones repetitivos sin tiempo fijo.
- **Irregularidad / Ruido:** Variaciones en las mediciones en los patrones observados.
- **Autocorrelación:** Indica qué tan correlacionado está un punto concreto de la medición con las propias mediciones pasadas.

	Trend	Seasonality	Cyclical	Irregularity
Time	Fixed Time Interval	Fixed Time Interval	Not Fixed Time Interval	Not Fixed Time Interval
Duration	Long and Short Term	Short Term	Long and Short Term	Regular/Irregular
Visualization				
Nature - I	Gradual	Swings between Up or Down	Repeating Up and Down	Errored or High Fluctuation
Nature – II	Upward/Down Trend	Pattern repeatable	No fixed period	Short and Not repeatable
Prediction Capability	Predictable	Predictable	Challenging	Challenging
Market Model				Highly random/Unforeseen Events – along with white noise.

Designed by Author (Shanthababu)

Figura 3.6. Componentes de una serie temporal (fuente: [\[10\]](#)).

Mediante el módulo `statmodels` de Python es posible descomponer una gráfica para mostrar algunas de las características mencionadas. Aplicando este método a la gráfica, estableciendo una estacionalidad de 1 minuto (tiempo de ciclo de inyección habitual) se obtiene la siguiente descomposición que puede verse en la Figura 3.7.

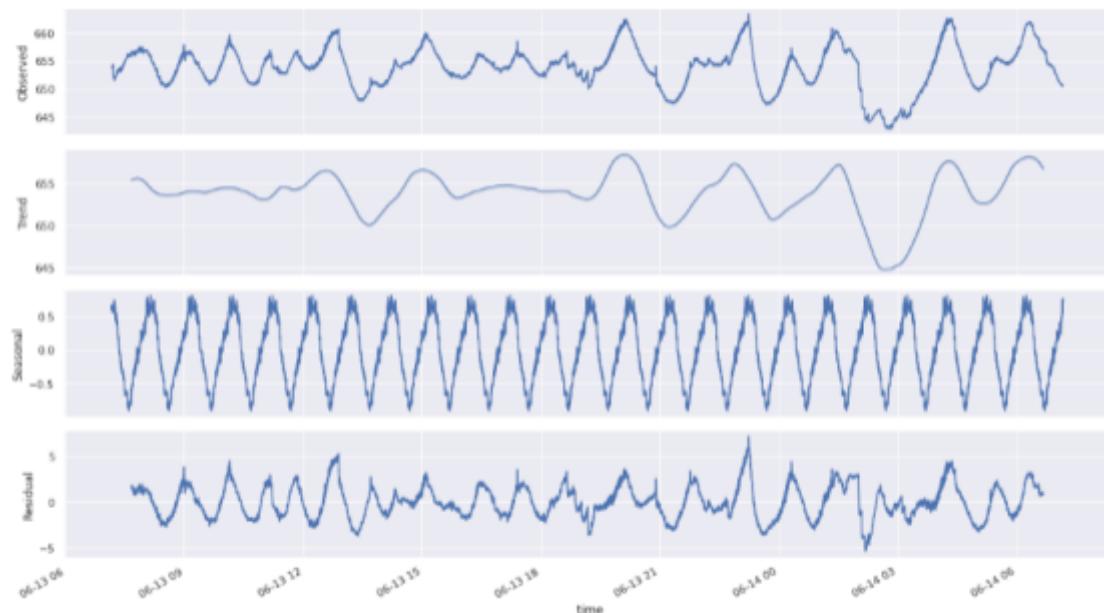


Figura 3.7. Descomposición de la serie temporal de Temp_Horno.

Se puede observar que la gráfica está descompuesta en tendencia, estacionalidad y ruido. Examinando la estacionalidad se puede percibir la periodicidad de los eventos y, en cuanto al ruido, también se puede observar que es bastante regular.

Para verificar la estacionalidad hay diferentes métodos estadísticos que lo comprueban. Por ejemplo, el *Augmented Dickey-Fuller Test* (ADF) [12] o el *Kwiatkowski-Phillips-Schmidt-Shin Test* (KPSS) [13]. En este caso, se utilizará el test Dickey-Fuller para identificar la estacionalidad en el sensor *Temp_Horno*:

	Values	Metric
0	-1.890343	Test Statistics
1	0.336618	p-value
2	64.000000	No. of lags used
3	85977.000000	Number of observations used
4	-3.430426	critical value (1%)
5	-2.861574	critical value (5%)
6	-2.566788	critical value (10%)

Figura 3.8. Resultados del Augmented Dickey-Fuller Test sobre Temp_Horno.

Como se puede observar en la Figura 3.8 el valor de **p es mayor a 0.05** por lo que se falla en rechazar la hipótesis de raíz unitaria y se concluye que la temperatura del horno no es estacionaria.

Aunque los datos no sean estacionarios, se pueden convertir datos no estacionarios a estacionarios mediante conversiones. Algunas de ellas son:

- **Detrending:** Eliminar la tendencia para observar solo la diferencia en valores.
- **Differencing:** Transformar los datos en una nueva serie de tiempo donde se estabiliza la media.
- **Transformation:** Hay diferentes metodologías como la transformación en potencia, raíz cuadrada o logarítmica.

Aplicando *detrending* sobre la temperatura del horno se obtiene el siguiente resultado:

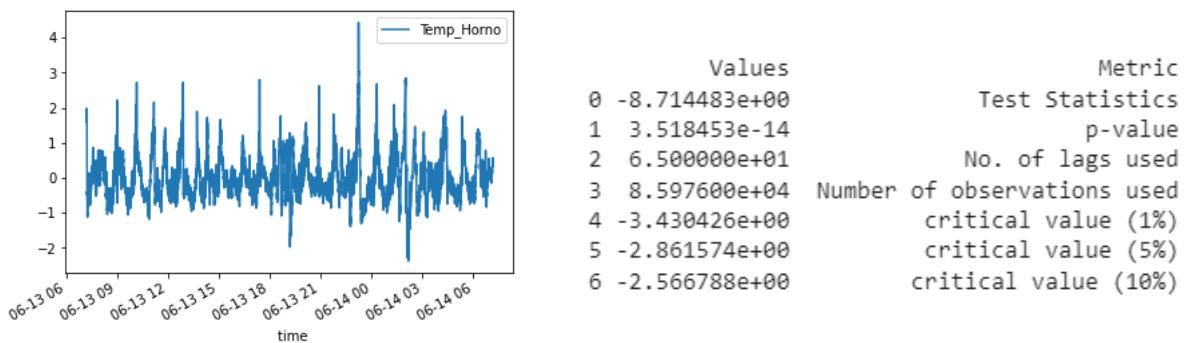


Figura 3.9. Resultados de aplicar la técnica de *detrending* sobre temp Horno.

Como se puede observar en la Figura 3.9, al eliminar la tendencia se obtiene un valor de **p menor a 0.05** en el test ADF por lo que se consigue que sea estacionaria. Algunos modelos como ARIMA/SARIMA [14] parten de esta premisa, así que estos pasos formarían parte del preprocesado.

Otra característica que se puede verificar haciendo uso de métodos del módulo `statmodels`, es la autocorrelación de las muestras.

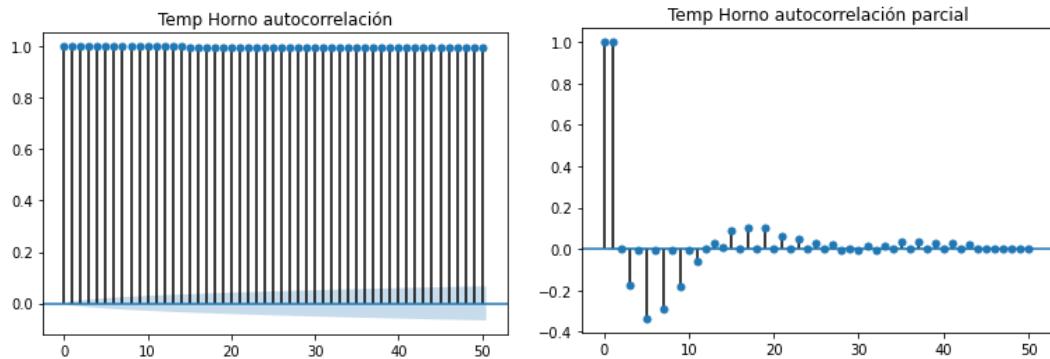


Figura 3.10. Gráfica de autocorrelación y autocorrelación parcial sobre Temp_Horno.

La Figura 3.10 muestra que la autocorrelación es altísima en el gráfico de la izquierda debido a que es calculada en función de todos los puntos anteriores de forma lineal. En el gráfico de la autocorrelación parcial, se elimina la autocorrelación acumulada, por lo que se puede observar mejor la continuidad. Algunos modelos como los vectores autorregresivos (ver [Anexo C](#)) se basan en estos factores, por lo que es interesante realizar un análisis previo.

3.3.2. Análisis multivariado

Otra de las características principales de los datos con los que se trabaja es que se trata de un dataset multivariado, debido a que hay que tener en cuenta múltiples variables diferentes. Estudiar cada una y la relación entre ellas, permite tener una mejor conciencia sobre los datos y puede habilitar a tomar algunas decisiones de preprocesado. Algunas estadísticas del dataset utilizado son las siguientes:

	Caudal_Piston	Flow_CIR_1	Flow_CIR_2	Flow_CIR_3	Grueso_Colada	Pres_Entrada_Agua_Maquina	Pres_Entrada_Aire	Pres_Final_Mult
count	86042.000000	86042.000000	86042.000000	86042.000000	86042.000000	86042.000000	86042.000000	86042.000000
mean	196.169824	0.764349	2.645786	2.537213	54.232913	4.954380	6.505865	220.698101
std	11.797493	0.019085	0.022159	0.016608	2.462766	0.297291	0.238552	98.701425
min	157.000000	0.587246	2.570288	2.492890	28.830017	3.796000	5.580000	0.000000
25%	189.000000	0.761541	2.632976	2.523827	52.890015	4.752000	6.368000	256.000000
50%	198.000000	0.765691	2.643872	2.536890	54.239990	4.980000	6.536000	257.000000
75%	204.000000	0.771197	2.653531	2.548286	55.820007	5.160000	6.688000	258.000000
max	222.000000	0.796118	2.733320	2.586644	61.890015	5.696000	6.992000	307.000000

Figura 3.11. Resumen estadístico de la muestra de datos.

La tabla de la Figura 3.11. muestra que las variables tienen distintas estadísticas teniendo en cuenta que algunas también tienen magnitudes diferentes.

Esto implica el hecho de que hacer un buen análisis multivariado es una tarea compleja, aunque hay algunas técnicas estadísticas populares que facilitan la tarea. En los siguientes puntos se revisan algunas de estas técnicas.

- Análisis correlativo

Realizar un análisis correlativo siempre es buena idea debido a que se pueden extraer algunas características importantes de los datos y sus relaciones.

La correlación de dos variables es la relación entre la evolución del movimiento de dos variables. Normalmente, viene expresada mediante un coeficiente y el método más estandarizado es el coeficiente de Pearson [15].

El coeficiente de Pearson, se puede definir como la covarianza entre dos variables dividida entre el producto de las desviaciones estándar de estas. El resultado está estandarizado entre 1 y -1 indicando una correlación positiva o negativa, respectivamente.

Mediante el módulo pandas y el módulo de visualización seaborn, se ha creado la siguiente matriz donde se pueden obtener los valores de los coeficientes entre todas las variables.

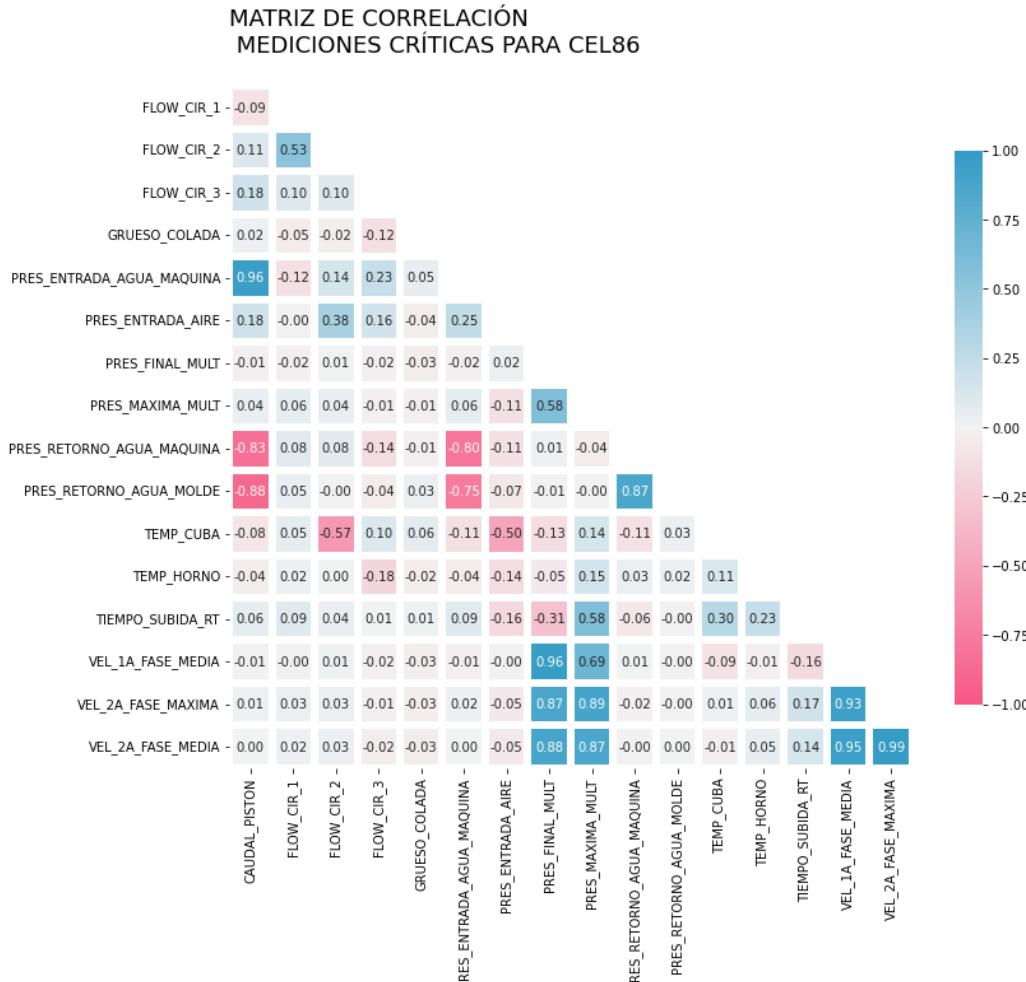
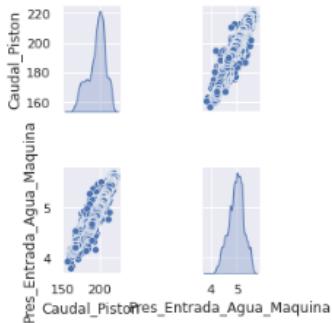


Figura 3.12. Matriz de correlación de la muestra.

Como se puede ver en la matriz de la Figura 3.12 hay varias variables con correlaciones muy altas, sobre todo velocidades y presiones. En la Figura 3.4, ya se podía visualizar que dichas variables presentaban comportamientos muy similares.

Una buena forma de inspeccionar las correlaciones visualmente es mediante los *pairplots*.



En la Figura 3.13 se puede ver como tanto la distribución de los valores como la densidad son muy similares por lo que esto explica la alta correlación entre las variables Caudal_Piston y Pres_Eentrada_Agua_Maquina (0.96).

Aunque no es el caso, podría darse el hecho de que se trataran de la misma variable medida con sensores diferentes. En estos casos, se podrían tomar decisiones como aplicar técnicas de reducción de dimensionalidad combinando variables correlacionadas.

Figura 3.13. Pairplot entre la variable *Caudal_Piston* y *Pres_Eentrada_Agua_Maquina*

- **Análisis de componentes principales**

Una técnica muy popular para reducir la complejidad o dimensionalidad de un conjunto de datos, conservando la información, es el Análisis de Componentes Principales (PCA, *Principal Component Analysis*). Esta técnica consiste en obtener, mediante combinaciones lineales, las características que expliquen mejor el conjunto de datos y reduciendo aquellas que no aporten información.

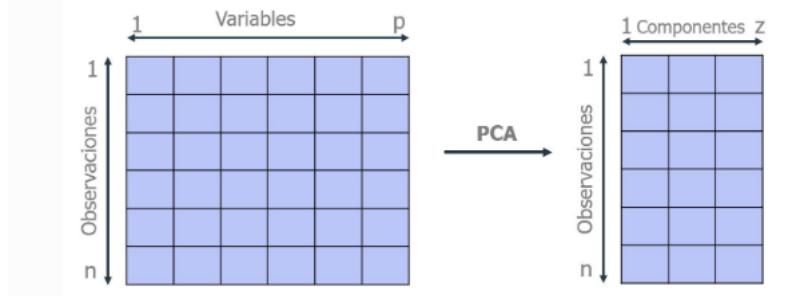


Figura 3.14. Transformación de variables a componentes (fuente: [17]).

Las componentes resultantes del algoritmo corresponden a los conceptos matemáticos de *eigenvectors* y *eigenvalues*, ordenándolas de manera que la primera componente es el *eigenvector* con el *eigenvalue* más alto.

Cada componente principal se especifica como la combinación lineal de las variables originales, normalizadas, con mayor varianza.

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \dots + \phi_{p1}X_p$$

Figura 3.15. Ecuación de un componente principal (fuente: [17]).

En la ecuación de la Figura 3.15, Z_1 representa la primera componente y esta viene dada por la suma de la combinación lineal entre la variable y un peso (*loading*) que indica la importancia de dicha variable.

El proceso para construir estas componentes se basa en restar el valor de la media de una variable a ella misma para que todas las variables tengan media 0. Despues, se asigna el *loading* resolviendo un problema de optimización en el que el objetivo es maximizar la varianza. La intuición geométrica del resultado del análisis es la siguiente:

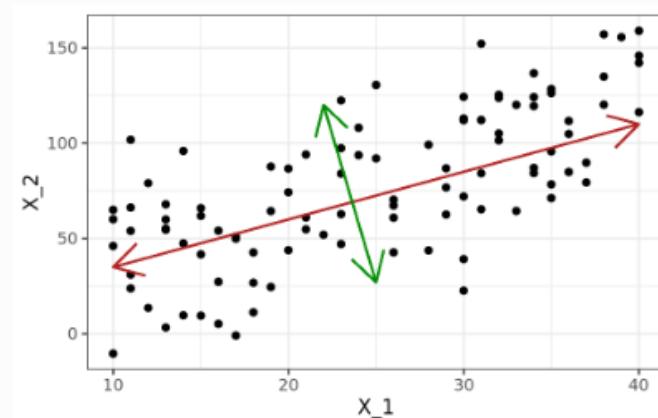


Figura 3.16. Ejemplo de representación geométrica de dos componentes principales (fuente: [17]).

Como se puede observar en la Figura 3.16, hay dos vectores que representan los diferentes componentes. El vector rojo sigue la línea de las observaciones con más varianza. El vector verde representa la segunda componente y se tratan de los valores con más varianza, pero que no están correlacionados con la primera componente.

Aplicando el análisis de componentes principales sobre el *dataset* se obtienen los siguientes resultados mostrados en la Figura 3.17.

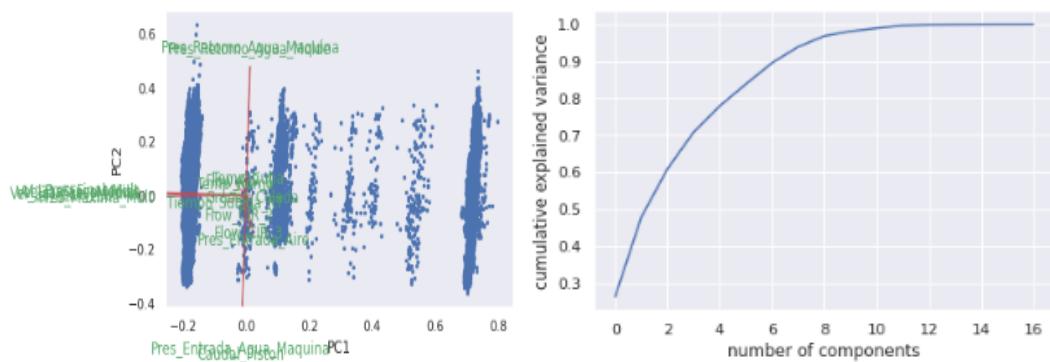


Figura 3.17. Resultados obtenidos al aplicar PCA sobre la muestra.

Como se puede observar en la gráfica de la derecha de la Figura 3.17, con 6 componentes se puede explicar el 90% de la varianza del problema, por lo que se podría reducir la información de 17 variables a 6 componentes principales.

Seleccionando las 6 primeras componentes, se pueden ver las variables que más influyen en cada componente mediante el mapa de calor de la Figura 3.18.

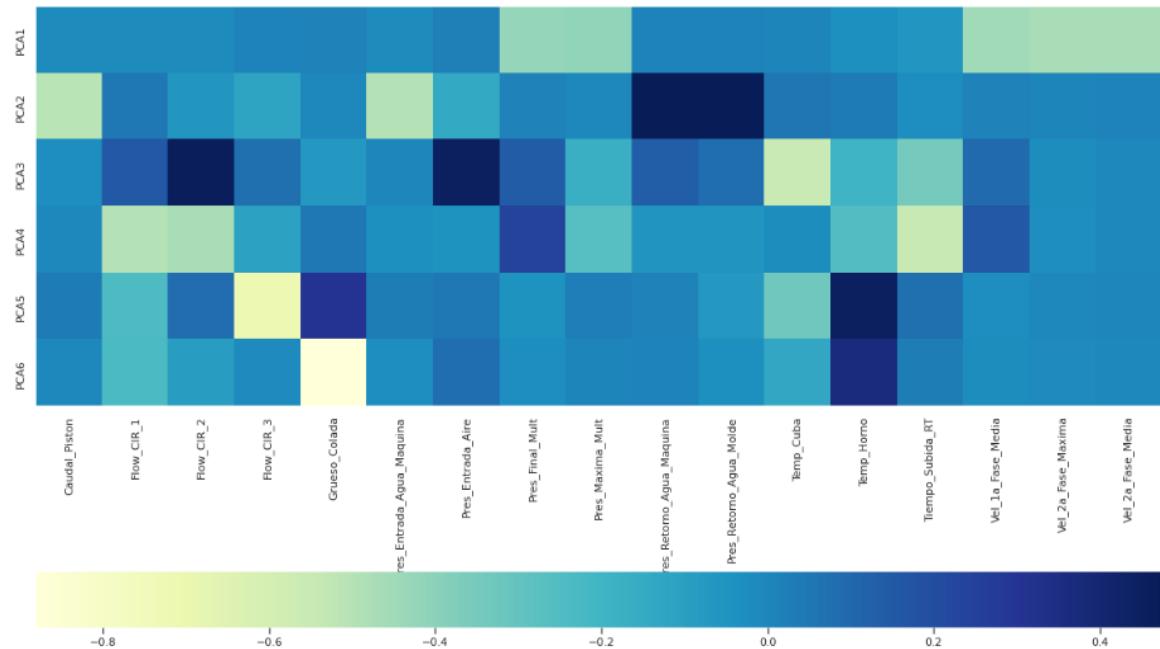


Figura 3.18. Mapa de calor que indica la variabilidad de las variables en los componentes.

El mapa de calor (*heat map*) indica que la componente principal 1 está formada por varianzas homogéneas por parte de las variables que la constituyen. La componente 2 tiene las variables Pres_Retorno_Agua_Maquina y Pres_Retorno_Agua_Molde como características principales.

Este análisis es muy interesante para la ingeniería de características, un componente del preprocesado donde el objetivo es encontrar qué características explican mejor el problema.

- ***Clustering***

El análisis de clústeres o *clustering* es una de las técnicas más utilizadas en la minería de datos. Esta técnica permite dividir la información asignando los datos a grupos de datos o clústeres. Esto permite estructurar la información y poder identificar grupos de datos que posiblemente se puedan formar teniendo en cuenta el contexto del problema.

Actualmente, hay distintos tipos de algoritmos de agrupamiento disponibles para aplicar esta técnica. Estos algoritmos se dividen, principalmente, en las siguientes tipologías:

- **Particionales:** Algoritmos que dividen los datos en grupos independientes donde cada punto solo puede formar parte de un grupo.
- **Jerárquicos:** En este caso, se construye un árbol jerárquico llamado *dendograma*. Los grupos se ordenan jerárquicamente de forma que los datos que pertenecen a un grupo también pertenecen al grupo de su antecesor. El árbol se puede cortar a una cierta profundidad (valor elegible) obteniendo así la jerarquía de formación de grupos.

- **Densidad:** Se basan en la densidad de los datos para formar grupos. Se especifica la distancia mínima entre los puntos para que estos formen un grupo.

En este proyecto se utilizará el método particular *K-means* [18]. El motivo de escoger este método es su relación velocidad-eficacia y la posibilidad de escalar a grandes volúmenes de datos.

El algoritmo *K-means* requiere la especificación del número de clústeres (*k*) que tiene que formar. Una vez especificado el número de clústeres empieza el siguiente proceso:

1. Se inicializan los centroides de manera aleatoria.
2. Se asignan los puntos más próximos al centroide más cercano.
3. Despues, se calcula la media de los puntos asignados a cada centroide repositionando este al valor obtenido.
4. Se vuelve a iterar el proceso.

Este proceso se realiza continuamente hasta que la posición de los centroides converge y no se vuelven a reposicionar. Es un proceso iterativo que escala muy bien en grandes cantidades de datos.

Uno de los principales retos que tiene el aplicar este algoritmo es escoger correctamente el número de clústeres a formar. Hay diferentes métodos para escoger este valor. Los principales son el método *elbow* [19] o el análisis de silueta. En el método *elbow* se iteran sobre diferentes valores de *k* y se grafican los resultados escogiendo el valor de *k* en el momento en que la gráfica hace la forma de codo.

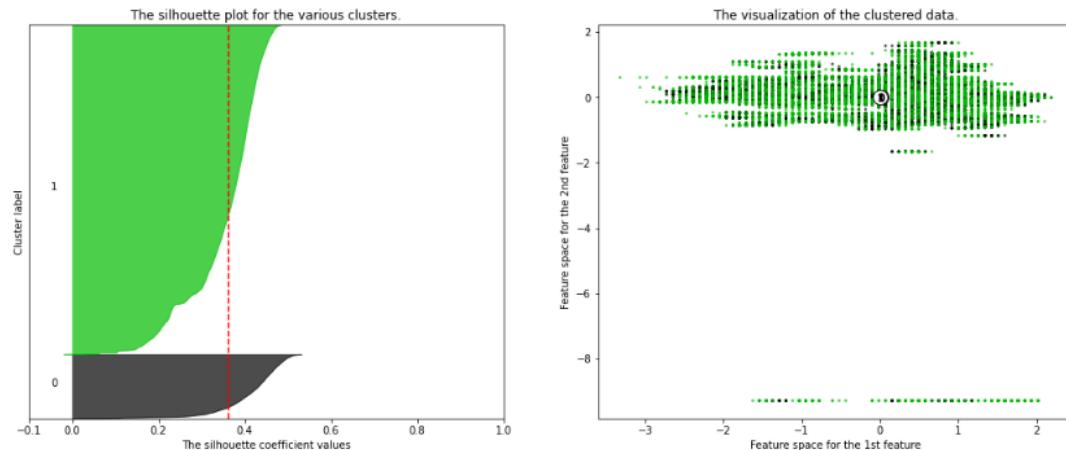
En este proyecto se utiliza el método de la silueta debido a que también permite visualizar la cohesión y separación de los clústeres. Aplicando *K-means* mediante el módulo *scikit-learn* de Python [20] con un valor de *k* iterativo se obtienen los siguientes resultados:

```

For n_clusters = 2 The average silhouette_score is : 0.36250572831404243
For n_clusters = 3 The average silhouette_score is : 0.22645784769535224
For n_clusters = 4 The average silhouette_score is : 0.25933448603645937
For n_clusters = 5 The average silhouette_score is : 0.27350705033266837
For n_clusters = 6 The average silhouette_score is : 0.28252342272759673
For n_clusters = 7 The average silhouette_score is : 0.23776335358263698
For n_clusters = 8 The average silhouette_score is : 0.2184692158445888
For n_clusters = 9 The average silhouette_score is : 0.2126912636626093
For n_clusters = 10 The average silhouette_score is : 0.22308677649312128

```

Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



Silhouette analysis for KMeans clustering on sample data with n_clusters = 3

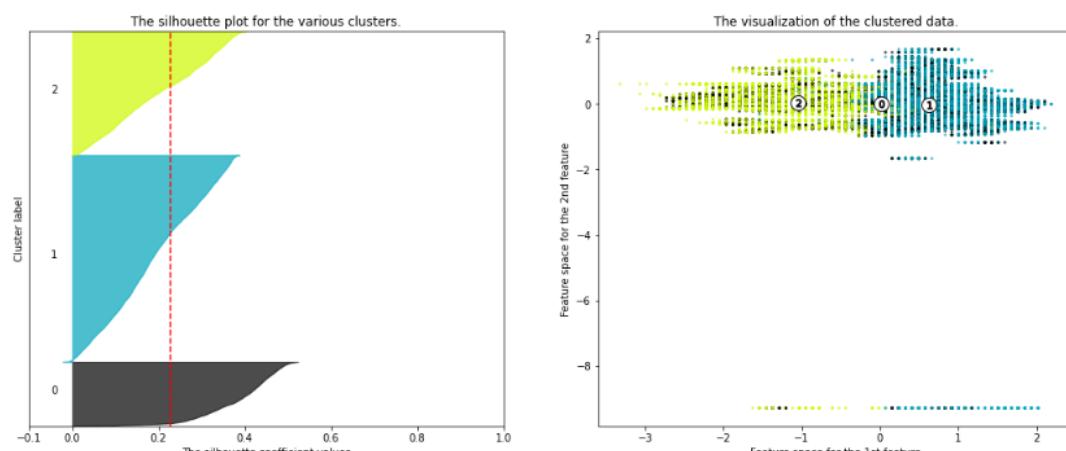


Figura 3.19. Aplicación de K-Means y análisis de silueta sobre la muestra.

En la Figura 3.19, se observa que la mejor puntuación de silueta la ha obtenido el valor de $k=2$, separando los clústeres en dos grupos principales. Esta puntuación considera la distancia entre un punto y la media de todos los otros puntos de un mismo clúster (distancia inter-clúster) y también la distancia entre el punto y la media de todos los puntos del clúster más cercano. El valor varía entre -1 y 1, siendo 1 un clúster denso y bien separado de otros clústeres.

A parte de la puntuación, viendo el gráfico de la silueta, se puede decidir el valor óptimo de k . Siendo $k=2$ el que ha obtenido la mejor puntuación, se puede observar que la anchura de la silueta de un clúster es mucho mayor que la del otro clúster. Estableciendo $k=3$, se obtienen anchuras más repartidas y todas pasan de la media.

Para tomar la decisión será necesario estudiar los datos de los clústeres o aplicar un paso de preprocesado para ver si se trata de posible ruido en la información.

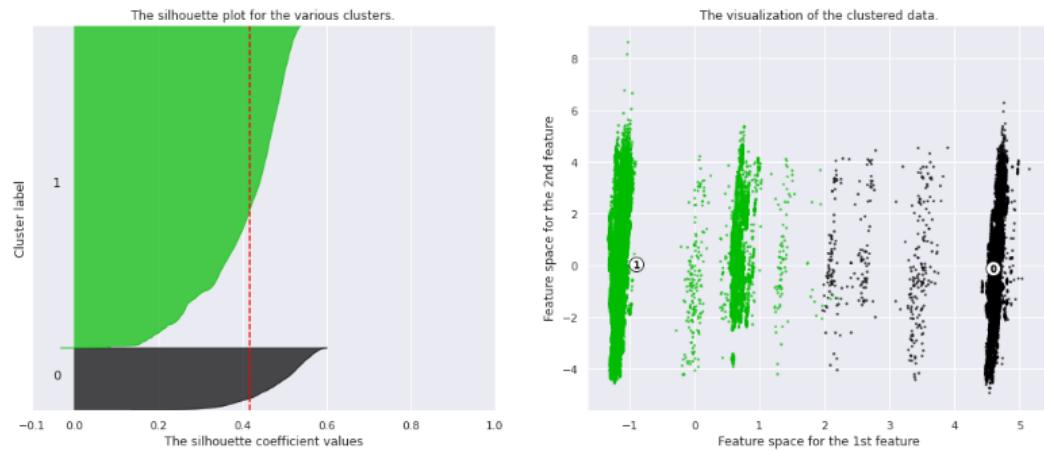
Aplicando PCA con 6 componentes **antes de** realizar el clustering, se obtienen los resultados mostrados en la Figura 3.19.

```

For n_clusters = 2 The average silhouette_score is : 0.4183373992761845
For n_clusters = 3 The average silhouette_score is : 0.2776555125310013
For n_clusters = 4 The average silhouette_score is : 0.31509858090357892
For n_clusters = 5 The average silhouette_score is : 0.3348152018321715
For n_clusters = 6 The average silhouette_score is : 0.34579476583578234
For n_clusters = 7 The average silhouette_score is : 0.29456151756490095
For n_clusters = 8 The average silhouette_score is : 0.2741248910071771
For n_clusters = 9 The average silhouette_score is : 0.2687322857678117
For n_clusters = 10 The average silhouette_score is : 0.2691370967820705

```

Silhouette analysis for KMeans clustering on sample data with n_clusters = 2



Silhouette analysis for KMeans clustering on sample data with n_clusters = 3

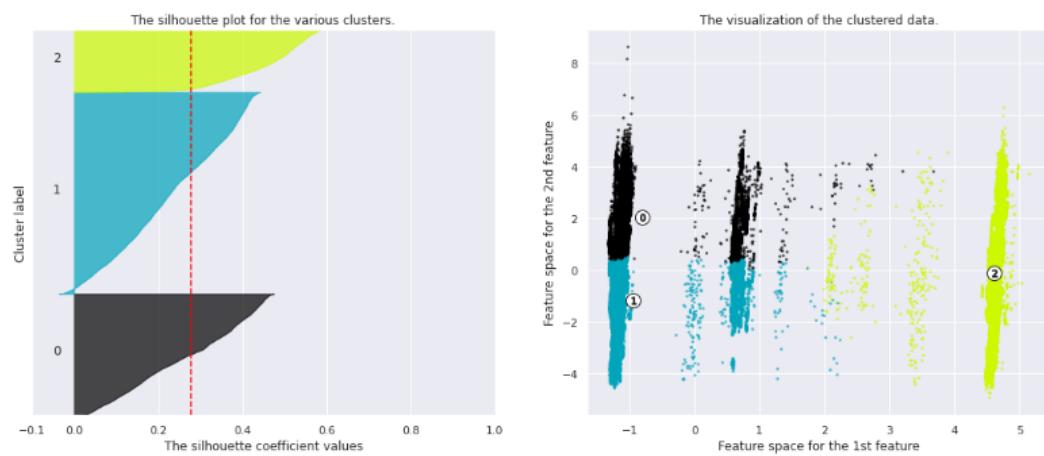


Figura 3.20. Aplicación de PCA + K-Means y análisis de silueta sobre la muestra.

Como se puede observar al comparar la Figura 3.19 con la Figura 3.20, se mejoran los resultados de agrupación. En los dos componentes la separación de los datos es mucho más visible y el coeficiente de silueta aumenta, por lo que esta transformación es efectiva.

3.4. Preprocesado

El preprocesado de datos es una de las tareas más importantes en el proceso de minería de datos. Los datos, normalmente, contienen ruido, valores nulos, errores, etc. Procesar los datos y mejorar la calidad de éstos permitirá que los modelos tengan mejor rendimiento y se puedan obtener mejores resultados.

Hay que tener en cuenta la naturaleza de los datos a procesar y también el objetivo del análisis a realizar con esos datos. En el contexto de este trabajo, los datos provienen de sensores de una célula (ver apartado 2.1) que podría estar parada o tener problemas de comunicación, situación en la cual se podrían producir datos nulos.

Tratar los datos nulos en series de datos temporales es una tarea que hay que realizar debido a que algunos modelos lo requieren.

Aunque la muestra extraída en el análisis no indique valores nulos, es muy importante incluir este paso en el preprocesado porque es esperable la posible pérdida de los sensores o la aparición de errores de lectura.

El primer paso es identificar aquellos sensores donde la mayoría de los valores son cero o nulo y eliminarlos, puesto que no aportan nada. En este caso, se han eliminado completamente aquellas columnas que lleguen con todos los valores a 0 o que el 70% de los valores sean nulos. En caso que haya alguna columna que tenga hasta el 30% de los valores nulos, se interpolan linealmente los valores uniendo los puntos mediante una recta como se representa en la Figura 3.21.

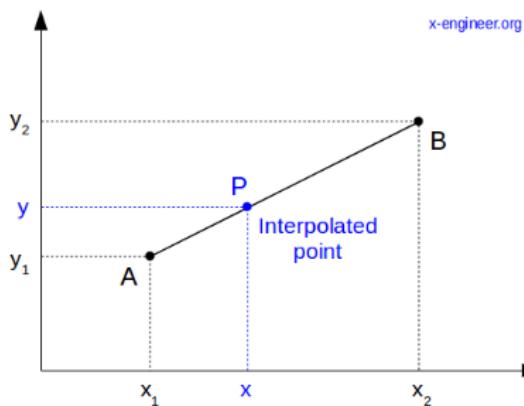


Figura 3.21. Representación de Interpolación lineal de dos puntos (fuente [21]).

La reducción de ruido depende del objetivo del análisis que se quiera realizar. En este caso, como uno de los objetivos es identificar anomalías, eliminar el ruido no es buena idea debido a que la finalidad de la tarea es encontrar las inconsistencias de comportamiento.

Otro paso común dentro del preprocesado es la Ingeniería de características (*feature engineering*) [22], que suele ser la parte más interesante. En este proyecto se han escogido las variables críticas definidas por ingeniería. No obstante, sería interesante encontrar qué sensores explican mejor el problema. Se trata de una tarea que requiere entender muy bien el problema y tener dominio sobre el proceso. Seleccionar las características adecuadas puede simplificar mucho los modelos y tener mejores resultados.

3.5. Modelado

Una vez hecho el estudio exploratorio de los datos y el preprocesado se pueden empezar a realizar modelos que intenten responder a las hipótesis planteadas.

Existen diferentes enfoques para efectuar el modelaje. Dentro del aprendizaje automático, los dos enfoques clásicos son el aprendizaje supervisado y el no supervisado, aunque hay más formas como el aprendizaje semisupervisado y el aprendizaje por refuerzo.

- **Aprendizaje supervisado:** Este enfoque, normalmente, tiene los datos etiquetados (se conoce la entrada y la salida de cada ejemplo u observación)

con los que se entrena un modelo. El modelo, aprendido con un algoritmo que ha identificado patrones en los datos, se utiliza para predecir la etiqueta en una nueva observación.

- **Aprendizaje no supervisado:** Este enfoque no tiene los datos etiquetados, por lo que se basa solamente en ellos para efectuar predicciones. Las técnicas más habituales son las regresiones y el *clustering*.

Ambos enfoques, generalmente, requieren una gran cantidad de datos para garantizar la coherencia y la confiabilidad. Un amplio conjunto de datos garantiza disponer de un tamaño de muestra representativo y donde se pueda disminuir el ruido. En el caso específico de los datos de naturaleza temporal, también garantiza que cualquier tendencia o patrón descubierto no sea un valor atípico y pueda explicar la variación estacional. Además, los datos de series temporales se pueden usar para efectuar pronósticos, es decir, predecir datos futuros en función de datos históricos (**forecasting**).

Otros modelos predictivos que se pueden integrar a este tipo de análisis son:

- **Clasificación:** Asignar etiquetas (clases) a los datos.
- **Curve fitting:** Encontrar las relaciones de las variables con los datos mediante curvas.
- **Descriptive analysis:** Encontrar patrones, tendencias, ciclos, etc.
- **Explanative analysis:** Entender los datos y la relación entre las variables.
- **Exploratory analysis:** Describir las características principales de los datos
- **Intervention analysis:** Estudio de eventos y cómo afectan a los datos.
- **Segmentation:** Se dividen los datos para estudiar períodos y extraer información.

Para la evaluación de los modelos, dependiendo de los datos que se tienen, es relevante contar con diferentes métricas. Aquí entran en juego muchos factores, puesto que, dependiendo de lo que se quiere encontrar, se pueden considerar unas métricas u otras. Por ejemplo, la *accuracy* no es una buena métrica para conjuntos de datos que se encuentran desbalanceados.

En este proyecto se crean dos modelos no supervisados para la detección de anomalías (clasificación) y predicción a futuro (regresión).

3.5.1. Predicción temporal (Regresión)

La aplicación principal para implementar el mantenimiento predictivo es la predicción temporal del comportamiento de los sensores. Predecir el comportamiento de los sensores puede ayudar a evitar prevenir fallos o comportamientos no deseados.

La predicción a futuro de series de datos temporales es un campo amplio de estudio que se aplica a muchos sectores como en el mundo financiero, meteorológico, etc. Poder obtener predicciones en el futuro puede ser muy beneficioso para ser capaz de anticiparse a los movimientos y también entender los diferentes patrones y características de comportamiento.

El primer paso es definir un objetivo concreto. En este proyecto, el objetivo es poder predecir el comportamiento de todos los sensores con una ventana de tiempo a futuro. Se trata de una tarea compleja debido a que concretamente se quiere obtener un modelo que sea capaz de predecir varios pasos en el futuro de todas las variables a la vez.

Existen varios modelos de predicción temporal a considerar dependiendo de lo que se quiera lograr. Algunos modelos tienen requisitos en el preprocesado y hay otros que son más flexibles. Algunos de los modelos más utilizados para predicción de series temporales son:

- **Moving Average, Smooth-based:** Estos modelos utilizan técnicas estadísticas de agregación para encontrar las tendencias en el futuro.
- **ARIMA / SARIMA / VAR [14]:** Técnicas de predicción mediante datos históricos a través de combinaciones lineales y análisis de autocorrelaciones. Estos modelos requieren que el muestreo sea de un periodo constante y que los datos sean estacionarios.
- **RNN (LSTM o Encoders):** Los métodos de aprendizaje profundo (*deep learning*) están ganando popularidad en tareas de predicción y clasificación debido a que son muy flexibles al adaptarse a los datos y entender los patrones.
- **Métodos clásicos de Machine Learning (distance based, tree based, SVM, etc.):** Es importante considerar los métodos clásicos debido a que suelen ser efectivos y eficientes. Dependiendo de los datos, modelos más simples pueden ser significativamente más viables.
- **Ensambles:** Combinar diferentes modelos para obtener un resultado más contrastado y robusto es un factor a considerar.

Con los datos preprocesados, dependiendo del modelo, se deberán dividir los datos en diferentes conjuntos. Normalmente, estos conjuntos son el de entrenamiento, el de validación y el de test. El conjunto de entrenamiento sirve para crear el modelo, mientras que el de validación sirve para encontrar los mejores parámetros. Finalmente, el conjunto de test es el que se usa para evaluar el modelo; aplicando el modelo sobre datos nuevos, se obtienen sus predicciones y se comparan con los valores reales/esperados.

Otro factor a tener en cuenta es si se quiere predecir la próxima medición o las próximas mediciones. Hay modelos que solo son capaces de predecir la próxima medición y hay algunos que son capaces de predecir más de una a la vez. El número de mediciones en el horizonte hace que el error crezca muy rápido.

En este proyecto se han estudiado varios modelos de predicción temporal. Se ha tenido en cuenta rendimiento, escalabilidad y eficacia para determinar el mejor modelo. Los principales modelos probados son los vectores autorregresivos (ver [Anexo C](#)) y las redes neuronales. Los vectores autorregresivos requieren pasos de preprocesado y parten de la premisa de que toda variable influye a las demás, por lo que no es del todo cierto. Las redes neuronales, por otra parte, han sido el modelo con mejores resultados. Además, se trata de un modelo más flexible y escalable para el tipo de problema que se quiere solucionar.

Redes Neuronales (*Deep Learning*)

Las redes neuronales han supuesto uno de los mayores avances tecnológicos en los últimos años. Se trata de modelos que son capaces de adaptarse a prácticamente cualquier problema gracias a su algoritmo de reducción de error (*backpropagation* [23]) en el entrenamiento.

Este modelo puede ser descrito como un conjunto de neuronas artificiales que reciben datos de entrada y devuelven valores formados por combinaciones lineales con pesos.

La arquitectura de estos modelos puede incluir capas intermedias que proporcionan la capacidad de dar respuesta a problemas no lineales.

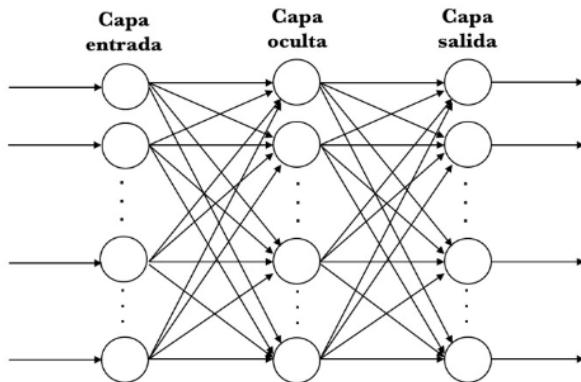


Figura 3.22. Arquitectura genérica de una red neuronal *feed-forward* (fuente: [3]).

Como se puede visualizar en el esquema de la Figura 3.22, se trata de un modelo básico con solo tres capas. La capa de entrada (*Input layer*) recibe unos valores y se pasan directamente a la capa escondida (*Hidden layer*) donde se realizan unos cálculos, que combinan las entradas con sus pesos (valores desconocidos), y se envían a la capa de salida (*Output layer*). En esta última capa, se vuelve a realizar otro cálculo para generar la salida (*Output*). Una vez obtenido el resultado, la propia red neuronal autorregula los pesos para disminuir el error mediante el algoritmo de descenso del gradiente (*back-propagation*). Se trata de un modelo *feed-forward* debido a que los datos recibidos en una capa son enviados a la siguiente capa.

Hay varias clases y arquitecturas de redes neuronales. En este proyecto se han escogido las **redes neuronales recurrentes (RRN)** que implementan una memoria interna para recordar información pasada gracias a que contienen neuronas que implementan conexiones cíclicas. Es decir que se utilizan predicciones antiguas en el input. Esta naturaleza la hace una arquitectura ideal para la tarea que se quiere realizar, que es predecir el funcionamiento de la máquina en base a observaciones antiguas.

Dentro de las redes neuronales recurrentes hay una variante especial muy popular llamada **LSTM** (*Long Short Term Memory*). Este tipo de redes se caracteriza por tener

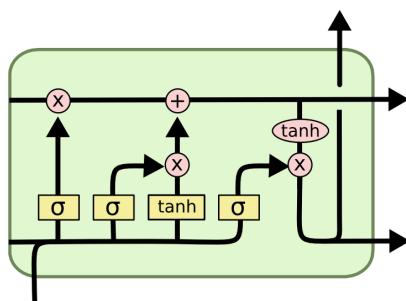


Figura 3.23. Composición de una neurona LSTM (fuente: [25]).

mecanismos que simulan una memoria y son capaces de recordar muchos pasos en el pasado.

En la Figura 3.23 se puede observar como a la neurona se le añaden un conjunto de puertas que modifican la información de entrada. Las funciones de estas puertas tienen como objetivo saber qué información recordar y cuál olvidar. Esto implica que cada neurona es capaz de añadir un nuevo estado o mantener el que tiene.

Juntando varias capas de conjuntos de neuronas de este tipo, se puede obtener un modelo poderoso capaz de reconstruir secuencias teniendo en cuenta información pasada. El modelaje es un reto debido a que hay que tener en cuenta muchos factores. La cantidad y densidad de las capas puede provocar *overfitting* y aumentar el tiempo de inferencia y entrenamiento.

También es importante tener en cuenta que el modelo tiene que ser capaz de predecir el comportamiento de diferentes sensores a la vez y con varios pasos al futuro. Eso hace que la tarea sea muy compleja y que sea muy difícil obtener buenos resultados.

Para la implementación de la red neuronal, se ha utilizado el módulo *Keras*, una API de alto nivel que forma parte de la biblioteca *TensorFlow* (Google). El módulo *Keras* contiene las clases necesarias para crear diferentes modelos empleando diferentes arquitecturas.

Para implementar la predicción al futuro se dividen los datos en entrenamiento, validación y test, se especifica el número de pasos que se quiere ver al futuro y se crea la arquitectura teniendo en cuenta estos datos. La combinación de estos parámetros afecta al tiempo de entrenamiento, al rendimiento del modelo y a los resultados. Cuanto más se incrementa el número de pasos en el futuro, peor son los resultados en todos los aspectos.

Teniendo en cuenta estas condiciones, es importante recalcar que no hay unas reglas fijas para obtener buenos resultados modelando. Esto hace que el modelado sea una tarea que requiere intuición, experiencia y mucho prueba y error. El modelaje, en este caso, ha consistido en encontrar modelos base de partida e, iterativamente, ir añadiendo complejidad al modelo y evaluando, hasta encontrar un modelo que mantenga una buena relación entre resultados y cumpla las condiciones establecidas.

El modelo está formado por dos capas con neuronas LSTM. Este tipo de arquitectura se conoce como *Stacked LSTM* y es capaz de recordar muchos pasos en el pasado.

La primera capa LSTM está formada por 256 neuronas y además es una capa bidireccional por lo que la secuencia de input es leída al revés también. Después de la primera capa se añade una capa *Dropout* para evitar *overfitting*. La segunda capa LSTM es normal y contiene 128 neuronas. La siguiente capa es una capa *Dense* (neuronas normales) donde el número de neuronas es igual a la multiplicación del número de variables por

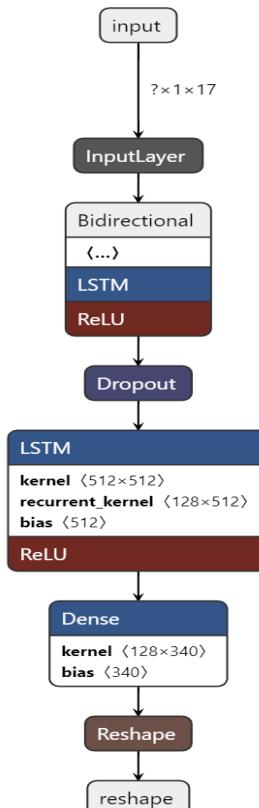


Figura 3.24. Arquitectura del modelo final (generada mediante la herramienta netron.app [27])

los pasos al futuro que se quieren predecir. Eso hace que haya muchos valores de salida que finalmente en la capa *Reshape* se transforman para obtener los vectores de valores de predicción. En conjunto, este tipo de arquitectura tiene el nombre de *Vector Output*.

En la Figura 3.24 se puede observar que se ha entrenado de manera que recibe 1 paso en el tiempo con 17 variables y es capaz de predecir **20 pasos en el futuro** de estas.

No ha sido posible ver más al futuro debido a problemas con el tiempo de inferencia. No obstante, con mejores capacidades de proceso se podría intentar mirar más a futuro.

3.5.2. Detección de anomalías (Clasificación)

Otra de las aplicaciones que se ha implementado en este proyecto es la detección de anomalías o comportamientos inconsistentes de la célula de inyección.

Los modelos que se han estudiado han sido también modelos no supervisados debido a que no se dispone de una muestra de datos donde se indiquen anomalías (los datos no están etiquetados). Este escenario hace que también sea complicado evaluar el rendimiento de los modelos.

Una pregunta que hay que responder al estudiar diferentes modelos es si se quiere encontrar anomalías de forma multivariada o univariada. En este caso, si se aplicara un enfoque univariado, se trataría cada sensor individualmente y se obtendrían las anomalías de cada uno por separado. Por otro lado, el enfoque multivariado supondría tratar todos los sensores a la vez, verificando si la combinación de estos es inconsistente en relación con lo observado. En la siguiente imagen se puede observar un ejemplo de detección multivariada.

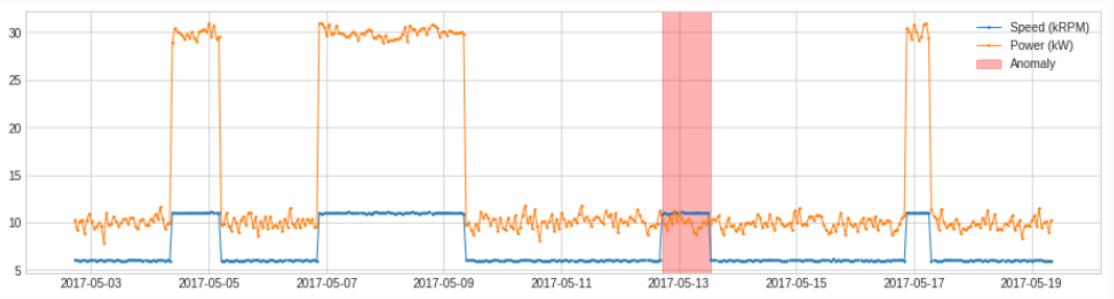


Figura 3.25. Ejemplo de detección multivariada de anomalías (fuente: [28]).

En este proyecto se ha modelado teniendo en cuenta, principalmente, el estudio multivariado realizado previamente. Se han estudiado diferentes modelos para la detección no supervisada de anomalías, entre los que se incluyen *clustering* (*K-Means*), *PCA*, *Isolation Forest* [29], entre otros. Finalmente, el modelo escogido se basa en una combinación de *PCA* y *clustering*.

El algoritmo de clasificación no supervisada seleccionado para la tarea ha sido *K-means* por su relación resultados-velocidad. El proceso que ejecuta el algoritmo es tratar todos los sensores como puntos independientes en el espacio y después agrupa

los diferentes puntos en clústeres. Finalmente, se seleccionan los puntos del clúster más pequeño como anomalías. La Figura 3.26 muestra un ejemplo de este proceso.

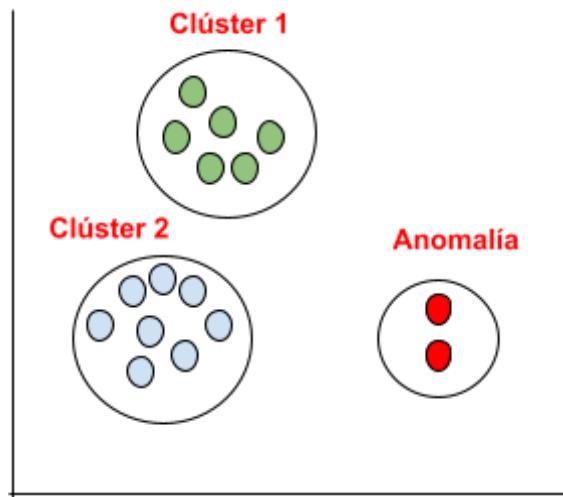


Figura 3.26. Ejemplo visual de cómo se detectan las anomalías

Para la implementación del modelo se ha utilizado el módulo adtk [28] de Python que proporciona un conjunto de herramientas y modelos para la detección de anomalías. Principalmente, se ha utilizado la herramienta *Pipenet* que permite combinar transformadores, modelos y agregados para formar un modelo más complejo. En la imagen siguiente se puede visualizar el *pipenet* creado:

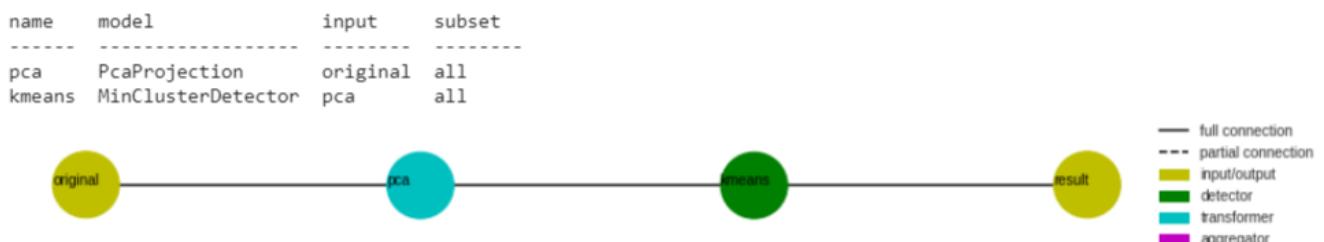


Figura 3.27. Diagrama generado de la *pipenet* creada que combina PCA + KMeans.

En la Figura 3.27, se puede ver que los datos pasan por dos procesos. Primero pasan por PCA, para transformar los datos y reducir su dimensionalidad, y el resultado de éste pasa a K-Means, que se encarga de hacer el *clustering* y marcar los valores del clúster más pequeño como anomalías. El paso de PCA no solo permite separar mejor los datos sino que también optimiza el rendimiento de K-Means.

3.6. Evaluación

En este apartado se describe la metodología empleada para evaluar y mejorar los modelos no supervisados descritos en los apartados anteriores. La evaluación ha sido una tarea que se ha realizado iterativamente durante el modelado. En el siguiente apartado se describirán algunas de las metodologías y decisiones tomadas.

Al ser modelos no supervisados, la tarea de evaluación es más ambigua que al tratar con modelos supervisados. Esto es debido a que se suele tener en cuenta el comportamiento teórico de los modelos para evaluarlos.

En el modelo de predicción al futuro, para encontrar los mejores parámetros y un buen balance en la arquitectura se han tenido en cuenta tanto los resultados en el subconjunto de validación como los resultados en el subconjunto de test. Se ha empezado con un modelo sencillo, y se ha ido añadiendo iterativamente complejidad al modelo y analizando cómo evolucionan los resultados. La evolución de los resultados del modelo final se pueden ir observando gráficas. En el ejemplo de la Figura 3.28, se muestran las métricas del modelo entrenado con 48h de datos, 20 epochs (pasadas completas sobre los datos) y una proporción de datos de 15% para validación.

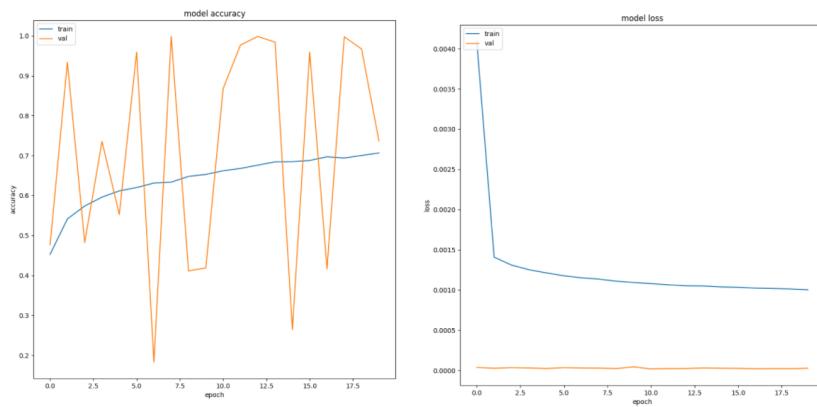


Figura 3.28. Métricas del modelo en el conjunto de entrenamiento y validación, con una muestra de 48h.

En las gráficas de la Figura 3.28, podemos observar que la *accuracy* del modelo durante el entrenamiento empieza a converger sobre la quinceava pasada en un 65% aproximadamente. El resultado en el conjunto de validación es muy inestable, pero se consiguen picos de 80%.

El método que mejor ha permitido visualizar los resultados ha sido utilizar *walk validation* conjuntamente con la métrica de error cuadrático medio (RSME) [30]. En la fórmula de la Figura 3.29 se puede observar que la métrica consiste en la raíz cuadrada del sumatorio de la diferencia entre los valores predichos y observados al cuadrado. Concretamente, se ha recorrido el conjunto de test y se ha ido prediciendo 20 pasos al futuro punto a punto calculando la métrica en el proceso.

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^n (P_i - O_i)^2}{n}}$$

Figura 3.29. Fórmula del error cuadrático medio (fuente: [30]).

En el resultado de *walk validation*, se obtiene el valor medio de RSME y también se grafican los resultados para observar los valores predichos y los reales de cada sensor. Los resultados en el subconjunto de test es el que se muestra en la Figura 3.30.

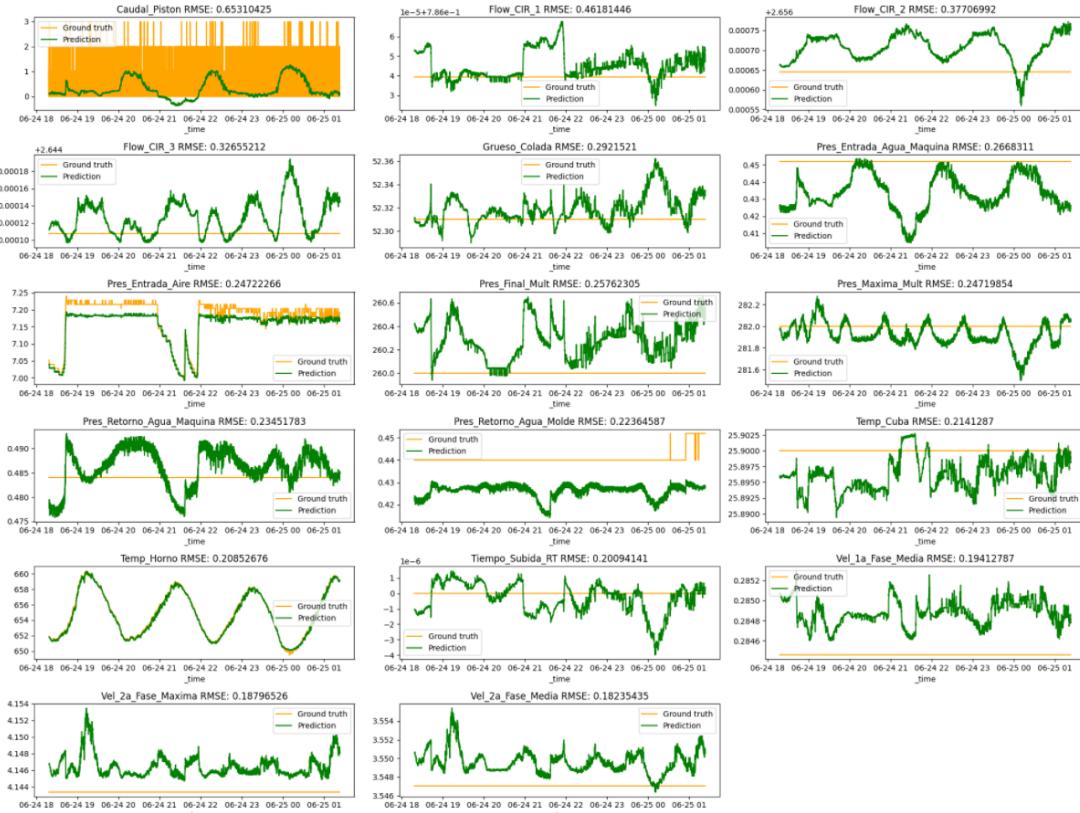


Figura 3.30. Gráficos generados por el método *walk validation* sobre el conjunto de test.

En cada gráfica de la Figura 3.30, se pueden observar los valores predichos (verde) y los observados (amarillo). Los resultados, en general, son buenos. No obstante, se puede ver que el comportamiento predicho de los sensores no coincide en muchos casos. Aunque son los mejores resultados obtenidos, todavía hay mucho camino de mejora a recorrer.

En cuanto al modelo de detección de anomalías, la selección de parámetros se ha realizado mediante el estudio multivariado explicado anteriormente. La evaluación de los resultados se ha hecho graficando las anomalías detectadas en el conjunto de datos. La Figura 3.31 muestra un ejemplo.

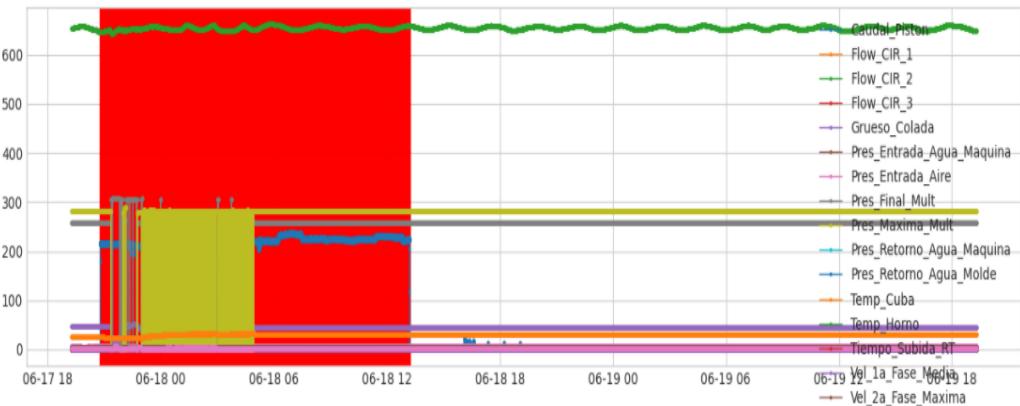


Figura 3.31. Resultado de detección de anomalía multivariada sobre una muestra de 24h.

Como se puede observar en la Figura [3.31](#) generada, se detecta el período de variabilidad de la Pres_Retorno_Agua_Molde como anomalía. Probablemente, si no hubiera sido la presión del molde, la variabilidad de las velocidades también hubieran sido marcadas como anomalía. En términos generales se puede decir que el modelo funciona correctamente.

4. DESARROLLO Y DESPLIEGUE APPLICACIÓN

Con los modelos creados y evaluados, el siguiente paso es crear la aplicación que se encargará de recoger los datos, tratarlos y finalmente, poder visualizar los resultados.

Para este tipo de aplicaciones es importante estudiar y escoger bien las herramientas que se van a utilizar. Dependiendo del objetivo, algunas herramientas pueden funcionar mejor que otras. Es fundamental tener en cuenta factores como la accesibilidad, escalabilidad, comunicación y el rendimiento.

En este trabajo, el resultado final será una aplicación web donde se podrán visualizar resultados fruto de todos los procesos anteriores. Estará destinada a los ingenieros de proceso para poder realizar tareas de análisis y monitorización de datos.

La aplicación está centrada únicamente en la Célula 86, aunque se han tenido en cuenta factores de escalabilidad a la hora de hacer el diseño y desarrollo de la aplicación.

4.1. Herramientas

Hacer un estudio de las herramientas a utilizar es muy importante debido a que se puede perder mucho tiempo y recursos con herramientas inadecuadas para el tipo de proyecto que uno quiere hacer.

En este trabajo, para escoger las herramientas de software necesarias he tenido en cuenta los siguientes factores: accesibilidad, familiaridad y adecuación. A continuación, se detallan las principales herramientas.

- **Node-RED (v2.2.2) [31]**

Se trata de un framework basado en Node.JS [32] que se basa en programación de flujos de mensajes y nodos. Con esta herramienta es posible integrar hardware, APIs y funcionalidades de una manera muy accesible.

Node-RED permite conectar gráficamente bloques predefinidos, llamados nodos, para desarrollar una tarea concreta.

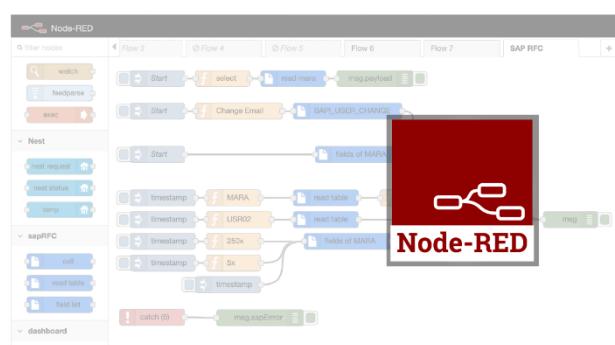


Figura 4.1. Logo y vista de Node-RED (fuente: [33]).

La conexión de los nodos, habitualmente una combinación de nodos de entrada, nodos de procesamiento y nodos de salida, forman lo que conocemos como *flow* (*flujos*).

Se trata de una herramienta de código abierto muy configurable y personalizable. Su versatilidad la está convirtiendo en una de las opciones principales para soluciones IoT e Industria 4.0.

Para el apartado del *frontend*, Node-RED cuenta con un módulo instalable muy interesante llamado ui-builder [34]. Este módulo permite configurar una interfaz web para la visualización de datos. ui-builder permite configurar diferentes tecnologías web. En este trabajo se creará una aplicación web mediante VueJS (v2) + Bootstrap 4 [35].

- **Python (v3.9.7)** [36]

Para el tratamiento de los datos, Python será la herramienta principal. Se trata de uno de los lenguajes de programación más utilizados actualmente debido a su gran versatilidad, usabilidad y gran comunidad.

Es un lenguaje de código abierto con una gran cantidad de módulos para todo tipo de aplicaciones. Es la herramienta preferida para tareas de ciencia de datos e inteligencia artificial.



Figura 4.2. Logo Python (fuente: [36]).

Los módulos de Python que se usan en este proyecto son principalmente pandas, numpy, scikit-learn, keras y otras herramientas de visualización de datos (ver Anexo D).

- **InfluxDB (v2.0.0)** [37]

Base de datos de series temporales (*Time Series Database*, TSDB). Este tipo de bases de datos están optimizadas para datos temporales.



Figura 4.3. Logo InfluxDB (fuente: [37]).

InfluxDB tiene el concepto de “cubos” que permite almacenar grandes cantidades de datos de forma eficiente, teniendo la posibilidad de especificar un tiempo de retención para cada cubo. El tiempo de retención indica que cada medición, pasada ese tiempo, es eliminada.

Por otra parte, la propia base de datos permite la configuración de *dashboards*, alertas y otros sistemas de monitorización de la información. También es muy versátil en

cuanto a la integración con otras fuentes de datos. Estas características la hacen una base de datos ideal para este proyecto.

- **Docker (v18.09.1) [38]**

Se trata de una aplicación que permite el despliegue de aplicaciones dentro de contenedores, proporcionando una capa de virtualización, permitiendo que las aplicaciones se puedan aislar y automatizar.

Al igual que las herramientas anteriores, se trata de un proyecto de código abierto que se basa en el uso de espacios de nombres (*namespaces*) para virtualizar pequeñas aplicaciones.

En este proyecto se creará un contenedor para lo que es Node-RED y otro contenedor para InfluxDB.

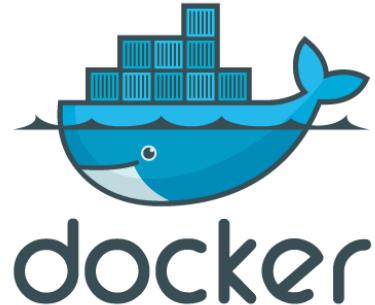


Figura 4.4. Logo Docker (fuente:[38]).

4.2. Arquitectura

Para el desarrollo de la aplicación, se ha habilitado una máquina virtual con Debian 9 recién instalado como sistema operativo. En cuanto al hardware proporcionado en la máquina virtual, tenemos la configuración que se muestra en la Figura 4.5.

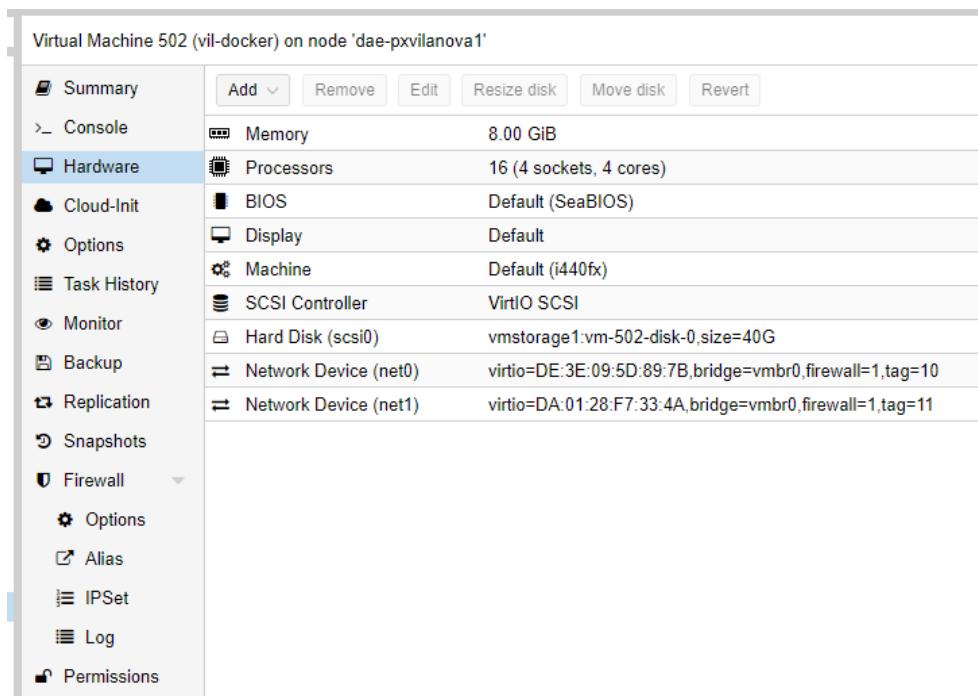


Figura 4.5. Hardware proporcionado para el proyecto.

La máquina virtual cuenta con muchos recursos de procesador debido a que los entrenamientos son muy demandantes. En cuanto a la RAM, con 8GB es suficiente.

También es importante destacar que la máquina virtual tiene 2 interfaces de red para poder acceder también a la red de autómatas (PLCs) y a la red de ordenadores (PCs).

En la máquina virtual, se creará la arquitectura de la aplicación teniendo en cuenta las herramientas mencionadas anteriormente. Una de las principales tareas para desarrollar la aplicación es asegurar que todos los componentes funcionan independientemente y que es posible la comunicación entre ellos. Docker es la herramienta adecuada para esta premisa.

Docker facilita el proceso de automatizar la creación y la inicialización de los diferentes contenedores mediante la herramienta docker-compose. Esta herramienta permite producir aplicaciones que están formadas por otras, las cuales se llaman “servicios”. Cada servicio es configurable y tiene su propio contenedor. Toda la configuración de los diferentes servicios se encuentra en un archivo con extensión `yml` (ver [ANEXO E](#)).

Una de las ventajas más importantes de Docker es que permite la persistencia de datos mediante el concepto de volúmenes. Es decir, al inicializar los servicios, se monta la información dentro de los contenedores que la forman.

Teniendo en cuenta estos conceptos, la arquitectura está montada principalmente por dos servicios: Node-RED para la parte lógica de la aplicación y para el almacenamiento, y InfluxDB que se encargará de almacenar los datos. En la Figura [4.6](#) se puede visualizar la arquitectura de la aplicación.

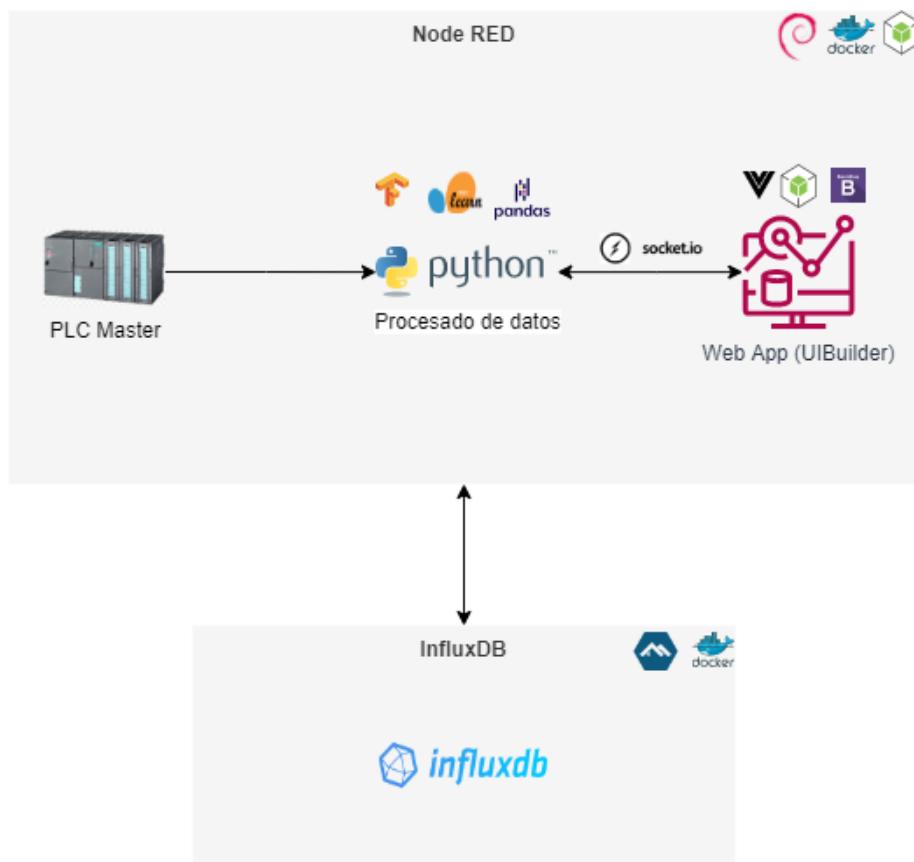


Figura 4.6. Diagrama de la arquitectura de servicios y componentes de la aplicación final.

Como se puede observar en el diagrama de la Figura 4.6, hay dos contenedores (servicios) que contendrán las diferentes funcionalidades de la aplicación. Cada contenedor tiene su propio sistema operativo. En el caso del contenedor de InfluxDB se cuenta con Linux Alpine y en el caso de Node-RED se trata de un Debian.

El contenedor de InfluxDB se ejecuta en el puerto 8086. Si se inicializan los servicios por primera vez, en el archivo `docker-compose.yml` (ver [Anexo E](#)) se configura un servicio adicional que crea el *setup* inicial de la base de datos especificando usuario, contraseña, *token*, organización y un cubo inicial. De esta manera, no hay que pasar por el asistente de inicialización de la base de datos y, además, se asegura que la comunicación desde otros servicios pueda ser especificada.

Por otra parte, el contenedor de Node-RED se ejecuta en el puerto 1880 y contiene toda la lógica e integra todos los elementos de la aplicación. Como se puede ver en el mismo diagrama, desde Node-RED se lee la información del PLC, se realizan todos los procesos de minería de datos, se guardan los resultados en la base de datos y también se comunica con la aplicación web que permite visualizar los datos.

Es importante destacar que la comunicación entre Node-RED y la aplicación web se realiza mediante *sockets* web ([Socket.IO \[41\]](#)). Esto permite visualizar los datos en tiempo real sin tener que pasar por la base de datos.

Para poder integrar las diferentes herramientas se han instalado los siguientes módulos principales en Node-RED:

- **node-red-contrib-s7 (v3.1.0)** [\[42\]](#): Comunicación PLCs de la marca Siemens S7.
- **node-red-contrib-influxdb (v0.6.1)** [\[43\]](#): Comunicación con instancias de InfluxDB.
- **node-red-contrib-uibuilder (v5.0.2)** [\[34\]](#): Creación de una instancia web y comunicación con ella mediante *sockets*.
- **node-red-contrib-machine-learning (v1.0.8)** [\[44\]](#): Kit con nodos para aprendizaje automático. En este proyecto se utiliza solamente el nodo que permite cargar modelos entrenados y devolver predicciones).

Para poder integrar la lectura e inferencia de los modelos, se ha editado el código fuente del módulo de machine learning [\[44\]](#) añadiendo algunos ajustes en el nodo de predicción y creando un nodo nuevo con un comportamiento similar pero adaptándolo para que funcione con modelos de la biblioteca Keras de Python.

Una de las grandes ventajas de Node-RED es que se basa en Node JS. Esto le proporciona mucha flexibilidad y una gran comunidad. Hay muchos otros complementos instalables que podrían darle muchas más funcionalidades a la aplicación y también está la posibilidad de generar nodos propios.

4.3. Desarrollo

El desarrollo de la aplicación se puede dividir en tres componentes principales: el pipeline de datos, el entrenamiento de modelos y el desarrollo de la aplicación web.

Una gran ventaja de Node-RED es que, al ser una herramienta gráfica basada en nodos, es muy fácil visualizar estas partes y cómo se comunican.

4.3.1. Pipeline de datos

El pipeline de datos es el flujo de datos que alimenta toda la aplicación, desde que se recogen los datos hasta que se visualizan en la aplicación final. En la Figura 4.7 se puede observar el flujo de datos principal en Node-RED.

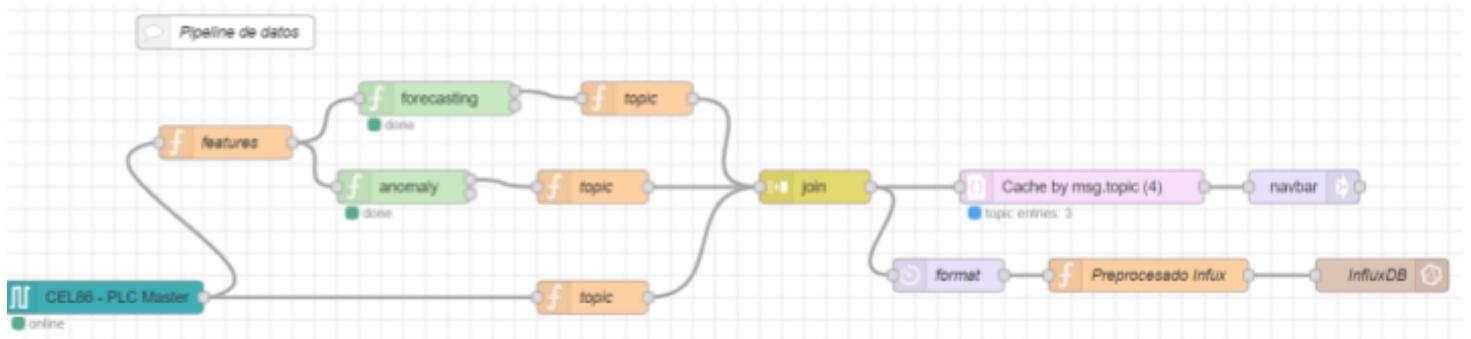


Figura 4.7. Flujo principal de Node-RED donde se procesan los datos.

Leyendo el flujo de la Figura 4.7 de izquierda a derecha, se puede observar que el primer nodo es el encargado de leer los datos del PLC. Este nodo está configurado para leer los datos con una frecuencia de muestreo de 1 segundo, leyendo los más de 100 valores especificados.

Después de leer los datos hay dos flujos de datos. El inferior simplemente contiene todos los datos leídos en ese instante de tiempo para poder guardarlos en la base de datos y mostrarlos en la web.

Por otra parte, en el flujo superior es donde está la inferencia de los modelos creados. En el primer nodo se seleccionan las características principales que se pasarán como entrada a los modelos entrenados; después, los modelos procesan la entrada y devuelven su predicción. El modelo de *forecasting* (predicción al futuro) recibe las características y devuelve un vector con todos los pasos predichos en el tiempo. Por otra parte, el modelo de detección de anomalías devuelve un valor binario etiquetando la medición como “anomalía” o “no anomalía”.

Los tres flujos de datos resultantes se anotan con una etiqueta y pasan al nodo que se encarga de juntarlos en uno solo. Se pueden acceder a los resultados de cada nodo posteriormente mediante la etiqueta anteriormente establecida.

Finalmente, con todos los resultados se ocasionan dos flujos finales de datos. El primero envía los resultados a la aplicación web mediante *sockets* y el segundo pasa por un preprocesado y se guarda en la base de datos InfluxDB. Esto permite tener datos en tiempo real para poder realizar monitorización y también datos históricos para poder realizar análisis y minería de datos.

4.3.2. Entrenamiento de modelos

Para tener disponibles los modelos, se ha creado un mecanismo para poder entrenarlos y que se carguen automáticamente en el *pipeline* de datos.

Se han programado dos scripts en Python donde cada uno se encarga de recoger los datos de la base de datos, preprocesarlos, entrenar los modelos y guardarlos. Al guardar los modelos se sobrescriben los anteriores, por lo que en el *pipeline* de datos se cargan los más recientes.

Node-RED dispone de un nodo que permite ejecutar comandos de sistema. Este nodo es el que se utiliza para ejecutar los scripts en Python y entrenar los modelos. También es posible especificar que la ejecución sea periódica en el tiempo.

En la Figura 4.8 se puede ver el proceso descrito. Como se puede observar, en ambos casos la estructura es la misma. Un flujo que permite ejecutar los scripts y otro para poder visualizar los resultados del entrenamiento. La visualización de los resultados es gracias al complemento node-red-contrib-image-tools (v2.0.4) [45].



Figura 4.8. Flujo de entrenamiento de modelos y visualización de resultados.

4.3.2. Aplicación web

El último componente de la aplicación es la interfaz web en la cual se pueden visualizar los resultados de todo el proceso de recogida y predicción de datos.

La aplicación web ha sido desarrollada mediante el complemento **UIBuilder** [34] de Node-RED. Este componente permite crear instancias web en el mismo servidor de Node-RED y crear una conexión mediante *sockets* para la comunicación entre las dos. Gracias a la comunicación mediante *sockets* es posible visualizar datos en tiempo real sin tener que pasar por la base de datos. No obstante, también es posible comunicar con base de datos teniendo a Node-RED como intermediario.



Figura 4.9. Flujo de la instancia web (UIBuilder) y posibilidad de consultas a InfluxDB.

En el flujo de datos de la Figura 4.9, se puede observar cómo se recibe desde la aplicación web un mensaje con la consulta y se ejecuta en el nodo proporcionado por InfluxDB. El resultado de la consulta es transformado a formato JSON y enviado de vuelta a la web.

En cuanto a las tecnologías de la aplicación web, el propio complemento **UIBuilder** da la flexibilidad de escoger la tecnología. En este proyecto, se ha escogido el framework **Vue.JS** para la programación y el componente **VueBootstrap** (**Bootstrap 4.0**) [46] que se basa en el diseño estético. También se han empleado otras bibliotecas para la interfaz y la visualización. Las bibliotecas y la configuración se pueden ver en el archivo **package.JSON** en el [ANEXO D](#).

Vue JS es un framework de desarrollo web progresivo. Esto quiere decir que es posible crear aplicaciones sencillas y escalar a aplicaciones muy complejas. Este framework es declarativo y se basa en programación basada en componentes.

Los componentes son elementos de la interfaz que se programan de manera aislada, conteniendo cada uno su propia parte visual y parte lógica. Los componentes después se pueden organizar y comunicar entre ellos para crear la interfaz.

El objetivo era tener un *dashboard* de visualización para poder tanto hacer monitorización como análisis histórico de las señales conjuntamente con los resultados de los modelos. El boceto inicial de la aplicación puede verse en la Figura 4.10.

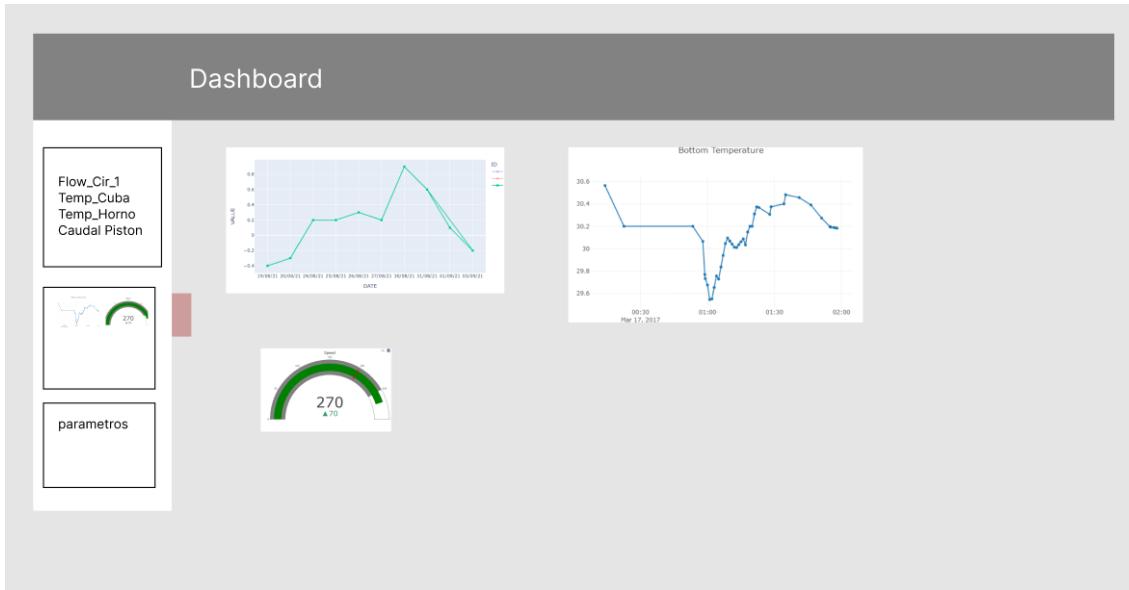


Figura 4.10. Boceto inicial del dashboard web ([47]).

El caso de uso básico sería escoger el sensor, el *widget* de visualización y los parámetros y poder ver en pantalla el resultado. Los *widgets* son diferentes formas de visualización de datos y los parámetros podrían ser, por ejemplo, la ventana de tiempo en la que se quiere ver un histórico.

4.4. Demo

En este apartado se podrá visualizar el resultado final de todos los pasos descritos en el apartado [4.3](#). Concretamente, se describe la aplicación web, que será un *dashboard* que permitirá visualizar los resultados del proceso, y sus funcionalidades.



Figura 4.11. Partes principales del dashboard web.

En la Figura 4.11 se puede observar el *dashboard* creado para la visualización de los resultados. La pantalla se divide en dos zonas principales: La cuadrícula de visualización y la barra lateral que permite añadir *widgets* a la cuadrícula. También se dispone de un botón que permite desplegar y esconder la barra lateral y otro que muestra un modal con información.

- **Barra lateral**

Inicialmente, la cuadrícula se encuentra vacía. Para añadir elementos a la cuadrícula hay que recurrir a la barra lateral haciendo clic en el botón que la despliega. En la barra lateral aparecen 3 opciones principales y un botón de Añadir que inicialmente está desactivado.

Para añadir un elemento, el requisito principal es escoger un Sensor. En la Figura 4.11 se puede observar que algunos sensores aparecen con una alerta roja. La alerta indica que el valor de la medición se ha encontrado fuera de los límites de tolerancia fijados en la máquina durante los últimos 30 minutos.

Una vez escogido un sensor (Figura 4.12) ya se deshabilita el botón de añadir y se puede agregar un elemento a la cuadrícula. Si no se especifica el *widget* y los parámetros, por defecto se muestra el *widget* PlotlyGrafica que permite ver datos históricos correspondientes a los últimos 30 minutos del sensor. En caso de querer especificar *widget*, es posible escoger uno de los disponibles. Cada *widget* permite visualizar al sensor de diferente forma. Hay *widgets* para visualizar el histórico y realizar análisis PlotlyGrafica y hay otros que son para ver datos en tiempo real y para monitorización (Medidor y ApexGrafica).

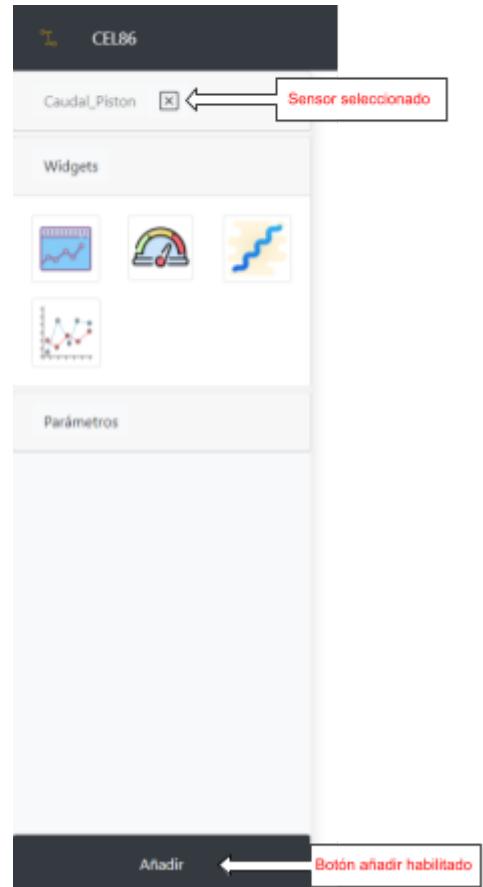


Figura 4.12. Opciones barra lateral.

En la cuadrícula se pueden añadir *widgets* diferentes para el mismo sensor. El objetivo es crear muchos *widgets* para poder enriquecer el análisis y monitorización del funcionamiento de la célula. Los *widgets* disponibles son:

- **PlotlyGrafica:** Este *widget* (Figura 4.13) está basado en la biblioteca Plotly y su propósito es la realización de análisis histórico agregando los resultados obtenidos de los modelos. En esta gráfica se pueden observar las predicciones al futuro y las anomalías predichas. También se pueden observar eventos como cuando la célula se encuentra parada o los momentos en que la máquina hace la inyección.

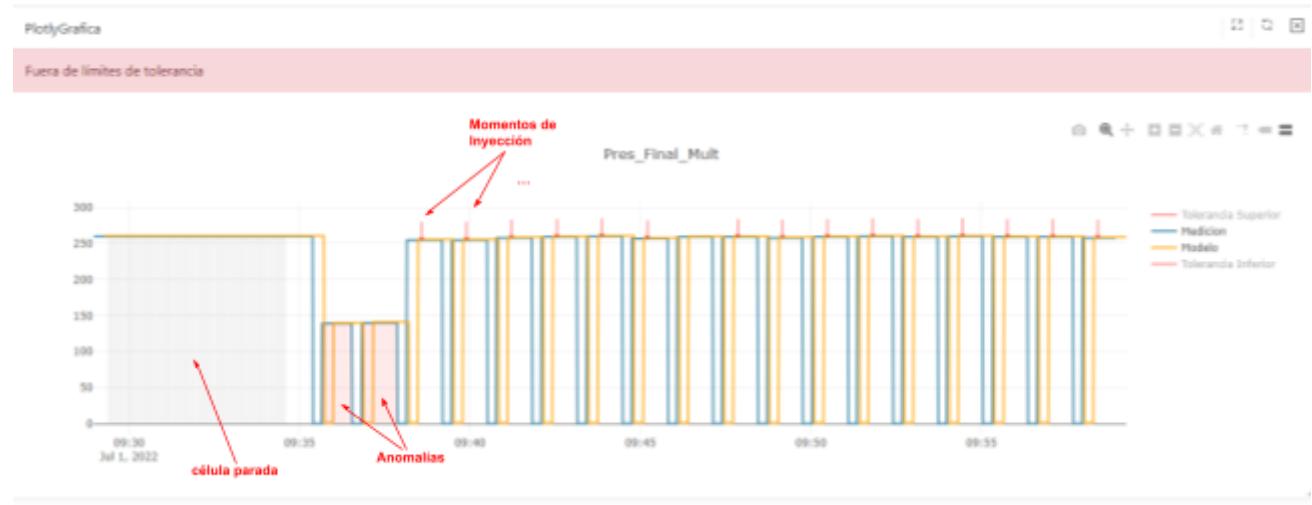


Figura 4.13. widget PlotlyGrafica y sus características.

- **Medidor:** Este widget (Figura 4.14) está pensado para la monitorización del valor en tiempo real de un sensor.

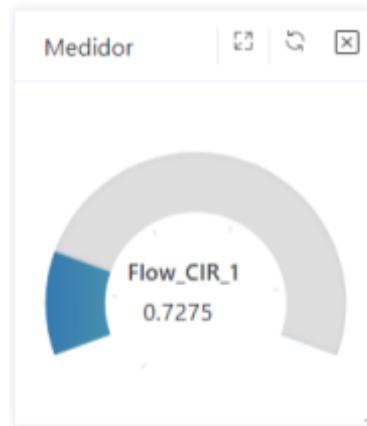


Figura 4.14. widget Medidor.

- **ApexGrafica:** Este widget (Figura 4.15) está también pensado para la monitorización en tiempo real y en este caso se muestra una ventana de tiempo de las últimas mediciones y se agregan las predicciones a futuro de la última medición. También se pueden visualizar los eventos como inyecciones o anomalías.

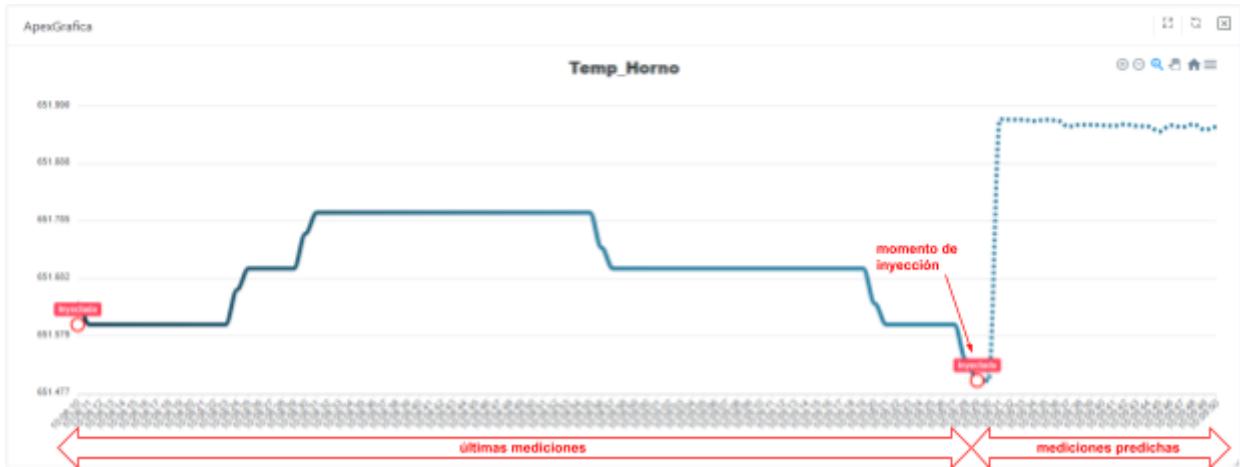


Figura 4.15. widget ApexGrafica y sus características

- **EchartsGrafica:** Este *widget* (Figura 4.16) está pensado para combinar los datos históricos con datos reales de manera que primero carga una ventana de tiempo específica en el pasado desde la base de datos y después se empiezan a añadir valores en tiempo real mediante *sockets*. Este *widget* ha quedado en desarrollo.



Figura 4.16. widget EchartsGrafica.

Algunos *widgets* tienen la posibilidad de especificación de parámetros desde la barra lateral. Por ejemplo, en el caso de PlotlyGrafica o EchartsGrafica, se puede especificar la ventana de tiempo de consulta.

Finalmente, en la Figura 4.12, se puede apreciar que cada vez que se selecciona un paso éste queda seleccionado, aunque también es posible deshacerlo y cambiar de opción.

• Cuadrícula de visualización

Una vez añadido el *widget* a la cuadrícula de visualización, hay algunas opciones que hacen que la cuadrícula sea flexible. Es posible cambiar el tamaño de los *widgets* a medida, cambiarlos de lugar o eliminarlos de manera que se pueda personalizar la visualización.

En la parte superior de cada *widget*, hay opciones que permiten hacer que el *widget* se muestre en pantalla completa y también actualizar el contenido de este de manera que se muestre el contenido como si se hubiera añadido por primera vez (ver Figura 4.17).

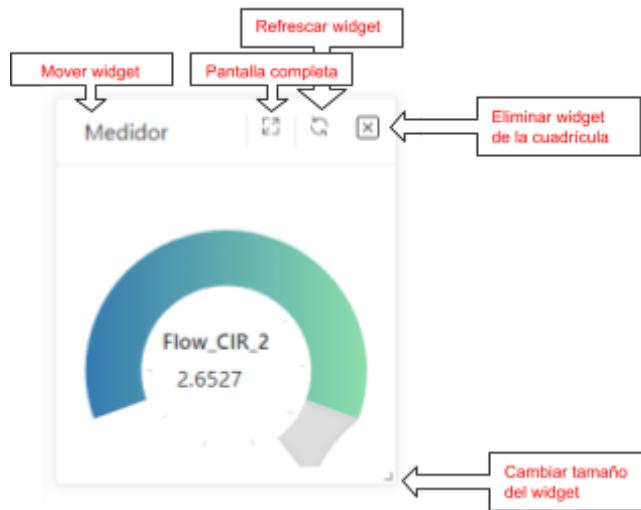


Figura 4.17. Opciones generales de cada *widget*.

Algunos *widgets* pueden tener sus propias opciones internas, como es el caso de PlotlyGrafica, que permiten habilitar y deshabilitar otras líneas de medición, realizar zoom o guardar imagen a modo de captura de la gráfica (ver Figura 4.18).

La idea general es disponer de una gran variedad de *widgets* con muchas opciones y parámetros para poder hacer un análisis lo más personalizado y extenso posible.

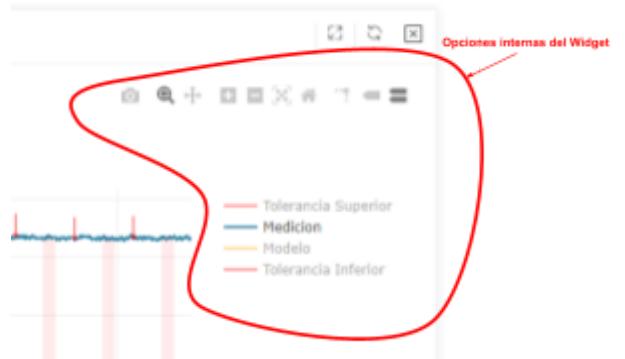


Figura 4.18. Opciones internas del widget *PlotlyGrafica*

También añadir que la cuadrícula de visualización ocupará el máximo espacio posible en caso de ocultar la barra lateral. Los *widgets* se adaptarán al tamaño de la nueva cuadrícula. En la Figura 4.19 se puede ver el resultado final.

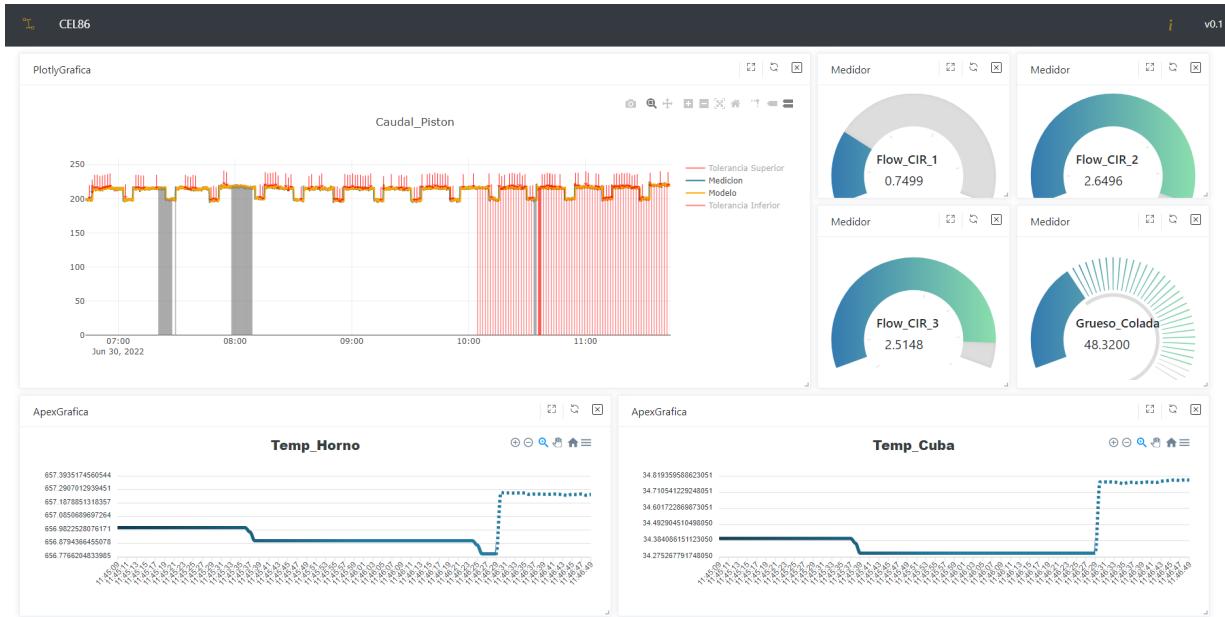


Figura 4.19. Dashboard con widgets y la barra lateral escondida.

Como se puede observar, la cuadrícula del *dashboard* ocupa todo el espacio de la pantalla y los *widgets* han sido dimensionados para crear una pantalla de análisis, donde con el *widget* PlotlyGrafica se está realizando un análisis histórico del sensor Caudal_Piston y, por otra parte, los otros *widgets* (Medidor y ApexGrafica) sirven para realizar una monitorización en tiempo real de los sensores. Finalmente, la aplicación también dispone de un botón que abre un modal donde se incluye la información de la aplicación.

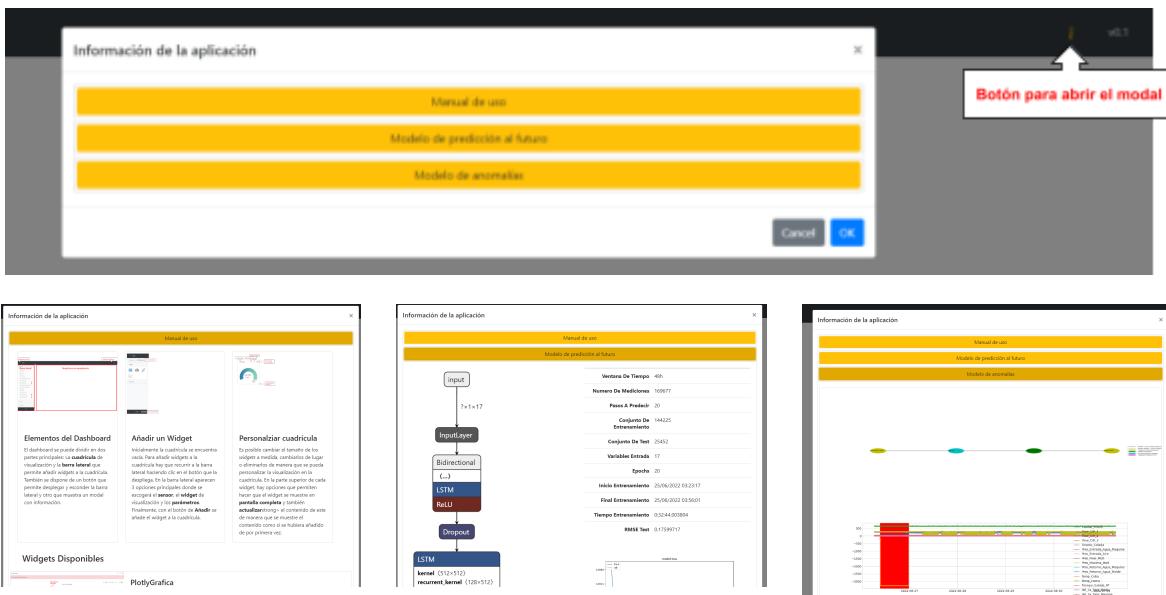


Figura 4.20. Modal con manual de la aplicación e información de los modelos.

Este modal, que se muestra en la Figura 4.20, consta de 3 apartados principales. En el primero se puede encontrar un manual de usuario de la aplicación donde se

explican las diferentes partes y también los *widgets* que hay disponibles. El segundo apartado muestra información del modelo de predicción al futuro. Se pueden observar la arquitectura, una tabla con métricas del último entrenamiento y después gráficas generadas donde se puede ver los resultados en el conjunto de validación y test. Finalmente, el tercer apartado trata el modelo de anomalías y se puede ver el diagrama del modelo, así como los resultados de entrenamiento.

Para concluir este apartado, decir que la aplicación permite crear escenarios de análisis y monitorización del estado de los sensores. También permite visualizar los resultados de los modelos y sus últimos entrenamientos para verificar la eficacia teórica. El objetivo futuro es que los ingenieros que trabajen con la aplicación puedan contrastar la información de los modelos con su efectividad real teniendo en cuenta el conocimiento del proceso.

5. TRABAJO PARA FUTURO

Aunque en el proyecto se han descrito e implementado las diferentes etapas de la metodología de minería de datos, todavía hay mucho margen de mejora en cada una de ellas.

Un mejor análisis de negocio podría sugerir nuevas hipótesis a resolver que podrían inducir a mejoras en todos los sentidos. También podrían aparecer nuevas propuestas integrables con el trabajo realizado. Por ejemplo, se podría estudiar y añadir la propuesta de calidad predictiva a la aplicación creada.

Estudiar detenidamente la fundición a presión y también trabajar conjuntamente con ingenieros de producto, puede enriquecer mucho el análisis de datos y preprocesado. Concretamente, en el enfoque multivariado del análisis. Quedan todavía pendientes muchas técnicas y enfoques por estudiar.

En cuanto al modelaje, hay todavía mucho margen de mejora e investigación para optimizar los modelos actuales. El modelo de regresión está basado en *Deep learning*, una rama del aprendizaje automático muy poderosa dónde actualmente hay mucha investigación. Mejorar el modelo de regresión en sí es un reto que daría para varios proyectos nuevos. Por otra parte, el modelo de detección de anomalías también puede ser mejorado, añadiendo más casos de detección y trabajando conjuntamente con ingeniería para supervisar la efectividad de este.

Finalmente, la aplicación también tiene un largo recorrido por delante. Aunque, de momento, es una aplicación de análisis de datos sobre una ventana reducida de tiempo, es posible ir añadiendo módulos y sistemas, de manera que sea una aplicación que también incluya sistemas de control de supervisión y control del proceso. En la Figura 5.1, se ha realizado una propuesta de lo que podrían ser futuras versiones de la aplicación.



Figura 5.1. Roadmap de las próximas versiones de la aplicación.

En la Figura 5.1 se observa que para la versión 1 (v.1) de la aplicación, el objetivo es acabar de completar la aplicación de manera que dé soporte completo a una célula.

Consistiría en añadir todos los sensores e iterar en la minería de datos mejorando todas las etapas. Después añadir también más *widgets* de visualización como tablas, diagramas, etc. Finalmente, asegurar la robustez de la aplicación, creando tests y eliminando posibles errores.

Para la versión 2 (v.2), el objetivo es extender la aplicación a más células de producción y a más piezas. En este caso habría que reiterar nuevamente en los pasos de minería de datos por toda la variabilidad añadida. Otros objetivos son crear sistemas de roles y usuarios y hacer que los dashboards se puedan guardar y compartir. En este punto también es interesante añadir un sistema de alarmas y reportes en la aplicación.

En la versión 3 (v.3), ya se realiza un paso hacia una aplicación más compleja donde se propongan otros enfoques de análisis de datos y la posibilidad de interactuar con los modelos desde la propia interfaz web. El objetivo es crear entornos donde se puedan efectuar análisis de varias variables a la vez y también poder realizar simulaciones mediante los modelos para ver cómo cambiaría el comportamiento ajustando algunas variables. Finalmente, empezar a trabajar aspectos de seguridad.

Finalmente, en la versión 4 (v.4), el objetivo es añadir sistemas de control a la aplicación que permitan interactuar con la célula. Mediante Node-RED, es posible escribir en registros en el PLC Máster, por lo que se podrían tomar ciertas acciones desde la propia aplicación web. Con buenos modelos, sería interesante poder experimentar con la autorregulación de parámetros para la autocorrección del proceso. Otro sistema que sería interesante añadir es un sistema gestor de órdenes de mantenimiento para obtener más datos descriptivos de los fallos y cómo han sido reparadas para poder implementar **mantenimiento prescriptivo** (el próximo paso al mantenimiento predictivo) donde no solo se predicen los fallos, sino que además se recomiendan acciones específicas de mantenimiento.

En general, todavía queda muchísimo por hacer debido a que el potencial de este proyecto es muy grande. Uno de los retos principales será aplicar iterativamente la metodología de minería de datos para asegurar que es posible escalar el problema a toda la variabilidad y opciones añadidas. Por otra parte, añadir todos los sistemas descritos requiere también de una mejor arquitectura y mucho desarrollo.

6. CONCLUSIONES

Este proyecto partía de la idea de implementar la metodología de minería de datos en un proceso productivo real. La metodología ha supuesto un proceso con diversas iteraciones, donde ha sido muy importante trabajar las diferentes etapas de forma ágil y concurrente. Por ejemplo, como se refleja en la planificación temporal (ver apartado [7.1](#)), primero se ha intentado obtener modelos básicos para llevarlos a la aplicación y, posteriormente, se ha realizado el proceso de evaluación y mejora de estos.

Durante el proceso, se han obtenido dos modelos de naturaleza diferente y con resultados también distintos. El modelo de regresión para predicción al futuro, ha supuesto un problema muy complejo y los resultados indican que todavía hay mucho margen de mejora para ser capaz de predecir el comportamiento de los sensores. Por otra parte, en el modelo de clasificación para detección de anomalías se han obtenido mejores resultados teóricos. Aun así, será valioso mejorarlo y verificar su efectividad conjuntamente con los ingenieros de producto.

Generalmente, la parte de minería de datos requiere de mucha más iteración para obtener mejores resultados. Estudiar mejor los fundamentos y explorar con mayor profundidad las técnicas disponibles será fundamental para las decisiones técnicas. Por otra parte, entender mejor el proceso y trabajar conjuntamente con los ingenieros de proceso permitirá un análisis mucho más enriquecedor y, consecuentemente, mejores resultados.

En cuanto, la parte de la aplicación, aun siendo muy limitada al escenario, implementa toda la parte de minería de datos estudiada y se trata de un buen punto de partida para el análisis de los parámetros de proceso. En algunas decisiones de diseño se han tenido en cuenta factores de escalabilidad, por lo que futuramente se podrían añadir más sensores, células, piezas o modelos, por ejemplo.

Desde una perspectiva global, el proyecto ha resultado en una prueba de concepto o punto de partida para la implementación de mantenimiento predictivo. Aunque los resultados no han sido excelentes, el proceso seguido ha sido todo un reto donde he aprendido mucho en todos los aspectos.

La minería de datos es una disciplina muy prometedora donde queda mucho por hacer e investigar. Aunque el contexto de este proyecto es la Industria 4.0, este tipo de metodología se podía aplicar a muchos otros ámbitos y crear todo tipo de aplicaciones.

Finalmente, quiero decir que ha sido una gran oportunidad para iniciarse en la minería de datos. Trabajar en un escenario real me ha enseñado el potencial que tiene la disciplina y me ha empujado a entrar en el Máster de Ciencia de Datos en la Facultat d'Informàtica de Barcelona.

7. GESTIÓN DEL PROYECTO

Toda la gestión del proyecto se ha realizado mediante la herramienta *Notion*. Se ha creado inicialmente un *dashboard* [50] donde se han ido anotando las etapas, tareas, referencias, roles, recursos, etc. En el *dashboard* están todas las figuras que se verán en los siguientes apartados.

En los siguientes apartados se entrará en detalle sobre la planificación temporal del proyecto y de la gestión de costes.

7.1. Planificación temporal

La temporalidad de este trabajo está concentrada entre los meses de febrero y julio de 2022. Al ser un proyecto realizado mediante convenio de cooperación educativa para prácticas académicas externas, ha sido posible aprovechar las horas en la empresa para llevarlo a cabo. El convenio efectuado tenía fecha de inicio el día 25 de febrero de 2022 y fecha de finalización el día 14 de julio de 2022, con un total 484 horas.

Aparte de las horas en la empresa, también se han efectuado horas extra fuera de ella para poder llegar a cumplir los objetivos. La mayoría del tiempo invertido en esas horas extra ha sido dedicado a aprender o a documentar. La gran parte del desarrollo y proceso de minería de datos se ha hecho en la empresa. El total de horas efectuadas ha sido de 590 horas.

Para poder cumplir los objetivos a tiempo, el reto principal ha sido la planificación y la priorización de las tareas. La decisión de centrarse en una versión ultra simplificada del problema, con solo una célula, una pieza específica y seleccionando un subconjunto de los parámetros, ha sido la clave para poder llegar a obtener resultados. Al principio, se empezó el análisis con todos los parámetros, pero era inviable obtener resultados y el proceso resultaba muy complejo, por lo que se simplificó esta parte también. El motivo de esta decisión es que la idea del proyecto era intentar centrarse más en la **metodología**.

All Dates	Calendar	Drafts	+ Add view
Name	Dates		
Exploración de ideas	February 1, 2022 → March 10, 2022		
Análisis de negocio	February 13, 2022 → March 21, 2022		
Reunión de análisis de datos	February 17, 2022		
Reunión para trazabilidad piezas	February 28, 2022		
Recolección de datos	April 2, 2022 → April 10, 2022		
Procesado de datos	April 10, 2022 → April 30, 2022		
Aplicación de modelos	April 30, 2022 → May 20, 2022		
Evaluación	May 20, 2022 → June 1, 2022		
Despliegue de aplicación	June 1, 2022 → June 20, 2022		
Preparar Memoria	June 20, 2022 → July 5, 2022		
Preparar Lectura	July 5, 2022 → July 13, 2022		
Lectura!!! 🎉	July 13, 2022 10:30 AM		

Figura 7.1. Planificación inicial de etapas realizadas.

En la Figura 7.1 se puede observar cuál ha sido la planificación de etapas inicial donde se ha intentado seguir la metodología de minería de datos de forma lineal. No obstante, se han trabajado muchas etapas de manera iterativa para poder llegar a cumplir los objetivos.

Tabla 7.2. Tabla de tareas realizadas con la etapa y el rol que pertenecen.

Name	Status	Hours	Etapa	Rol	Date
Tareas de obtención de información	Completed	160		Student	March 1, 2022 → June 30, 2022
Modelaje	Completed	50	Aplicación de modelos	Data Scientist	May 4, 2022 → June 22, 2022
Integración de herramientas	Completed	50	Despliegue de aplicación	BackEnd Developer	April 6, 2022 → June 25, 2022
FrontEnd Dev	Completed	45	Despliegue de aplicación	Front End Developer	June 1, 2022 → June 25, 2022
FrontEnd: Setup	Completed	30	Despliegue de aplicación	Full Stack Developer	May 20, 2022 → May 31, 2022
Estudiar fuentes de datos	Completed	25	Analisis de Negocio	Data Analyst	March 1, 2022 → March 24, 2022
Memoria	In progress	25	Preparar Lectura	Student	June 5, 2022 → July 1, 2022
Docker: Node RED + InfluxDB (SETUP)	Completed	24	Despliegue de aplicación	BackEnd Developer	March 31, 2022 → April 18, 2022
MLOps	Completed	22	Despliegue de aplicación	ML Enginner	May 20, 2022 → May 31, 2022
Analisis Multivariado	Completed	21	Procesado de datos	Data Analyst	April 6, 2022 → April 18, 2022
Contexto	Completed	20	Aplicación de modelos	Student	March 1, 2022 → March 11, 2022
Evaluación	Completed	18	Evaluación	Data Scientist	June 12, 2022 → June 26, 2022
SETUP Entorno Producción	Completed	18	Despliegue de aplicación	BackEnd Developer	May 15, 2022 → May 19, 2022
Estudiar arquitectura de despliegue	Completed	12	Despliegue de aplicación	BackEnd Developer	March 4, 2022 → March 8, 2022
Estudiar herramientas de análisis	Completed	12	Procesado de datos	Data Analyst	March 4, 2022 → March 8, 2022
Descripción del escenario	Completed	10	Analisis de Negocio	Student	March 12, 2022 → March 15, 2022
Preprocesado	Completed	9	Procesado de datos	Data Scientist	April 22, 2022 → May 4, 2022
FrontEnd Design	Completed	8	Despliegue de aplicación	Front End Developer	June 1, 2022 → June 9, 2022
InfluxDB Schema	Completed	6	Despliegue de aplicación	Data Scientist	April 24, 2022 → April 27, 2022
Machine Learning Crash Course (Googl)	Completed	6		Student	April 3, 2022 → April 7, 2022
Planificación	Completed	5	Analisis de Negocio	Project Manager	February 25, 2022 → February 28, 2022
Descripción del producto mínimo	Completed	5	Analisis de Negocio	Project Manager	April 18, 2022 → April 19, 2022
Webminar Altair Data Analytics	Completed	4		Student	May 30, 2022
Estado del arte	Completed	3	Preparar Lectura	Student	June 28, 2022
Simulación de trazabilidad	Completed	2	Aplicación de modelos	Data Scientist	May 25, 2022 → May 26, 2022
Feature engineering	Not started		Procesado de datos	Data Scientist	
Articulo Resumen	Not started			Student	

+ New

Calculate ▾

SUM 590

En cada etapa se han realizado diferentes tareas. En cada tarea se ha ido apuntando el número de horas de dedicación, el período de tiempo en el que se ha efectuado y el rol que se ha cumplido. En la Tabla 7.2 se desglosan las tareas ejecutadas.

Como se puede observar, no se ha seguido del todo la planificación inicial. Muchas etapas se han hecho de forma concurrente y algunas no se han realizado acorde a la planificación. Por ejemplo, la evaluación de los modelos se ha hecho una vez desarrollada la aplicación. Esto es debido a que se ha priorizado tener los modelos funcionando en la aplicación antes de intentar mejorarlos, ya que es una tarea compleja y costosa.

Para observar mejor la distribución de las tareas en el tiempo, se ha creado el siguiente diagrama de Gantt (Figura 7.3) a partir de la Tabla 7.2 .

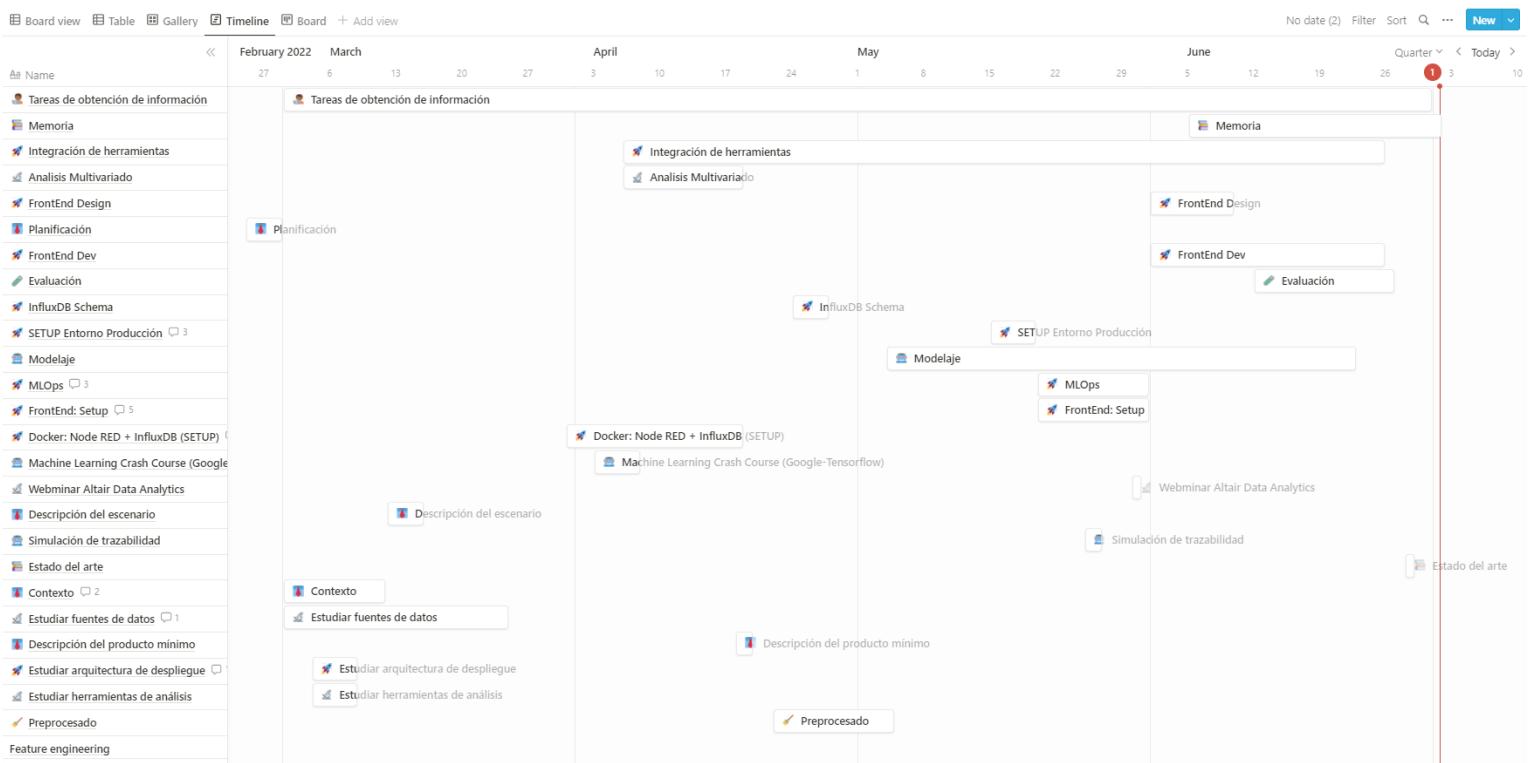


Figura 7.3. Diagrama Gantt de las tareas realizadas (ampliación y leyenda en [Anexo G](#)).

En la Figura 7.3 se muestra que las tareas se han realizado concurrentemente y no se ha seguido del todo la planificación de etapas inicial. Sobretodo, destacar el hecho que se ha empezado con el desarrollo muy temprano respecto a lo previsto y la tarea de evaluación se ha realizado hacia el final del tiempo.

Finalmente, de las tablas se puede extraer que los análisis, procesado, modelaje y evaluación constituyen el 33% del trabajo y la parte de la aplicación es del 35% de las horas totales, por lo que hay un cierto balance de dedicación a los dos objetivos principales. Por otra parte, se ha requerido el 32% para el estudio y realización de documentación (Figura 7.4).

Distribución de horas por etapas

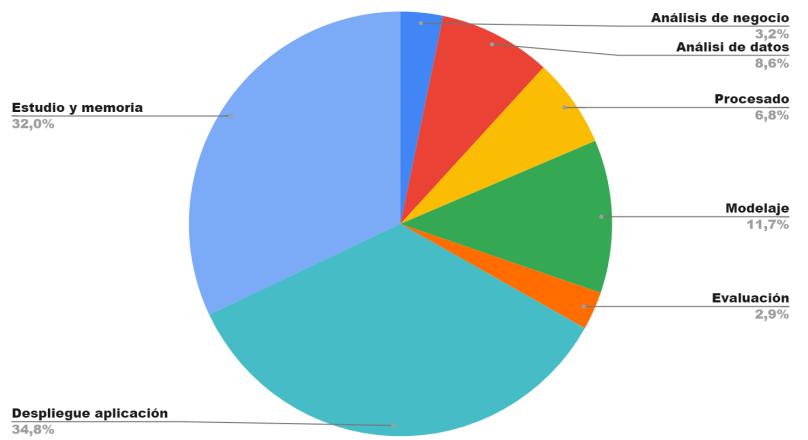


Figura 7.4. Gráfico circular que indica la distribución de horas por etapas.

7.2. Gestión económica

Para la estimación de los costes económicos se ha realizado un sumatorio de los costes humanos y de recursos (hardware y software utilizados).

Para los recursos humanos se ha ido apuntando el rol de cada tarea para poder realizar una estimación (Tabla 7.2). Por cada rol se ha mirado el salario base mediante el portal *glassdoor.es* [51] y se ha efectuado una multiplicación.

Tabla 7.5. Tabla de estimación de coste de recursos humanos.

Rol	Salario base (€/hora)	Horas realizadas	Costes totales (€)
BackEnd Developer	18,75	110	2062,5
Data Analyst	16,42	58	952,36
Data Scientist	19,21	79	1517,59
Front End Developer	16,37	53	867,61
Full Stack Developer	16,82	30	504,6
ML Engineer	23,77	22	522,94
Project Manager	18,72	10	187,2
Student	0	228	0
TOTAL		590h	6614,8€

Para el coste de recursos (Tabla 7.6) de *hardware* se ha tenido en cuenta el coste de un ordenador básico para el desarrollo y un servidor que tenga el hardware de la máquina virtual utilizada (Figura 4.5). Para el *software* se han usado herramientas de código abierto, por lo que ha sido gratuito.

Tabla 7.6. Tabla de estimación de costes de recursos.

	Recurso	Precio
Hardware	PC gama media	400€
	Servidor	1.200€
Software	Python	0€
	Node-RED	0€
	Docker	0€
	InfluxDB	0€
TOTAL		1.600€

Finalmente, realizando el sumatorio de los costes tanto humanos como de recursos, se estima que el presupuesto total del proyecto es el que se muestra en la Tabla 7.7.

Tabla 7.7. Tabla de estimación de costes totales del proyecto.

Costes	Precio Total
Humanos	6.614,80€
Recursos	1.600€
TOTAL	8.214,80€

8. AGRADECIMIENTOS

Primero quiero agradecer a mi familia y a mis amigos por su constante apoyo y cariño. Son mi fuente de inspiración y energía.

Agradecer especialmente a Neus Català i Roig por dirigir este trabajo. Sus consejos, seguimiento y dedicación han sido clave para la realización de este proyecto.

También agradecer a Angel Avilés García por darme esta oportunidad y codirigir este trabajo y a Sebastián Sarrias Manresa por poner a disposición los recursos que necesitaba y sus consejos en diseño web.

Por último, agradecer a la UPC y a CIE Automotive por enseñarme tanto en tan poco tiempo.

9. REFERENCIAS

9.1. Documentos

[1] Tiago Zonta, Cristiano André da Costa, Rodrigo da Rosa Righi, Miromar José de Lima, Eduardo Silveira da Trindade, Guann Pyng Li, *Predictive maintenance in the Industry 4.0: A systematic literature review*, Computers & Industrial Engineering, Volume 150, 2020, 106889, ISSN 0360-8352,
[https://doi.org/10.1016/j.cie.2020.106889.](https://doi.org/10.1016/j.cie.2020.106889)
(<https://www.sciencedirect.com/science/article/pii/S0360835220305787>)

[2] Jovani Dalzochio, Rafael Kunst, Edison Pignaton, Alecio Binotto, Srijnan Sanyal, Jose Favilla, Jorge Barbosa, *Machine learning and reasoning for predictive maintenance in Industry 4.0: Current status and challenges*, Computers in Industry, Volume 123, 2020, 103298, ISSN 0166-3615,
[https://doi.org/10.1016/j.compind.2020.103298.](https://doi.org/10.1016/j.compind.2020.103298)
(<https://www.sciencedirect.com/science/article/pii/S0166361520305327>)

[3] Jordi Torres. Introducción práctica con Keras [en línea] 2018. [Consulta: 28 mayo 2022]. Disponible en: <https://torres.ai/deep-learning-inteligencia-artificial-keras/>

9.2. Referencias Web

[4] Apartado de CIE C.Vilanova en la web de CIE Automotive
<https://cieautomotive.com/-/cie-vilanova>

[5] Web Mindsphere (Siemens)
<https://new.siemens.com/es/es/productos/software/mindsphere.html>

[6] Apartado de mantenimiento predictivo en la web de LimbleCMMS
<https://limblecmms.com/predictive-maintenance-software/>

[7] Definición de mantenimiento predictivo y diferencias con otro tipo de mantenimiento
<https://www.predicagroup.com/blog/predictive-maintenance/>

[8] Especificaciones de la máquina 86 (IDRA OL 1600)
<https://www.diecastmachinery.com/machine-pdfs/idra/idra-1600-cold-chamber-die-casting-machine-dcm-3622.pdf>

[9] Libreta Interactiva utilizada para el análisis y modelado de datos (Google Colab)
https://colab.research.google.com/drive/1v7YI_yHnaAYRnBX9SnEULuJzoBqDPqAH?usp=sharing

[10] Características de las series de datos temporales
<https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-to-time-series-analysis/>

[11] Referencia lenguaje Python

<https://www.python.org/about/>

[12] Definición y teoría del Augmented Dickey-Fuller Test (ADF)

https://es.wikipedia.org/wiki/Prueba_de_Dickey-Fuller_aumentada

[13] Definición y teoría del Kwiatkowski-Phillips-Schmidt-Shin Test (KPSS)

<https://www.statisticshowto.com/kpss-test/>

[14] Teoría de los modelos ARIMA Y SARIMA con ejemplos implementados en Python

<https://neptune.ai/blog/arima-sarima-real-world-time-series-forecasting-guide>

[15] Teoría y definición del coeficiente de correlación de Pearson

https://es.wikipedia.org/wiki/Coeficiente_de_correlaci%C3%B3n_de_Pearson

[16] Referencia módulo Seaborn de Python

<https://seaborn.pydata.org/>

[17] Teoría y ejemplos de Análisis de componentes Principales

https://www.cienciadedatos.net/documentos/35_principal_component_analysis

[18] Teoría y definición del algoritmo K-Means

https://en.wikipedia.org/wiki/K-means_clustering

[19] Método *elbow* para la selección del número de clústeres en K-Means

[https://en.wikipedia.org/wiki/Elbow_method_\(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))

[20] Referencia módulo scikit-learn de Python

<https://scikit-learn.org/stable/>

[21] Explicación de interpolación lineal y definición matemática

<https://x-engineer.org/linear-interpolation-extrapolation-calculator/>

[22] Explicación y pasos comunes en la ingeniería de características

<https://www.heavy.ai/technical-glossary/feature-engineering#:~:text=Feature%20engineering%20refers%20to%20the,machine%20learning%20or%20statistical%20modeling.>

[23] Algoritmo Backpropagation para Redes Neuronales

<https://en.wikipedia.org/wiki/Backpropagation>

[24] Referencia módulo Tensorflow Keras de Python

<https://www.tensorflow.org/guide/keras/functional?hl=es-419>

[25] Explicación de Redes Neuronales tipo LSTM

<https://colah.github.io/posts/2015-08-Understanding-LSTMs/>

[26] Arquitecturas de Redes Neuronales Recurrentes consideradas

<https://machinelearningmastery.com/how-to-develop-lstm-models-for-time-series-forecasting/>

[27] Visualizador de arquitecturas de redes neuronales

<https://netron.app/>

[28] Referencia módulo ADTK de Python

<https://arundo-adtk.readthedocs-hosted.com/en/stable/api/modules.html>

[29] Definición y guía de implementación del modelo Isolation Forest

<https://www.analyticsvidhya.com/blog/2021/07/anomaly-detection-using-isolation-forest-a-complete-guide/>

[30] Definición y explicación de la métrica error cuadrático medio (RSME)

[https://acolita.com/que-es-el-error-cuadratico-medio-rmse/#:~:text=El%20error%20cuadr%C3%A1tico%20medio%20\(RMSE,estad%C3%ADsticas%20m%C3%A1s%20utilizadas%20en%20SIG.](https://acolita.com/que-es-el-error-cuadratico-medio-rmse/#:~:text=El%20error%20cuadr%C3%A1tico%20medio%20(RMSE,estad%C3%ADsticas%20m%C3%A1s%20utilizadas%20en%20SIG.)

[31] Referencia a Node-RED

<https://nodered.org/>

[32] Referencia Node JS

<https://nodejs.org/es/>

[33] Tutorial y explicación de Node-RED

<https://www.pickdata.net/es/noticias/node-red-programacion-visual-iot>

[34] Referencia al módulo UIBuilder de Node-RED

<https://flows.nodered.org/node/node-red-contrib-uibuilder>

[35] Referencia al framework web VueJS (v2)

<https://es.vuejs.org/v2/guide/index.html>

[36] Introducción y ejemplos de Python

<https://www.freecodecamp.org/espanol/news/curso-intensivo-de-python-para-programadores-que-no-usan-python/>

[37] Referencia a InfluxDB

<https://www.influxdata.com/>

[38] Referencia a Docker

<https://www.docker.com/>

[39] Referencia a Debian

<https://www.debian.org/>

[40] Referencia a Linux Alpine

<https://www.alpinelinux.org/>

[41] Referencia a Socket.IO (Sockets WEB)

<https://socket.io/>

[42] Referencia al módulo de conexión con dispositivos PLCs S7 desde Node-RED

<https://flows.nodered.org/node/node-red-contrib-s7>

[43] Referencia al módulo de conexión con InfluxDB desde Node-RED

<https://flows.nodered.org/node/node-red-contrib-influxdb>

[44] Referencia al módulo de machine learning de Node-RED

<https://flows.nodered.org/node/node-red-contrib-machine-learning>

[45] Referencia al módulo que permite visualizar imágenes en Node-RED

<https://flows.nodered.org/node/node-red-contrib-image-tools>

[46] Referencia a la librería que integra Boostrap en VueJS

<https://bootstrap-vue.org/>

[47] Boceto inicial de la aplicación web realizado con la aplicación Figma

<https://www.figma.com/file/TJlhKrVlzzs6Zu8iCXIJdw/SPCAPP?node-id=0%3A1>

[48] Teoría y definición del Test de Causalidad de Granger

https://es.wikipedia.org/wiki/Causalidad_de_Granger

[49] Guía para la implementación realizada para los vectores autorregresivos

<https://www.machinelearningplus.com/time-series/vector-autoregression-examples-python/>

[50] Dashboard creado mediante la herramienta Notion para la gestión del proyecto

<https://aacraf.notion.site/TFE-b75f1fd04ea64fc2a85d2989df0576d9>

[51] Referencia a la plataforma utilizada para consultar sueldos

<https://www.glassdoor.es/Sueldos/index.htm>

[52] Conceptos básicos de ayuda y ejemplos de CRISP-DM

<https://www.ibm.com/docs/es/spss-modeler/SaaS?topic=dm-crisp-help-overview>

10. ANEXOS

ANEXO A: FUNDICIÓN A PRESIÓN

En este anexo se exponen los conceptos y procesos generales de la fundición a presión. Este resumen con todas las figuras que lo forman ha sido redactado a partir de la formación interna “Técnicas de Fundición a Presión - Nivel 2” (J.Montes, 2009).

En general, la fundición de presión es un proceso cuyo concepto es bastante simple, pero la mejora y el control del proceso requiere de mucho estudio y experiencia. En este resumen se podrán ver los elementos principales que forman parte de este proceso. Estos son la máquina, el molde, el material y los parámetros de proceso.

Máquina

La producción de piezas es posible gracias a las máquinas de fundición a presión que son capaces de producir piezas de forma masiva.

Es importante recalcar que, normalmente, las máquinas forman parte de una célula de producción. Es decir, aparte de la máquina, hay un equipo auxiliar sobre la máquina o a sus alrededores que forman parte del proceso o lo complementan.

Las máquinas de producción a presión están constituidas principalmente por dos grupos: el grupo de inyección y el grupo de cierre.

- **Grupo de Inyección**

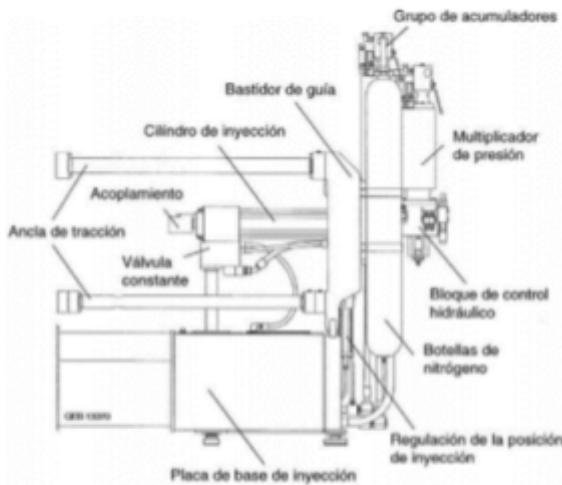


Figura 9.1. Esquema unidad de inyección en la máquina inyectora.

El grupo de inyección (Figura 9.1) es uno de los órganos vitales de la máquina. De él depende la regulación de los parámetros de proceso y la calidad de la pieza fundida. Es relevante prestar atención a los parámetros del grupo de inyección para poder producir piezas buenas.

El proceso en el que interviene este grupo se puede dividir principalmente en **3 fases**. En cada fase, varios elementos actúan de manera simultánea y bajo unos parámetros. La descripción del proceso de este grupo es la siguiente:

- **1 Fase (Fase de aproximación):** En esta fase, la válvula P_2 (Figura 9.2) se abre completamente para permitir el fluido hidráulico que se encuentra en el acumulador A. Este acumulador es accionado por la bomba y controlado por la válvula proporcional del circuito instalada en el distribuidor de cierre. El fluido hidráulico aflora a través de la válvula de retención R . Durante el movimiento acelerado del pistón, se forma una ola que acaba siendo plana. Si la ola no consigue ser plana puede provocar **porosidad** en la pieza. Por eso, es relevante regular adecuadamente los parámetros para evitar estos efectos. El control del fluido hidráulico y de la **velocidad** es importante.

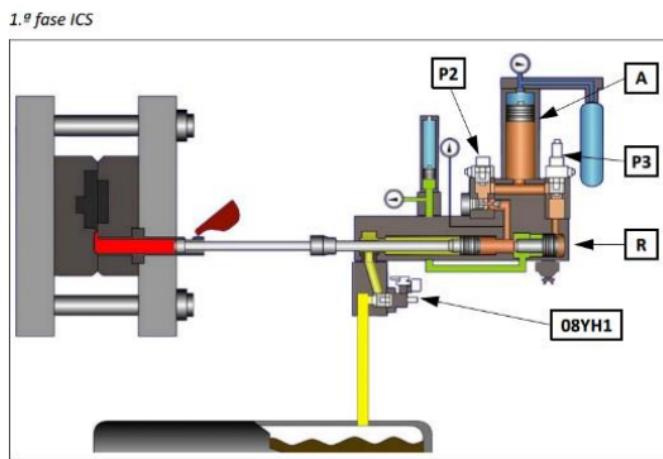


Figura 9.2. Esquema de la primera fase de la inyección (fase de aproximación).

Esta fase debe configurarse de manera adecuada para permitir el alcance de la cota de cierre del orificio de alimentación del contenedor, sin causar turbulencias en el material.

- **2 Fase (Fase de velocidad):** Durante la 1a fase, el pistón recorre la distancia hasta alcanzar la posición “final de carrera de 2a fase”. Entonces, la válvula P_2 (Figura 9.3) que tiene el valor de velocidad definido es activada y el fluido, contenido en el acumulador, pasa a través de la válvula de retención R dando al pistón la “velocidad de 2a fase”. El molde se llena muy rápidamente con el material fundido empujado por la presión del acumulador.

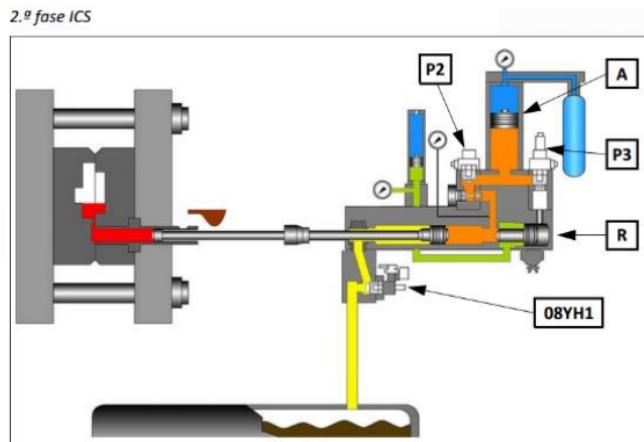


Figura 9.3. Esquema de la segunda fase de inyección (fase de velocidad).

Esta fase es muy importante porque es la que establece el **tiempo de llenado del molde**. El tiempo de llenado del molde es lo que determina en gran medida la calidad de la pieza. Normalmente, cuanto más bajo sea, mejor acabado superficial tendrá la pieza. La mayoría de moldes llenan entre 30 y 100 ms. Este parámetro está muy relacionado con la temperatura del molde, la temperatura del material y el espesor medio de la pieza.

- **3a fase (Fase de multiplicación):** Esta fase empieza al final del llenado del molde. Se activa la válvula proporcional P3. Inmediatamente o después de un retardo de 3a fase, se mueve el pistón multiplicador M. Cuando el cilindro de inyección alcanza la presión del acumulador, la válvula de retención R se cierra de modo automático y la presión aumenta por efecto del multiplicador hasta el valor requerido. Con esto se consigue compactar el material para alimentar la solidificación del material

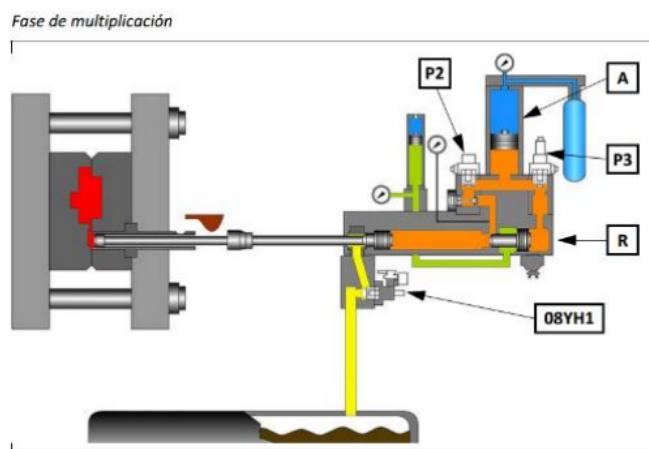


Figura 9.4. Esquema de la tercera fase de inyección (Fase de multiplicación).

Una vez acabado el tiempo de solidificación, se retira la presión del cilindro de inyección, se abren los cilindros radiales de la parte fija, se abre el plato móvil y se abren los cilindros radiales de la parte móvil. Con este proceso se abre el molde y el robot extrae la pieza del molde. Con la pieza extraída se lubrifica el molde y el cilindro inyector retrocede.

Un recurso muy interesante para optimizar los parámetros de inyección es **el análisis de las curvas de inyección** (Figura 9.5). Estas curvas reflejan los datos de los parámetros durante la inyección y se pueden contrastar con datos teóricos de ingeniería o ajustes debido a experiencia con piezas similares.

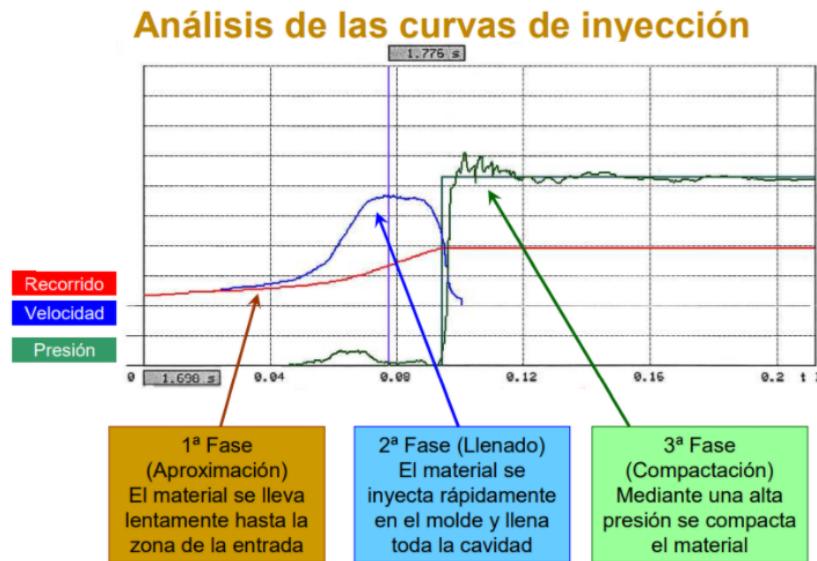


Figura 9.5. Ejemplo de curva de inyección.

El funcionamiento del pistón de inyección afecta directamente a la velocidad y la presión, así como otros sub-parámetros de la máquina. Por lo tanto, es muy importante asegurar que la **cámara de presión** se encuentre en el mejor estado posible.

Una manera de mejorar la cámara de presión es la utilización de **lubrificador** que permite un mejor deslizamiento del pistón. No obstante, el uso de este se vuelve un reto debido a que afecta nuevamente a otros parámetros de inyección. El alto consumo de lubricante y la relativa presencia del mismo lubricante en la pieza tiene consecuencias directas en el aumento de porosidad, por lo tanto, es un parámetro que hay que regular constantemente.

La otra parte fundamental de la máquina es el grupo de cierre. Se trata de la parte más robusta de la máquina, ya que es la que soporta los mayores esfuerzos durante el llenado del molde y las diferentes maniobras.

- **Grupo de cierre**

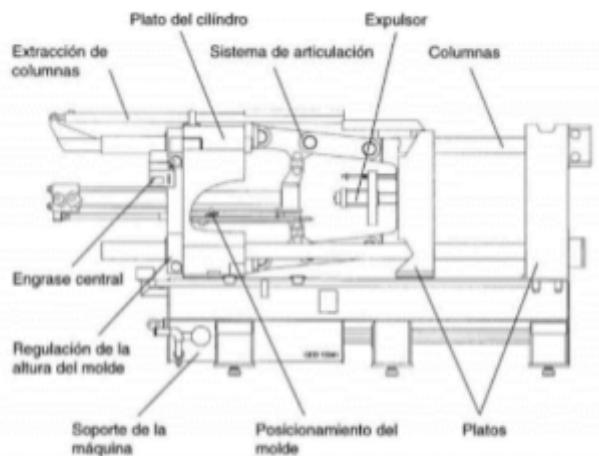


Figura 9.6. Esquema de la unidad de cierre de una máquina de inyección.

Como se puede observar en la Figura 9.6, está formado por: un plato fijo donde va montado el grupo de inyección, un plato de reacción donde va montada la unidad de cierre de la máquina que soporta la fuerza de la inyección y, finalmente, el plato móvil que es donde se monta la parte móvil del molde y el grupo de expulsión de la máquina.

En este caso es importante tener en cuenta la fuerza de cierre para que la distribución de fuerzas sea equitativa y reducir rebabas y rechazo. Esta fuerza es la encargada de cerrar las dos partes del molde y aguantar la inyección del pistón.

Molde

Uno de los principales retos de la fundición es el diseño y mantenimiento de moldes. Se trata de un área competitiva que está relacionada directamente con la calidad de la pieza. El plano de la pieza debe recoger todos los requerimientos y especificaciones, ya que se trata del contrato con el cliente.

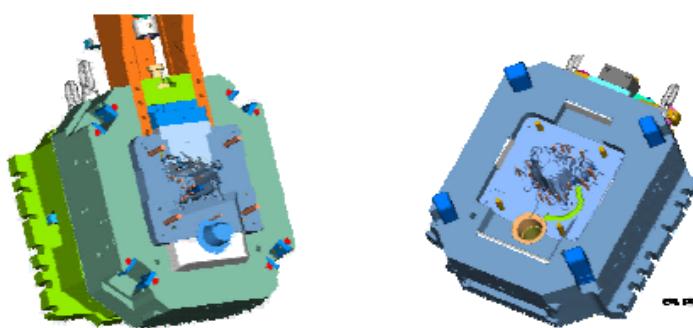


Figura 9.7. Ejemplo de diseño de la geometría del molde.

En el molde es donde se forma la pieza. Con esta premisa, es importante controlar la solidificación de la aleación, así como los fenómenos que se producen en el material que afectan directamente a la calidad de la pieza (poros, rechupes, rebabas, etc.). Ingeniería debe afrontar los siguientes retos principales:

- **Patrón de flujo de llenado**

- Es muy importante estudiar qué patrón de flujo sigue la aleación al ser inyectada dentro del molde. Este patrón afecta mucho a la calidad de la pieza en todos los sentidos y depende del estudio de muchos factores.
- Algunos de los factores principales son: la velocidad de llenado, el estudio de la zona de llenado y el estudio del proceso. Todo el estudio debe estar contrastado con los requisitos de calidad (precisión dimensional, acabado superficial, etc.)

- **Precisión dimensional de la pieza**

- Para obtener piezas con una buena precisión dimensional, es relevante tener en cuenta diferentes factores.
 - Factores de geometría de la pieza
 - Tamaño, forma, rigidez, espesor general, correderas, noyos, nervios, radiado general, etc.
 - Factores de posición de la pieza en el molde
 - Precisión en la parte fija del molde y consideraciones de la afectación de la parte móvil.
 - Factores dependientes del molde
 - Calidad de construcción, elementos (noyos, expulsores, correderas), uso del molde, etc.
 - Factores dependientes de la aleación
 - Colabilidad, temperatura, intervalo de fusión, etc.
- Es crucial controlar que el cierre del molde sea correcto y regular los parámetros como la presión de llenado y la compactación.
- Una buena práctica es establecer unos **límites de tolerancia y de control** para controlar la precisión de las mediciones.

- **Temperatura del molde y evacuación del calor**

- Se tiene que estudiar el coeficiente de transmisión de calor teniendo en cuenta factores como la forma, acabado superficial, lubrificación y estado del óxido del acero. Se trata de un factor determinante para la evacuación del calor del material fundido, el llenado del molde y la solidificación de la pieza.
- La temperatura del molde es importante tanto para la calidad de la pieza, como para la vida del molde y otros parámetros de proceso. En la Tabla 9.8 se pueden visualizar algunas relaciones entre la temperatura del molde y la velocidad de llenado.

Tabla 9.8. Tabla de relaciones entre la temperatura de molde, temperatura de material y la velocidad de llenado.

Velocidad del pistón	Temperatura del material	Temperatura del molde	Condiciones del llenado
OK	OK	OK	OK
Baja	OK	OK	Llenado pobre
Alta	OK	OK	Rebabas/ Porosidad
OK	Baja	OK	Llenado pobre
OK	Alta	OK	Rebabas/ Soldadura
OK	OK	Baja	Llenado pobre
OK	OK	Alta	Soldadura

- La cantidad de evacuación de calor depende del material del molde, el tipo de líquido de enfriamiento, el paso y la velocidad del líquido, el diámetro de los canales del circuito, la distancia de los canales a la superficie de la cavidad, etc.
- También es relevante entender que el proceso puede afectar a la fatiga térmica del acero, rindiendo negativamente en la vida del molde. Encontrar el equilibrio térmico del molde es otro reto.
- Existen diferentes técnicas y tecnologías para el enfriamiento del molde (*JetCool*, *Co2*, etc.). El uso dependerá del estudio del proceso.

- **Factores de evacuación de aire y control de vacío en el molde**

- La evacuación del aire es otra tarea relevante porque el aire puede afectar a la aparición de poros y otros defectos de calidad. Lo ideal es que durante el llenado, el aire sea empujado hacia el exterior mediante aberturas suficientemente dimensionadas.
- El empleo de *rebosaderos* permite la ventilación del molde y la extracción de aire.
- A veces, el uso de rebosaderos es insuficiente para la evacuación correcta del aire, por lo que también se utiliza el vacío (Figura 9.9) para evacuar totalmente el aire del molde y así poder obtener piezas aceptables.

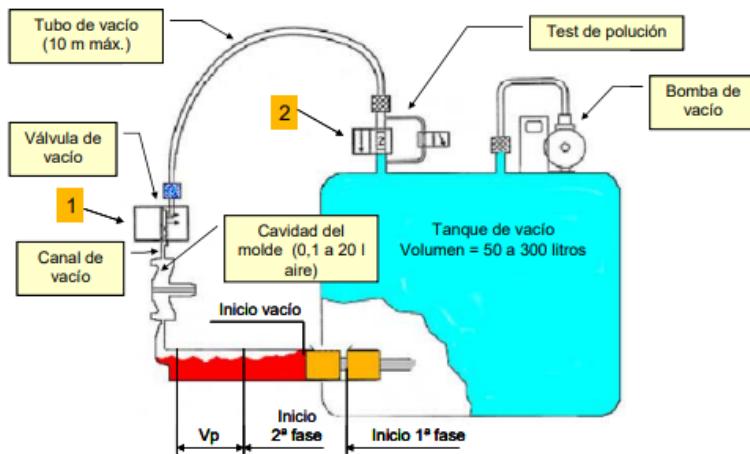


Figura 9.9. Esquema de un tanque de vacío para la evacuación del aire del molde.

- **Parámetros de proceso**

- Los parámetros de proceso son vitales tanto para la máquina como para el molde. Su estudio y regulación es una de las principales tareas en la fundición a presión.

Poner a punto un molde requiere seguir un protocolo que permita rebajar al máximo los riesgos que puedan afectar a la calidad de la pieza, a la máquina o a la propia vida del molde.

El procedimiento estándar para poner a punto un molde se resume en los siguientes pasos:

1. **Evaluar riesgos de calidad**

Es importante estudiar los requisitos y asegurar que estos contengan todo el pliego de condiciones de calidad. Identificar adecuadamente los problemas es una de las grandes dificultades. Los **defectos típicos** de la fundición a presión se pueden ver en la Tabla 9.10.

Tabla 9.10. Tabla clasificatoria de los defectos típicos.

Defectos de estructura	Defectos internos	Defectos superficiales
Desviaciones de medidas	Porosidad por gas	Ampollas superficiales
Formación de rebabas	Inclusiones	Falsas soldaduras, gotas frías

Llenado del molde incompleto	Rechupes	Marcas de los expulsores
Variaciones de volumen		Roces de expulsión
Deformaciones		Rechupe / depresión superficial
		Soldaduras
		Superficie jaspeada
		Erosión / Agrietamiento en caliente

2. Respetar reglas fundamentales de la fundición a presión (definición y parámetros de proceso)

Aunque se llegue a trabajar con proyectos muy complejos que requieran mucha exigencia técnica y de calidad, es muy importante tener en cuenta en todo momento los **principios básicos** de la fundición. Algunas de las reglas son:

- Optimizar el llenado de la pieza para evitar aire atrapado y rechupes por contracción.
- Favorecer el flujo de la aleación por las paredes del molde.
- Desplazar los efectos de solidificación a los rebosaderos para evitar defectos superficiales y porosidad.
- Trabajar con material limpio y no oxidado para obtener piezas con mejor aspecto (mantenimiento de la máquina).
- Buscar equilibrio térmico del molde.
- Aplicar el desmoldeante de un modo uniforme en las partes del molde, dejando la superficie del molde seca pero lubricada.

3. Conocer el estado de los medios afectados (capacidad de proceso)

- Un breve análisis de la geometría de la pieza para conocer las limitaciones del proceso. El criterio es tratar de que la solidificación de la pieza sea óptima y con una variación mínima.
- Estudiar los requisitos y posibles afectaciones es una tarea fundamental para obtener el mejor rendimiento posible.

4. Parámetros de proceso

Conociendo los requisitos y las limitaciones del proceso, será importante ajustar los parámetros de proceso siguiendo algunas de las siguientes normas:

- Definir el tiempo de llenado de la cavidad del molde.
- Determinar la velocidad de aleación en el ataque.
- Respetar las **reglas fundamentales** de la fundición a presión: evitar atrapos de aire y equilibrar el grado de llenado y la velocidad del pistón.
- Evitar picos de presión reduciendo la velocidad del pistón de inyección.
- En la puesta a punto, identificar correctamente los defectos repetitivos: poros, gas atrapado y rechupes por contracción, etc.
- Tener conocimiento de las referencias teóricas y de la posición de las correderas para controlar las carreras.
- Ajustar la fuerza de cierre de la máquina.

La primera exigencia que hay que tener en mente es obtener una producción estable durante un periodo de tiempo. La segunda exigencia es que si surge algún desvío respecto a la especificación, pueda ser identificado en la constelación de parámetros de proceso.

5. Fundición de muestras representativas

El ajuste óptimo del proceso sería ideal si después de cada pieza inyectada se pudiera obtener un resultado de calidad. Desgraciadamente, la calidad solo se puede valorar mediante dispositivos que se encuentran lejos de la máquina (Rayos X, máquinas tridimensionales, etc.).

- Con esta premisa, con los parámetros de proceso se asigna la etiqueta de “**probablemente buena**” o “**probablemente mala**” a cada pieza hasta que pase por el control de calidad.
- La exigencia técnica y de calidad de las piezas fundidas es cada vez más alta y la influencia de los parámetros en la calidad de la pieza es difícil de dominar.

6. Documentación y registro de las muestras

- Una etapa de muestras es importante para posicionar correctamente los parámetros de proceso previstos. También es importante observar las piezas obtenidas para valorar aspectos **dimensionales** y **metalúrgicos** obtenidos con los parámetros.
- En caso de encontrar cualquier desviación, es importante aplicar las correcciones apropiadas en los parámetros para obtener la calidad adecuada.

- Las muestras deben ser documentadas para poder evaluar la evolución de los parámetros y la calidad obtenida. Toda información recogida en el proceso es importante para mejorar.

7. Ajuste de la calidad

- En este estado conviene revisar ajustes de la máquina, pesos, dimensiones, material, temperatura, configuraciones de refrigeración, lubricador, etc.
- La experiencia y los buenos hábitos son la clave para que los ajustes ayuden a mejorar el proceso.

8. Diagnóstico de defectos

- En la puesta a punto de un molde nuevo, todos los problemas de calidad han de ser registrados y diagnosticados cuidadosamente.
- Los defectos diagnosticados pueden ser estructurales, internos, superficiales o mecánicos. Los defectos son complicados de medir y definir.
- Es importante conocer los niveles de aceptación de la pieza que normalmente están definidos en los requerimientos o pliego de condiciones. Las piezas pueden declararse como "buenas" si superan los niveles de aceptación acordados.

9. Evaluar la calidad

- Las muestras son el examen final de un proyecto.
- La evaluación de las muestras pondrá en evidencia el desarrollo del producto, el concepto del molde, el seguimiento, construcción y calidad de este, el desarrollo del proceso, el sistema de calidad, etc.
- Para evaluar la calidad se pueden utilizar diferentes indicadores. Por ejemplo:
 - La duración
 - Número de pruebas
 - La capacidad del proceso
 - El tanto por ciento de rechazo
 - La fiabilidad de los medios

10. Ajustar y disponer los parámetros para la fabricación en serie

- Una vez se ha pasado el control de calidad y se ha obtenido un resultado satisfactorio, se ajustan los parámetros de proceso y se empieza a fabricar en serie.
- La monitorización y mantenimiento son valiosos durante la fabricación en serie para obtener el máximo rendimiento.

El material

El aluminio es un material blanco brillante, no-ferromagnético con densidad 2,7 y un punto de fusión de 660 grados. En estado puro es blando y maleable aunque es muy oxidable. Es el tercer elemento más común en la tierra con un 8% de la corteza de la tierra. La bauxita es la mena de aluminio más importante, conteniendo entre un 30 y 54% de aluminio.



Figura 9.11. Almacén de reserva de aluminio.

El aluminio, aparte de ser blando y maleable, también es 100% recicitable. Además, hay muchos tipos de aluminio comercializables. Los cuatro grupos principales son productos laminados, planchas de imprentas, papel de aluminio y la “chatarra”.

Aunque son muchos los beneficios del aluminio, en sí el material posee una resistencia muy baja y poca dureza. Eso hace que sea necesario crear aleaciones que aporten esas características.

Las dos aleaciones utilizadas en la planta CIE C.Vilanova son:

- **GD-AISi9Cu3.**- Aleación universal de muy buena colabilidad. Caracterizada por una menor tendencia a producir cavidades superficiales y rechupes internos. Muy buena maquinabilidad con arranque de virutas. Especialmente buena para fundición a presión. Aleación de usos múltiples, también para piezas con formas complejas.
- **GD-AISi12 (Cu).**- Aleación eutéctica con excelente facilidad para el llenado del molde. Alta resistencia al agrietamiento en caliente y excelentes propiedades de colada. Para piezas fundidas complejas y de paredes delgadas.

Hay que mantener exactamente los valores en % de todos los elementos de la aleación, porque solamente así se pueden obtener resultados que soporten las exigencias mecánicas definidas en los requerimientos técnicos.

El aluminio requiere mucho poder calorífico para pasar de sólido a líquido (**fusión**). La aleación durante esta fase produce algunas impurezas que hay que ir eliminando para obtener el material lo más puro posible. También hay que evitar que el aluminio se oxide regulando correctamente el calor y evitando cualquier factor que pueda producir el fenómeno. En caso de que haya impurezas, también se puede recurrir a la refinación del material, dejándolo reposar durante mucho tiempo a baja temperatura, utilizando tratamientos con cloros o barridos con nitrógeno o/y utilizando técnicas de filtraje.

Otro fenómeno muy importante en la fundición a presión es la **solidificación** del material. El estudio químico de la cristalización de átomos es importante para obtener el material con unas buenas características mecánicas. Las muestras son fundamentales para comprobar que la composición atómica sea correcta.

Tener una buena aleación es muy importante porque todas las impurezas arrastradas se acentúan en los procesos posteriores. A través de muestras, y con la continua regulación de parámetros como temperaturas, se reducen los posibles defectos que se pueden producir por el material (rechupes, arrastres, uniones frías, etc.).

Parámetros de proceso

Una vez entendida la máquina, las exigencias del molde y las propiedades del material, es vital tener en cuenta los parámetros de proceso que están directamente relacionados con estos. Los parámetros de proceso afectan tanto a la calidad de la pieza como a la propia vida de la máquina y al molde. El estudio y control de estos es una de las tareas principales de ingeniería.

Es crucial recalcar que estos parámetros están interrelacionados entre sí y la variación de uno afecta directamente a otros. Con esta premisa, entender y comprender los vínculos entre diversos parámetros es importante para el dominio y el control del proceso.

Es importante recordar las exigencias de calidad y dominio de proceso:

- **Primera exigencia - Estabilidad:** Es fundamental que la especificación del proceso proporcione piezas buenas durante un periodo de tiempo prolongado.
- **Segunda exigencia - Control:** Sobre la primera exigencia, cuando surja un desvío respecto la especificación, pueda ser detectado como una desviación de algún parámetro dentro de la constelación de parámetros del propio proceso.

Para cumplir estas dos exigencias es necesario mantener un sistema para asegurar la calidad y regularidad de los ciclos y mantener bajo control los parámetros. Una buena metodología es establecer límites de tolerancia y control en los parámetros para así tenerlos bajo control. Aplicaciones de SPC (*Statistical Process Control*) pueden ayudar a estudiar el comportamiento y evolución de los parámetros.

Aparte de tener una buena sistemática, es fundamental el conocimiento teórico y experiencia en el ámbito para poder entender las relaciones entre estos. La mayoría de los fenómenos físicos que se producen durante el proceso tienen explicación matemática y pueden ser detallados. Los parámetros principales de proceso son los que se muestran en la Tabla [9.12](#).

Tabla 9.12. Tabla de los parámetros principales de proceso.

Parámetro	Descripción
Velocidad 1a Fase	Esta es la velocidad que coge el pistón inicialmente. Se trata de una velocidad que tiene que evitar la formación de olas.
Velocidad de la 2a Fase (tiempo de llenado)	Es el parámetro más importante, puesto que tiene relación directa con la calidad de la pieza. La mayoría de moldes llenan entre 30 y 100 milisegundos. Este tiempo

Presión de llenado	El llenado del molde es ideal cuando se consigue llenar la totalidad del molde y expulsar completamente el aire.
Presión de multiplicación (compactación)	Este parámetro se debe definir en la fase de diseño teniendo en cuenta factores como la geometría de la pieza. Al final de la fase de llenado hay que utilizar una presión suficientemente grande para que el material se contraiga.
Control térmico molde	La temperatura del horno de mantenimiento afecta al precalentamiento del horno, afectando así a todos los demás parámetros dependientes. Es muy importante tener un control térmico del molde para evitar problemas de calidad
Dosificación del material	Se trata de las unidades en las que se dosifica el material para ser inyectado. Depende del proyecto puede ser diferente.
Tiempo apertura del molde	Después de inyectada, el molde requiere de un tiempo para abrirse para poder evacuar correctamente.
Tiempo de expulsión	El tiempo en el que se extrae la pieza
Lubrificación del molde	Aparte de tener la función de lubricar, también tiene función refrigeradora y separadora. Por lo tanto, afecta a la temperatura del molde, a la extracción de la pieza y a la velocidad del pistón

En la tabla se encuentran algunos de los parámetros más relevantes. No obstante, la cantidad de parámetros es mucho mayor y puede crecer más con sensores adicionales. Cuanto más información haya del proceso, mejor será la descripción y esto puede conducir a mejoras de rendimiento y calidad.

Cabe recalcar que la constelación de parámetros es tan grande que conviene tener optimizados al máximo la máquina y el molde para así tener la máxima tolerancia a variaciones de estos.

ANEXO B: SEÑALES CRÍTICAS

En la siguiente tabla (Tabla 9.13) se puede ver una descripción de las señales críticas definidas por ingeniería y que son recogidas desde el PLC Máster.

Tabla 9.13. Tabla de las señales críticas recibidas desde el PLC Máster.

Señal (Sensor)	Descripción
Caudal_Piston	El pistón lleva por el interior un circuito de refrigeración , esta variable es el caudal de agua que pasa por ese circuito.
Flow_CIR_1	Hace referencia a la primera boquilla del cabezal del lubricador por donde pasan los litros de agua+desmoldante.
Flow_CIR_2	Hace referencia a la segunda boquilla del cabezal del lubricador por donde pasan los litros de agua+desmoldante.
Flow_CIR_3	Hace referencia a la tercera boquilla del cabezal del lubricador por donde pasan los litros de agua+desmoldante.
Pres_Entrada_Agua_Máquina	El circuito de agua es general en toda la fábrica, esta variable nos da la presión que hay en la línea justo en el bajante que va a la máquina.
Pres_entrada_aire	General en toda la fábrica, esta variable nos da la presión que hay en el circuito de aire que va a la máquina.
Pres_Final_Multi	Esa variable es de la fase multiplicación, que es la 3 ^a fase de la inyección, cuando se aplica una presión al aluminio para compactar la pieza. Esa presión se aplica durante un tiempo, mientras el aluminio aún está líquido, esta variable indica la presión justo al final del proceso.
Pres_Maxima_Mult	Esa variable es de la fase multiplicación, que es la 3 ^a fase de la inyección, cuando se aplica una presión al aluminio para compactar la pieza. Esa presión se aplica durante un tiempo, mientras el aluminio aún está líquido, esta variable indica la presión máxima alcanzada durante el proceso.
Pres_Retorno_Agua_Maquina	El circuito de agua es general en toda la fábrica, tenemos la presión en las entradas (variable 5) y esta variable nos da la presión que hay en la línea a la salida de la máquina.
Pres_Retorno_Agua_Molde	El circuito de agua es general en toda la fábrica, tenemos la presión en las entradas (variable 5) y esta variable nos da la presión que hay en la línea a la salida del molde.

Temp_Cuba	Temperatura del agua de la cuba donde se enfriá la pieza al salir del molde
Temp_Horno	Temperatura del aluminio en el horno mantenedor.
Tiempo_Subida_RT	Cuando acaba la segunda fase, esta variable es el tiempo que tarda en coger la presión de multiplicación (3 ^a fase) que se ha programado.
Vel_1 ^a _Fase_Media	La inyección tiene 3 fases, en este caso se trata de la primera en la fase en la que el pistón se mueve lentamente para no generar turbulencia y lleva el aluminio hasta la entrada de material a la pieza, se llena el canal. Esta variable es la velocidad media que ha tenido el pistón durante ese tiempo de aproximación (normalmente se programa por recorrido de pistón)
Vel_1 ^a _Fase_Maxima	La inyección tiene 3 fases, en este caso se trata de la primera en la fase en la que el pistón se mueve lentamente para no generar turbulencia y lleva el aluminio hasta la entrada de material a la pieza, se llena el canal. Esta variable es la velocidad máxima que ha tenido el pistón durante ese tiempo de aproximación (normalmente se programa por recorrido de pistón)
Vel_2 ^a _Fase_Media	La inyección tiene 3 fases, en este caso se trata de la segunda en la que el pistón va mucho más rápido y es la fase en la que se llena la pieza. Esta variable es la velocidad media que ha tenido el pistón durante ese tiempo de aproximación (normalmente se programa por recorrido de pistón)
Vel_2 ^a _Fase_Maxima	La inyección tiene 3 fases, en este caso se trata de la segunda en la que el pistón va mucho más rápido y es la fase en la que se llena la pieza. Esta variable es la velocidad máxima que ha tenido el pistón durante ese tiempo de aproximación (normalmente se programa por recorrido de pistón)

ANEXO C: VECTORES AUTORREGRESIVOS

Los vectores autorregresivos son modelos multivariados que se basan en la premisa de que todas las variables se influencian entre ellas. Por lo tanto, cada variable vendrá definida como una ecuación de combinaciones lineales que incluyen las otras variables. Concretamente, estas ecuaciones tienen la siguiente forma:

$$\begin{aligned}y_{1,t} &= c_1 + \phi_{11,1}y_{1,t-1} + \phi_{12,1}y_{2,t-1} + e_{1,t} \\y_{2,t} &= c_2 + \phi_{21,1}y_{1,t-1} + \phi_{22,1}y_{2,t-1} + e_{2,t},\end{aligned}$$

Figura 9.14. Ecuaciones que definen los vectores autorregresivos.

En la fórmula (Figura 9.14) se puede observar que $y_{1,t}$ (y_1 en el instante t) viene definida por su valor más la suma de las diferentes variables con un coeficiente $\phi_{ij, \ell}$ que captura la influencia de éstas. Finalmente, se añade un ruido blanco ($e_{1,t}$) que viene contemporáneamente correlacionado.

Para poder implementar correctamente este modelo, es importante tener en cuenta algunos pasos de preprocessado. Para el buen funcionamiento del modelo, se requiere que los datos sean estacionarios y que haya correlación entre las variables. Para verificar esto se han utilizado los test de causalidad de Granger [48] para las correlaciones y el Augmented Dickey-Fuller Test [12] para verificar la estacionalidad. Se ha aplicado el preprocessado para garantizar estas propiedades.

Finalmente, siguiendo la implementación localizada en [49] los resultados que se han obtenido al aplicar los modelos son los siguientes sobre una predicción de 15 pasos para el futuro:

	mape	me	mae	mpe	rmse	corr	minmax
Caudal_Piston	0.017284	-3.276615	3.276615	-0.017284	3.403252	0.386632	0.017284
Flow_CIR_1	0.000122	0.000093	0.000093	0.000122	0.000119	NaN	0.000122
Flow_CIR_2	0.000007	-0.000014	0.000018	-0.000005	0.000019	NaN	0.000007
Flow_CIR_3	0.000005	-0.000011	0.000011	-0.000005	0.000012	NaN	0.000005
Grueso_Colada	0.000452	0.024004	0.024004	0.000452	0.026173	NaN	0.000452
Pres_Entrada_Agua_Maquina	0.022161	0.102132	0.103544	0.021859	0.124963	-0.387076	0.021470
Pres_Entrada_Aire	0.004622	0.029843	0.030030	0.004593	0.037758	0.963142	0.004588
Pres_Final_Mult	0.162918	-49.689985	49.689985	-0.162918	56.977658	NaN	0.162918
Pres_Maxima_Mult	inf	-53.512246	53.512246	-inf	61.268926	NaN	inf
Pres_Returno_Agua_Maquina	0.010935	-0.014784	0.014784	-0.010935	0.016201	NaN	0.010935
Pres_Returno_Agua_Molde	0.007080	0.009612	0.009612	0.007080	0.011664	0.464953	0.007007
Temp_Cuba	0.000051	-0.001372	0.001587	-0.000044	0.001730	NaN	0.000051
Temp_Horno	0.000132	-0.085697	0.085697	-0.000132	0.095520	-0.635797	0.000132
Tiempo_Subida_RT	0.002367	0.000002	0.000002	-0.002300	0.000003	NaN	-0.002376
Vel_1a_Fase_Media	0.192719	-0.054793	0.054793	-0.192719	0.062767	NaN	0.192719
Vel_2a_Fase_Maxima	0.306936	-0.798293	0.798293	-0.306936	0.914617	NaN	0.306936
Vel_2a_Fase_Media	0.285409	-0.658612	0.658612	-0.285409	0.754683	NaN	0.285409

Figura 9.15. Métricas obtenidas de modelar mediante vectores autorregresivos.

Como se puede observar en la Figura 9.15, los distintos errores son bastante reducidos en la predicción. Esto es gracias al preprocessado realizado. No obstante,

teniendo en cuenta el RMSE como métrica principal, hay algunas variables que no han dado buenos resultados. Por ejemplo, la Pres_Final_Mult y la Pres_Maxima_Mult.

Los vectores autorregresivos, aun siendo una opción muy potente, parten de la premisa de que hay correlación entre todas las variables y eso no es algo que siempre se cumpla en los escenarios reales. Por este motivo, y por su escasa escalabilidad, no ha sido la opción escogida para modelar el problema en este proyecto

ANEXO D: MÓDULOS PYTHON

Los módulos de Python instalados en el contenedor de Node-RED son los siguientes (resultado de ejecutar la orden pip list):

Package	Version
absl-py	0.15.0
adtk	0.6.2
aiohttp	3.8.1
aiosignal	1.2.0
astor	0.8.1
astunparse	1.6.3
async-timeout	4.0.1
attrs	21.4.0
blinker	1.4
brotlipy	0.7.0
cachetools	5.1.0
certifi	2022.5.18.1
cffi	1.15.0
charset-normalizer	2.0.4
clang	5.0
click	8.0.4
conda	4.12.0
conda-package-handling	1.8.1
cryptography	3.4.8
cycler	0.11.0
flatbuffers	2.0
fonttools	4.33.3
frozenlist	1.2.0
gast	0.4.0
google-auth	2.6.6
google-auth-oauthlib	0.4.6
google-pasta	0.2.0
graphviz	0.20
grpcio	1.42.0
h5py	3.1.0
idna	3.3
importlib-metadata	4.11.4
influxdb	5.3.1
influxdb-client	1.29.1
joblib	1.1.0
keras	2.8.0
keras-nightly	2.5.0.dev2021032900
Keras-Preprocessing	1.1.2
kiwisolver	1.4.2
libclang	14.0.1
Markdown	3.3.7
matplotlib	3.5.2
mkl-fft	1.3.1
mkl-random	1.2.2
mkl-service	2.4.0
mkl-umath	0.1.1
msgpack	1.0.3
multidict	5.2.0
numpy	1.19.5
oauthlib	3.2.0
opt-einsum	3.3.0
packaging	21.3
pandas	1.4.2
patsy	0.5.2
Pillow	9.1.1
pip	21.2.4

protobuf	3.20.1
pyasn1	0.4.8
pyasn1-modules	0.2.8
pycosat	0.6.3
pycparser	2.21
pydot	1.4.2
PyJWT	2.1.0
pyOpenSSL	21.0.0
pyparsing	3.0.9
PySocks	1.7.1
python-dateutil	2.8.2
pytz	2022.1
requests	2.27.1
requests-oauthlib	1.3.1
rsa	4.8
ruamel-yaml-conda	0.15.100
Rx	3.2.0
scikit-learn	1.0.1
scipy	1.6.2
seaborn	0.11.2
setuptools	61.2.0
six	1.16.0
sklearn	0.0
statsmodels	0.13.2
tabulate	0.8.9
TBB	0.2
tensorboard	2.8.0
tensorboard-data-server	0.6.1
tensorboard-plugin-wit	1.8.1
tensorflow	2.4.1
tensorflow-estimator	2.6.0
tensorflow-hub	0.12.0
tensorflow-io-gcs-filesystem	0.26.0
tensorflowjs	3.18.0
termcolor	1.1.0
tf-estimator-nightly	2.8.0.dev2021122109
threadpoolctl	2.2.0
tqdm	4.64.0
typing_extensions	4.2.0
urllib3	1.26.9
Werkzeug	2.1.2
wheel	0.37.1
wrapt	1.14.1
yarl	1.6.3
zipp	3.8.0

ANEXO E: ARQUITECTURA (docker-compose.yml)

```
- version: "3"
- services:
  - node-red:
    depends_on:
      - influx
    container_name: nodered-spc
    user: root
    build:
      context: ./nodered_image
      dockerfile: Dockerfile
    environment:
      - TZ=Europe/Amsterdam
      - FLOWS=flows.json
    ports:
      - '1880:1880'
    volumes:
      - './nodered-data:/data'
  - influx:
    container_name: influx-spc
    image: 'quay.io/influxdb/influxdb:2.0.0-rc'
    ports:
      - '8086:8086'
    volumes:
      - 'influx-storage:/var/lib/influxdb'
      - './influx-data/init-influxdb.sh:/init-influxdb.sh'
  - influx-setup:
    image: 'julianfh/influxdb-setup:latest'
    depends_on:
      - influx
    environment:
      TIMEOUT: 20
      INFLUXDB_SCHEMA: http
      INFLUXDB_HOST: influx-spc
      INFLUXDB_PORT: 8086
```

```
-      INFLUXDB_TOKEN: cie-token
-      INFLUXDB_BUCKETID: cel86
-      INFLUXDB_ORG: cie
-      INFLUXDB_USER: admin
-      INFLUXDB_PW: verysecurepassword1
-      INFLUXDB_RETENTION: 1d
- volumes:
-   influx-storage:
```

ANEXO F: APLICACIÓN WEB (package.json)

```
{  
  "name": "src",  
  "version": "1.0.0",  
  "description": "",  
  "main": "app.js",  
  "scripts": {  
    "dev": "parcel watch ./src/index.html --public-url ./ --out-dir ./dist/ --no-source-maps",  
    "build": "rm -rf ./dist/* && parcel build ./src/index.html --public-url ./ --no-cache --out-dir ./dist/ --detailed-report --no-source-maps"  
  },  
  "author": "",  
  "license": "ISC",  
  "dependencies": {  
    "apexcharts": "^3.35.0",  
    "bootstrap-vue": "^2.21.2",  
    "chart.js": "^3.7.1",  
    "esbuild": "^0.14.42",  
    "vue": "^2.6.14",  
    "vue-apexcharts": "^1.6.2",  
    "vue-chartjs": "^4.0.7",  
    "vue-hot-reload-api": "^2.3.4",  
    "vue-json-pretty": "^1.8.2",  
    "vue-plotly": "^1.1.0",  
    "vue-router": "^3.5.4"  
  },  
  "devDependencies": {  
    "@snowpack/plugin-vue": "^2.6.2",  
    "@vue/component-compiler-utils": "^3.2.2",  
    "cssnano": "^4.1.10",  
    "less": "^4.1.2",  
    "parcel-bundler": "^1.12.5",  
    "sass": "^1.32.5",  
    "typescript": "^4.7.2",  
    "vue-template-compiler": "^2.6.14",  
    "@vitejs/plugin-vue": "^2.0.0",  
  }  
}
```

```
"vite": "^2.7.2",
"vite-plugin-vue": "^0.0.0",
"vite-plugin-vue2": "^1.9.3"
}
}
```

ANEXO G: DIAGRAMA GANTT AMPLIADO

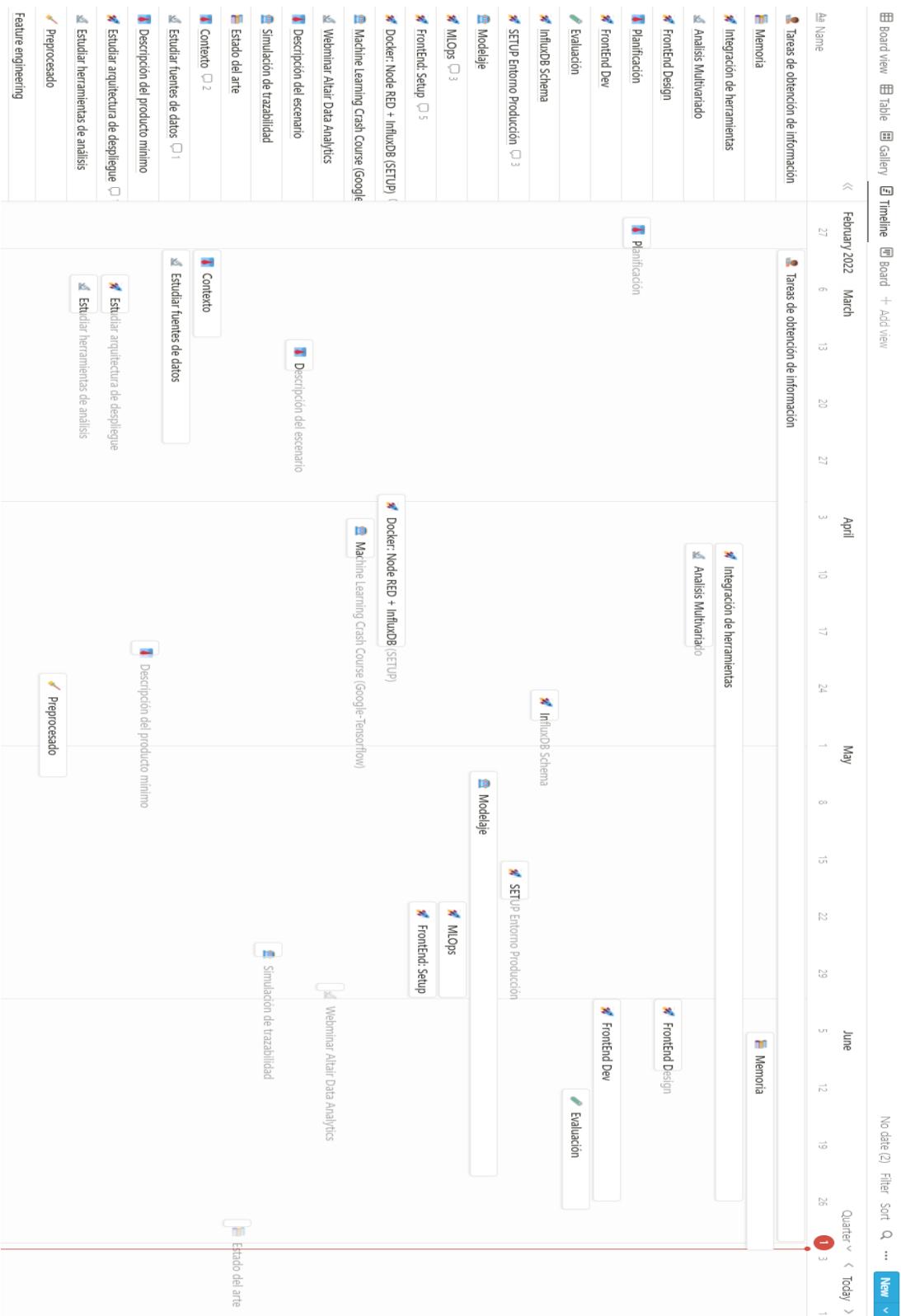


Figura 9.16. Diagrama Gantt ampliado.